

Szoftver- és rendszerellenőrzés

Részletes tervek ellenőrzése

Majzik István

<http://www.inf.mit.bme.hu/>

Tartalomjegyzék

- **Áttekintés**
 - Milyen szerepe van a részletes terveknek?
 - Milyen ellenőrzési módszerek vannak?
- Statikus ellenőrzés felülvizsgálattal
- Formális verifikáció modellellenőrzéssel
 - Módszerek áttekintése
 - Modellellenőrzés mérnöki modellek alapján
- Formális verifikáció ekvivalencia ellenőrzéssel
 - Trace ekvivalencia
 - Megfigyelési ekvivalencia (gyenge biszimuláció)
 - Lehetséges viselkedés szerinti rendezés
 - Mintapélda: Hibatűrés verifikációja

Részletes tervezés

- A részletes tervezés szerepe
 - Rendszerszintű részletes tervezés
 - Rendszerszintű algoritmusok: Komponensek együttműködése
 - Globális adatstruktúrák: Több modul által használtak
 - Komponens (modul) szintű részletes tervezés
 - Belső algoritmusok
 - Belső adatstruktúrák
- Módszerek jellegzetességei
 - Viselkedés leírása hangsúlyos: Állapotok, akciók, időbeliség, ...
 - Adatstruktúrák leírása: Absztrakt adatstruktúrák → implementáció
- Tervezési, modellezési nyelvek
 - Formális, félformális, strukturált módszerek
 - Implementáció-közeli „pseudo-kód” is lehet
- Verifikációs szempontok
 - Megfelelőség a szoftverkövetelmény-specifikációnak
 - Megfelelőség az előző fejlesztési lépések döntéseinek

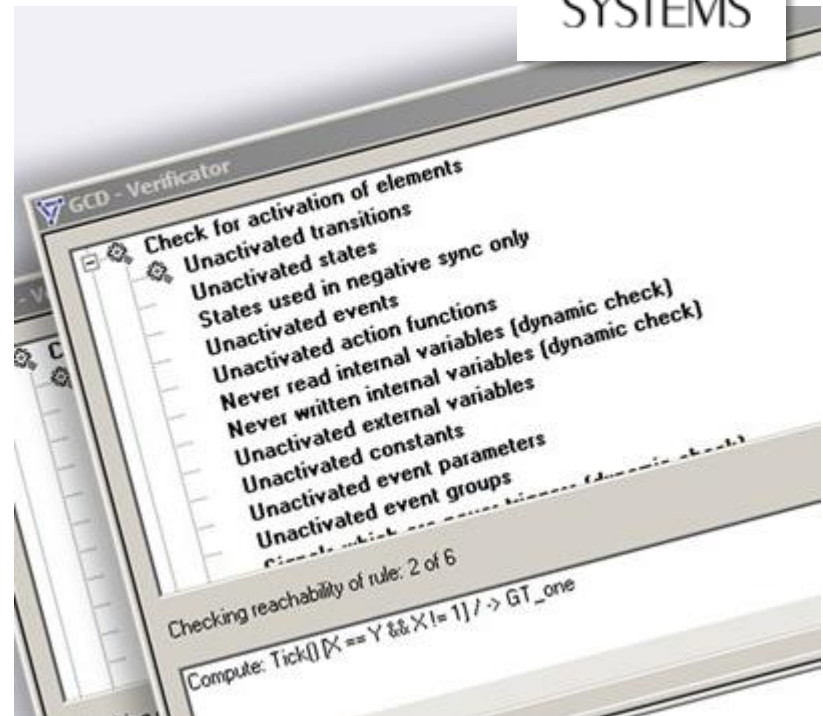
Ellenőrzési szempontok

- Követhetőség (követelmények fedettsége)
- Statikus ellenőrzés
 - Tipikus hibák ellenőrzése
 - Teljesség, ellentmondás-mentesség, megvalósíthatóság, tesztelhetőség
 - Vezérlési folyamat elemzése
 - Strukturáltság
 - Határérték elemzés
 - Adatáramlás elemzése
 - Értékadás és felhasználás kapcsolata
 - Futásidejű hibák ellenőrzése szimbolikus végrehajtással
- Dinamikus ellenőrzés
 - Szimuláció (modellek alapján)
 - Prototípus készítés és animáció
- Formális verifikáció

Eszköz példa: IAR visualSTATE Verificator

Checking for logical consistency:

- Are all elements used?
- Are all elements activated?
- Are there any ambiguities, such as conflicting transitions or dynamic ambiguous assignments?
- Does the system contain any dead ends?
- Is the signal queue neither too short nor too long?
- Is there any possibility of underflow, overflow, or similar arithmetic errors occurring?



<https://www.iar.com/iar-embedded-workbench/add-ons-and-integrations/visualstate/>

Eszköz példa: SCADE Suite Design Verifier

- Verification of safety properties expressed in SCADE Suite
- Automatic counter-example production in case of property failure
- Early detection of division-by-zero errors
- Easy and intuitive use of proof or bug-chasing modes



The image displays three windows from the SCADE Suite Design Verifier interface:

- Property definition:** A window titled "CruiseControl/VerifCruiseControl" showing a block diagram of a "CruiseControl" block with various input and output ports. A property definition is shown as a box with a left-pointing arrow and a right-pointing arrow, labeled "VerifCruiseSpeed".
- Design Verifier report:** A window titled "WPL Analyse Report" showing a "Sum Up" section with a table of tasks. The task "VerifCruiseControl/VerifCruiseSpeed" is listed as "Falsifiable".
- Generated counter example:** A window titled "VerifCruiseControl/VerifCruiseSpeed" showing a tree view of the counter example. The tree shows inputs (On, Off, Resume, Set, QuickAccel, QuickDecel, Accel, Brake, Speed) and outputs (VerifCruiseSpeed, Levels).

Arrows indicate the flow of information: from the property definition to the report, and from the report to the counter example.

Task	Result
VerifCruiseControl/VerifCruiseSpeed	Falsifiable

Node	Value
Node	CruiseControl:VerifCruiseControl
Output	VerifCruiseSpeed
Strategy	Default - Prove
Mapping	None
Group	None
Result	Falsifiable
Scenario	scenarios/VerifCruiseControl/VerifCruiseSpeed_01.sas [[Load Scenario]]

Eszköz példa: Verum ASD:Suite



ASD Verification Technology

Advantage: Precision

Verum's patented ASD technology is at the core of the model-driven design approach. It offers a new way to formally **verify the completeness and correctness of software designs** using our own mathematical methods.

Until now, most software architects have been unable to employ Formal Methods techniques to verify product design and function. Either they were considered too difficult to learn or too expensive to use and maintain for common code bases.

With the introduction of ASD:Suite software design (CAD) platform, the focus moves from defect removal to defect prevention. Its formal verification engine gives software architects the power to:

- Verify that specifications are complete.
- Model the **behaviours** of all system components and interfaces.
- Produce defect-free code after verifying designs are complete and correct.
- Guarantee **equivalence between specifications, designs, formal models and behaviour** of generated code.
- Ensure mathematical integrity throughout the design process.

<http://news.verum.com/Technology/analytical-verification.aspx>

Eszköz példa: Verum ASD:Suite



ASD Verification Technology

Advantage: Precisi

Verum
for

U
p
m
W
d

- No deadlocks
- No livelocks
- Guard completeness
- Correctly handled race conditions
- Invariants
- Absence of illegal behaviour (library components)
- No state variable out-of-range errors
- Determinism
- Queue does not get full
- Implementation vs. specification

ht

Tartalomjegyzék

- **Áttekintés**
 - Milyen szerepe van a részletes terveknek?
 - Milyen ellenőrzési módszerek vannak?
- **Statikus ellenőrzés felülvizsgálattal**
- **Formális verifikáció modellellenőrzéssel**
 - Módszerek áttekintése
 - Modellellenőrzés mérnöki modellek alapján
- **Formális verifikáció ekvivalencia ellenőrzéssel**
 - Trace ekvivalencia
 - Megfigyelési ekvivalencia (gyenge biszimuláció)
 - Lehetséges viselkedés szerinti rendezés
 - Mintapélda: Hibatűrés verifikációja

A felülvizsgálat (review) módszere

- Fázisok (meghatározott folyamat)
 1. Tervezés: Dokumentumok, résztvevők, kritériumok
 2. Kick-off: Folyamat ismertetése, tervek szétosztása
 3. Előkészítés: Tervek átvizsgálása, problémák feltárása
 4. Megbeszélés: Problémák egyeztetése, rögzítése, javaslatok
 5. Átdolgozás: Javítások, módosítások
 6. Követés: Javítások ellenőrzése, metrikák képzése, kilépés
- Szerepek
 - Moderátor, szerző (tervező), jegyzőkönyvvezető, átvizsgálók
- Típusok (a felülvizsgálat célja szerint)
 - **Átvizsgálás** (walkthrough): A szerző által végzett bemutató, célja információgyűjtés és konszenzus az átvizsgálókkal
 - **Technikai felülvizsgálat** (technical review): A technikai megközelítés tekintetében törekszik közös álláspontra jutni
 - **Vizsgálat** (inspection): Dokumentált eljárás a hibák és kiindulási dokumentumokhoz képest történő eltérések felderítése érdekében

Tartalomjegyzék

- **Áttekintés**
 - Milyen szerepe van a részletes terveknek?
 - Milyen ellenőrzési módszerek vannak?
- **Statikus ellenőrzés felülvizsgálattal**
- **Formális verifikáció modellellenőrzéssel**
 - Módszerek áttekintése
 - Modellellenőrzés mérnöki modellek alapján
- **Formális verifikáció ekvivalencia ellenőrzéssel**
 - Trace ekvivalencia
 - Megfigyelési ekvivalencia (gyenge biszimuláció)
 - Lehetséges viselkedés szerinti rendezés
 - Mintapélda: Hibatűrés verifikációja

Formális verifikáció

- Matematikai eszközök használata
 - Formális nyelv: Formális szintaxis és szemantika
 - Viselkedés leírás (terv, implementáció)
 - Tulajdonság leírás (követelmények, specifikáció)
 - Algoritmus (itt: verifikáció) a formális modellen
- A verifikáció módja
 - „Önmagában való” vizsgálat
 - Pl. tervek, modellek ellentmondás-mentessége
 - „Elvárt tulajdonságok” ellenőrzése
 - Pl. szoftverkövetelmény-specifikáció betartása
 - „Változások” vizsgálata
 - Pl. tervezői döntések hatása: funkció, tulajdonság megtartása

Tervek és modellek szerepe a formális verifikációban

• Viselkedés leírás lehetőségei

– Alapszintű:

- KS, LTS, KTS, automaták, Büchi automaták

– Magasabb szintű:

- Vezérlés orientált: Hierarchikus automata, Petri-háló
- Adatfeldolgozás orientált: Adatfolyam háló
- Kommunikáció orientált: Processz algebrák

– Mérnöki modellek:

- UML diagramok formális szemantikával

• Tulajdonság leírás lehetőségei

– Alapszintű:

- Elsőrendű logika, temporális logika, referencia automata

– Magasabb szintű:

- Message Sequence Charts, Live Sequence Charts, ...

Példa: UML állapottérképek modellellenőrzése

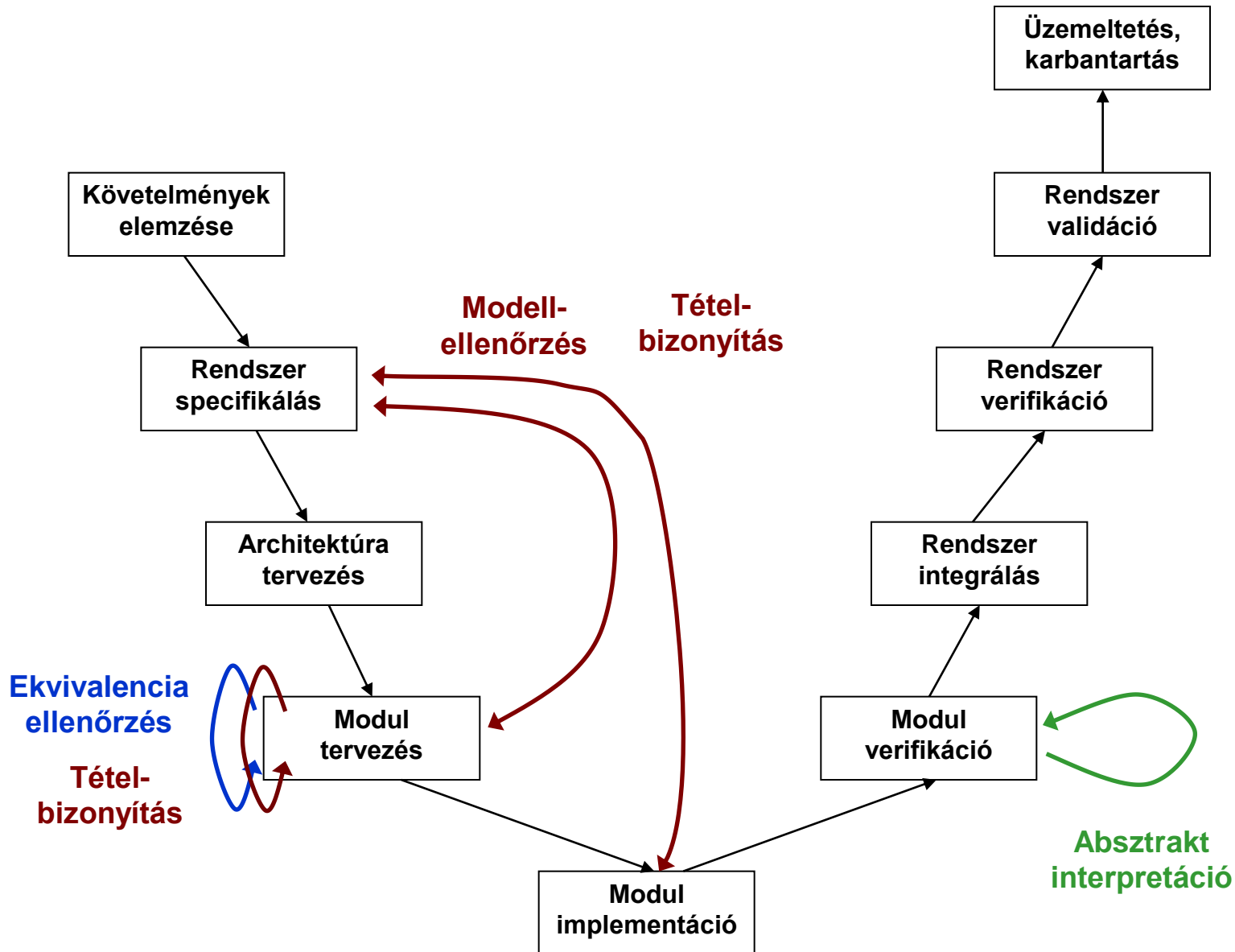
- Viselkedés-megtartó modelltranszformáció:
Kiterjesztett UML → **Időzített automata (UPPAAL)**

- Aktív objektumok (osztályok) leképezése automatákká
- Eseménykezelés leképezése
 - Külső környezet modellezése: Eseményeket „generál”
 - Eseménysor, esemény ütemező, RTC lépések
- Adatkezelés leképezése
 - Egyszerű adattípusok, korlátozott adattartományok
- Konkurens és hierarchikus állapotmodell leképezése
 - Állapotkonfigurációk
- Időkezelés (kiterjesztés)
 - Megfigyelő (clock observer) idő eseményeket generál

A legelterjedtebb formális verifikációs technikák

Modell / technika	Viselkedés modell (alapszintű)	Tulajdonság leírás (alapszintű)
Modellellenőrzés	Kripke struktúra	Temporális logika
Ekvivalencia ellenőrzés	Címkézett tranzíciós rendszer (LTS), automata	Referencia viselkedés: LTS, automata
Tételbizonyítás	Dedukciós rendszer (axiómák, levezetési szabályok)	Bizonyítandó tétel
Statikus analízis: absztrakt interpretáció	Kripke tranzíciós rendszer (programból absztrakcióval)	Assertion (elsőrendű logikai)

A formális verifikációs technikák szerepe



Modellellenőrzés

- Alacsony szintű formalizmusok (KS, LTS, KTS, TA)
- Magasabb szintű formalizmusok

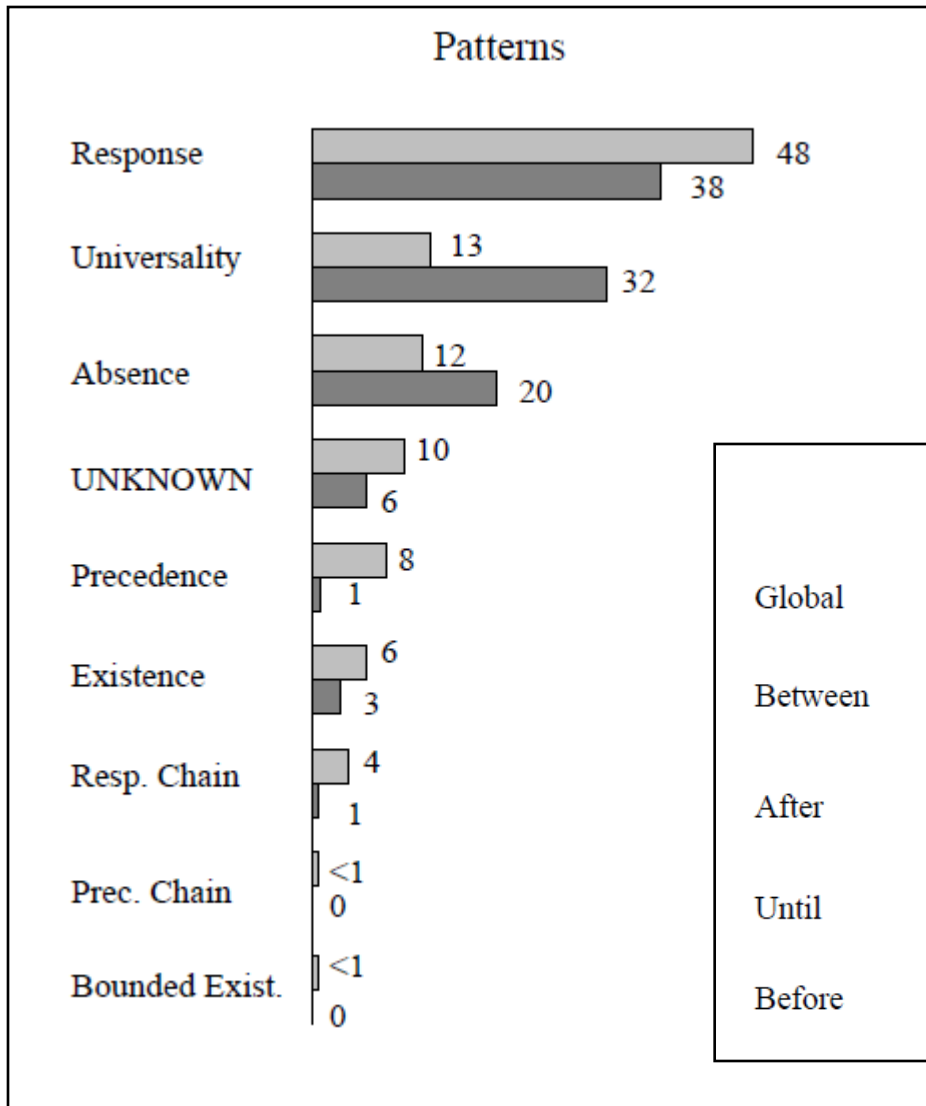
Temporális logikák:
PLTL, CTL, CTL*

Viselkedés modellje

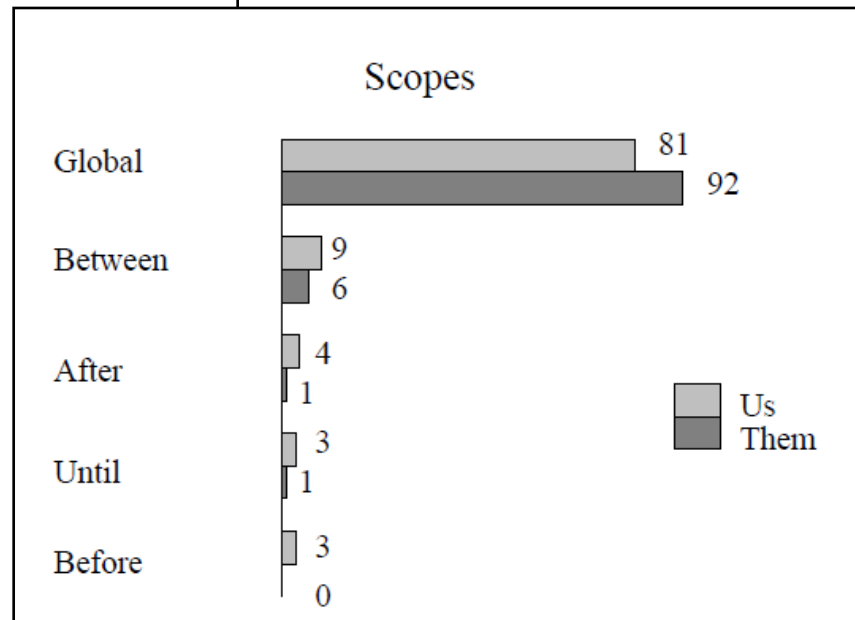
Tulajdonság megadása



Milyen jellegű követelmények ellenőrizhetők?



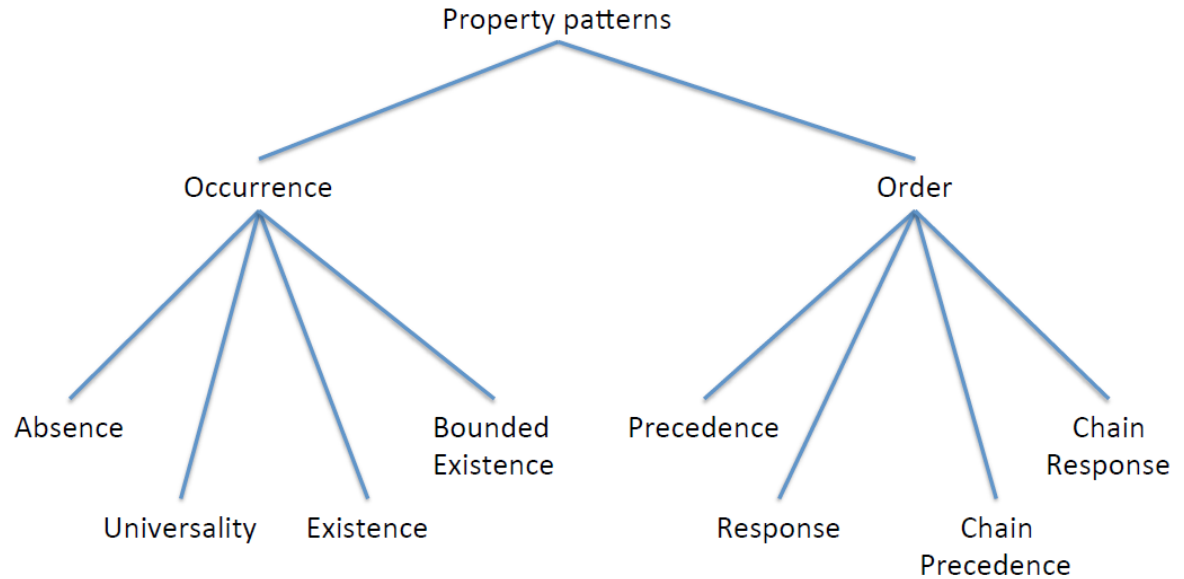
A (biztonsági) követelmények jelentős hányada meghatározott mintákra illeszkedik



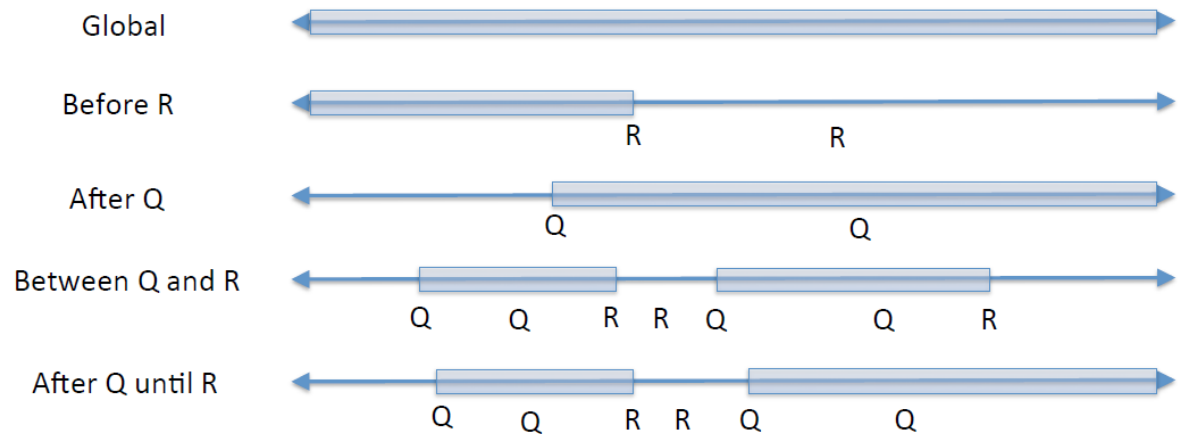
<http://patterns.projects.cis.ksu.edu/documentation/patterns.shtml>

Követelmény minták

Minták:
sorrendi
előírások



Hatókörök:
további
eseményekhez
képest



A minták jelentése

A sorrendi előírások az adott hatókörben:

- **Absence:** A hivatkozott állapot/esemény nem fordul elő.
- **Universality:** A hivatkozott állapot/esemény folyamatosan jelen van.
- **Existence:** A hivatkozott állapot/esemény biztosan előfordul.
- **Bounded existence:** A hivatkozott állapot/esemény biztosan előfordul „k” alkalommal (létezik „legalább k” és „legfeljebb k” változat is.)
- **Precedence:** Az egyik állapot/esemény mindenképpen meg kell, hogy előzze a másik állapotot/eseményt.
- **Response:** Az egyik állapotot/eseményt mindenképpen kell, hogy kövesse egy másik meghatározott állapot/esemény.
- **Chain precedence:** A Precedence minta általánosítása állapot / esemény sorozatokra.
- **Chain response:** A Response minta általánosítása állapot / esemény sorozatokra.

Példák a minták megjelenésére

- **Response minta Global hatókörben:**
 - A végrehajtás során bármikor ha **Request** kérés következik be, akkor vagy egy **Reply** vagy egy **Reject** válasznak kell következnie.
- **Precedence minta After hatókörben:**
 - A **NormalMode** állapot bekövetkezése után a **ResourceGranted** állapot csak akkor következhet be, ha ezt megelőzi **ResourceRequest** állapot.

Követelmény minták formalizálása

Universality within scope	Property in LTL	Meaning of LTL operators
Event P occurs in each step of the execution globally .	$G P$	Globally P
Event P occurs in each step of the execution before event Q .	$F Q \rightarrow (P U Q)$	Eventually Q implies (P Until Q)
Event P occurs in each step of the execution after event Q .	$G (Q \rightarrow G P)$	Globally (Q implies Globally P)
Event P occurs in each step of the execution between events Q and R .	$G ((Q \wedge \neg R \wedge F R) \rightarrow (P U R))$	Globally ((Q and not R and Eventually R) implies (P Until R))

Existence within scope	Property in LTL
Event P occurs in the execution globally .	$F P$
Event P occurs in the execution before event Q .	$\neg Q W (P \wedge \neg Q)$
Event P occurs in the execution after event Q .	$G (\neg Q) \vee F (Q \wedge F P)$
Event P occurs in the execution between events Q and R .	$G (((Q \wedge \neg R) \wedge (F R)) \rightarrow (\neg R W (P \wedge \neg R)))$

Modellellenőrzési technikák

- Algoritmusok (Id. Formális módszerek)
 - Teljes (kimerítő) modellellenőrzés
 - Tabló alapú (kibontás a modell alapján)
 - Szemantikán alapuló iterációs algoritmus (állapottér bejárás)
 - Hatékony állapot tér reprezentáció (ROBDD)
 - Korlátos modellellenőrzés
 - Az állapot térben adott mélységig ellenőriz
(kezdőállapottól indulva adott végrehajtási útvonal hosszig)
 - Hatékony SAT technikákra visszavezethető
- Előnyök és hátrányok
 - ☺ Teljesen automatikus ellenőrzés
 - ☺ Ellenpélda generálás (hibakereséshez)
 - ☹ Vezérlés-orientált ellenőrzés
 - ☹ Állapottér robbanás (részben kezelhető)

Tartalomjegyzék

- **Áttekintés**
 - Milyen szerepe van a részletes terveknek?
 - Milyen ellenőrzési módszerek vannak?
- **Statikus ellenőrzés felülvizsgálattal**
- **Formális verifikáció modellellenőrzéssel**
 - Módszerek áttekintése
 - Modellellenőrzés mérnöki modellek alapján
- **Formális verifikáció ekvivalencia ellenőrzéssel**
 - Trace ekvivalencia
 - Megfigyelési ekvivalencia (gyenge biszimuláció)
 - Lehetséges viselkedés szerinti rendezés
 - Mintapélda: Hibatűrés verifikációja

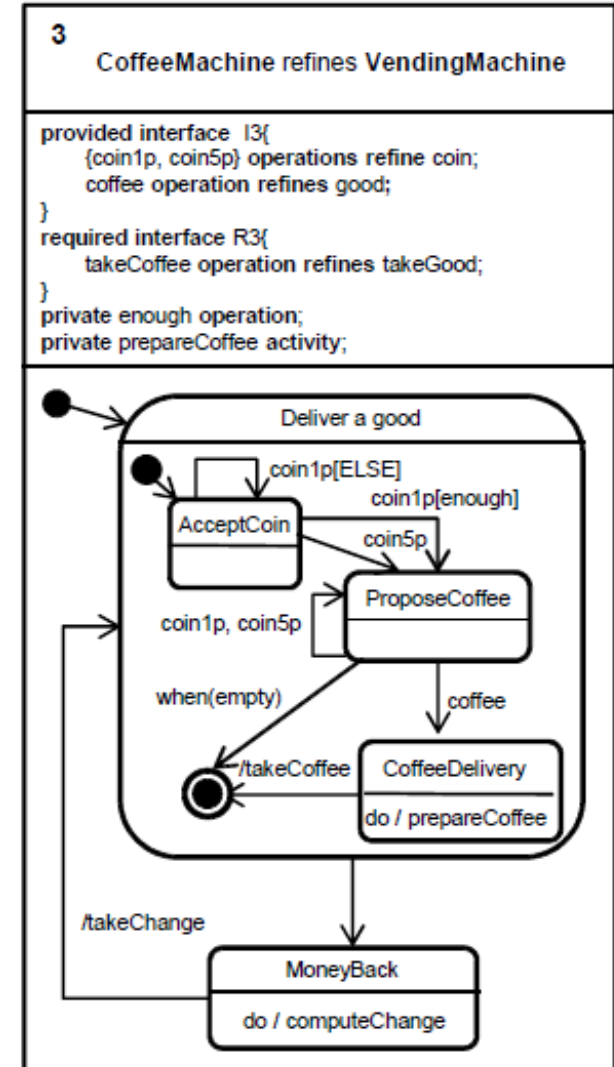
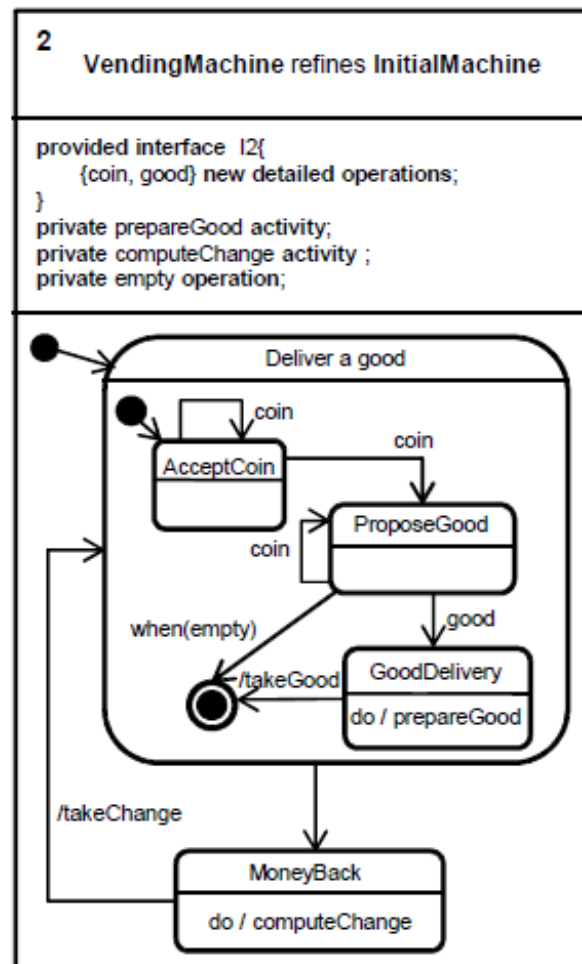
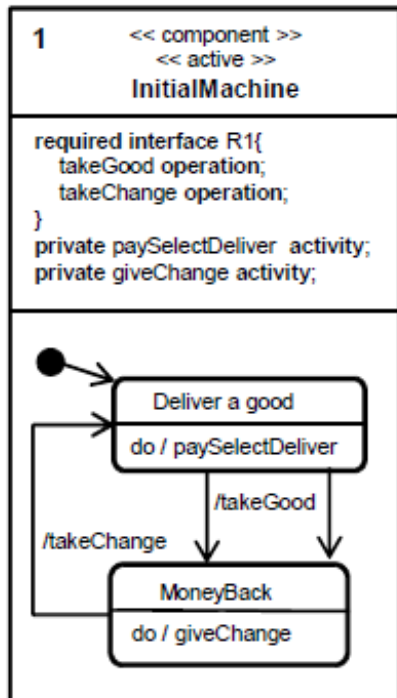
Használat: Tervezői döntések ellenőrzése

- **Megfelelőség (ekvivalencia) modellek között**
 - Referencia modell \leftrightarrow Vizsgált (módosított) modell
 - Specifikáció (absztrakt) \leftrightarrow Megvalósítás (konkrét)
 - Elvárt protokoll \leftrightarrow Nyújtott protokoll
 - Ideális rendszer \leftrightarrow Hibatűrő rendszer hibák mellett
- **Finomítás (rendezés) modellek között:**
 - Referencia viselkedés megtartása, meghatározott bővítésekkel
 - Nemdeterminizmus csökkentése

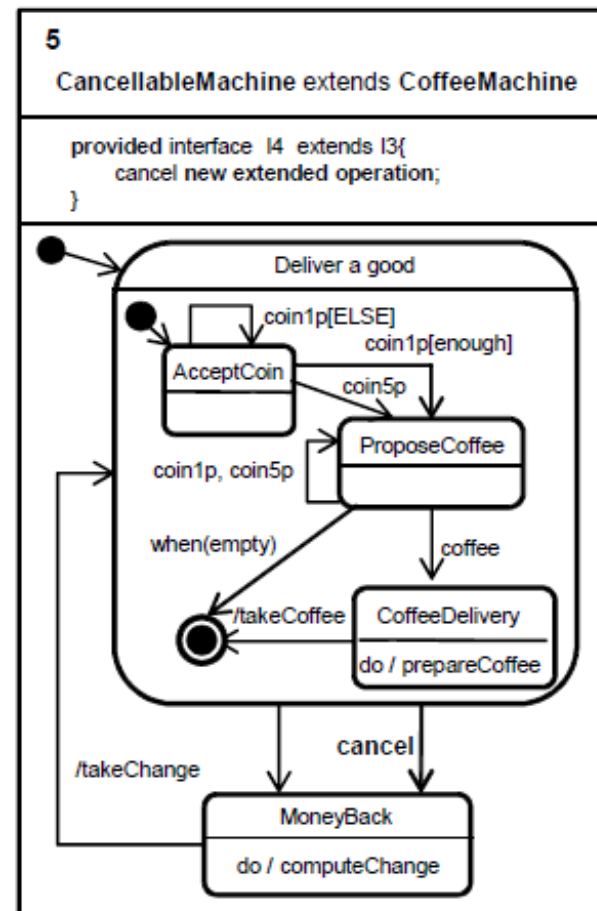
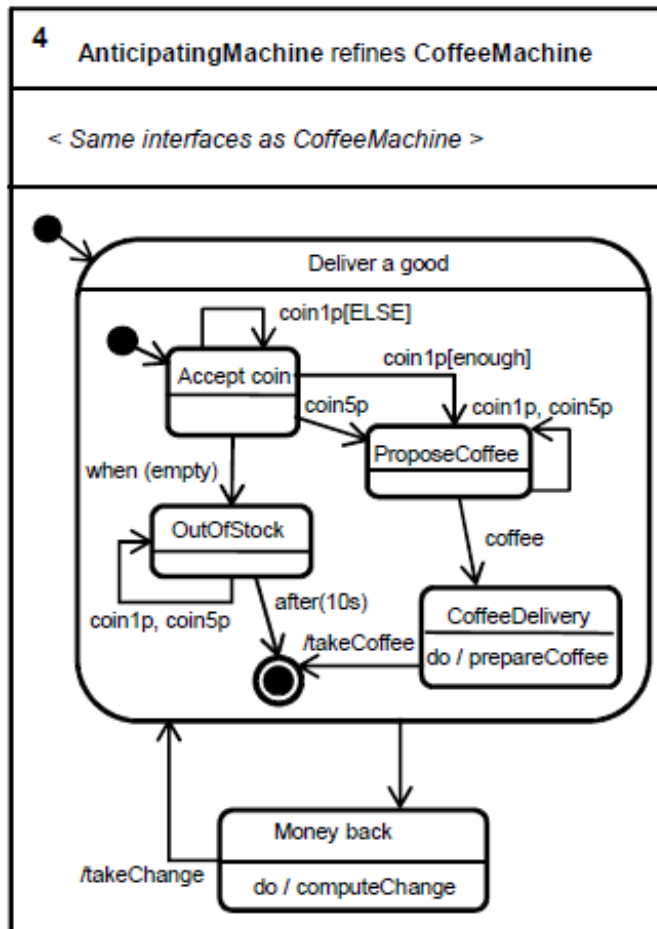
Használt matematikai relációk modellek között:

- **Ekvivalencia:** Reflexív, tranzitív, szimmetrikus
- **Rendezés:** Reflexív, tranzitív, antiszimmetrikus

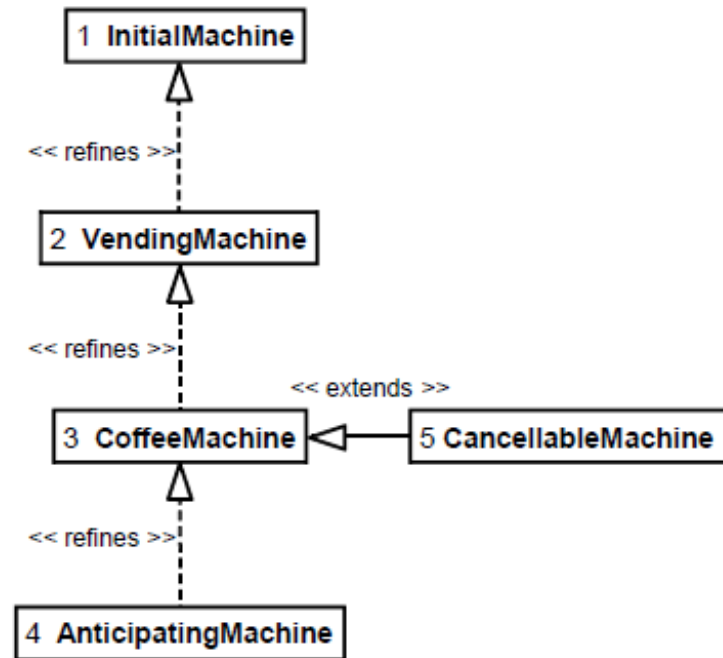
Példa: Állapottérkép modellek finomítása



Példa: Állapottérkép modellek finomítása (folytatás)



Példa: Mit szeretnénk? Relációkat ellenőrizni az állapottérkép modelleken!

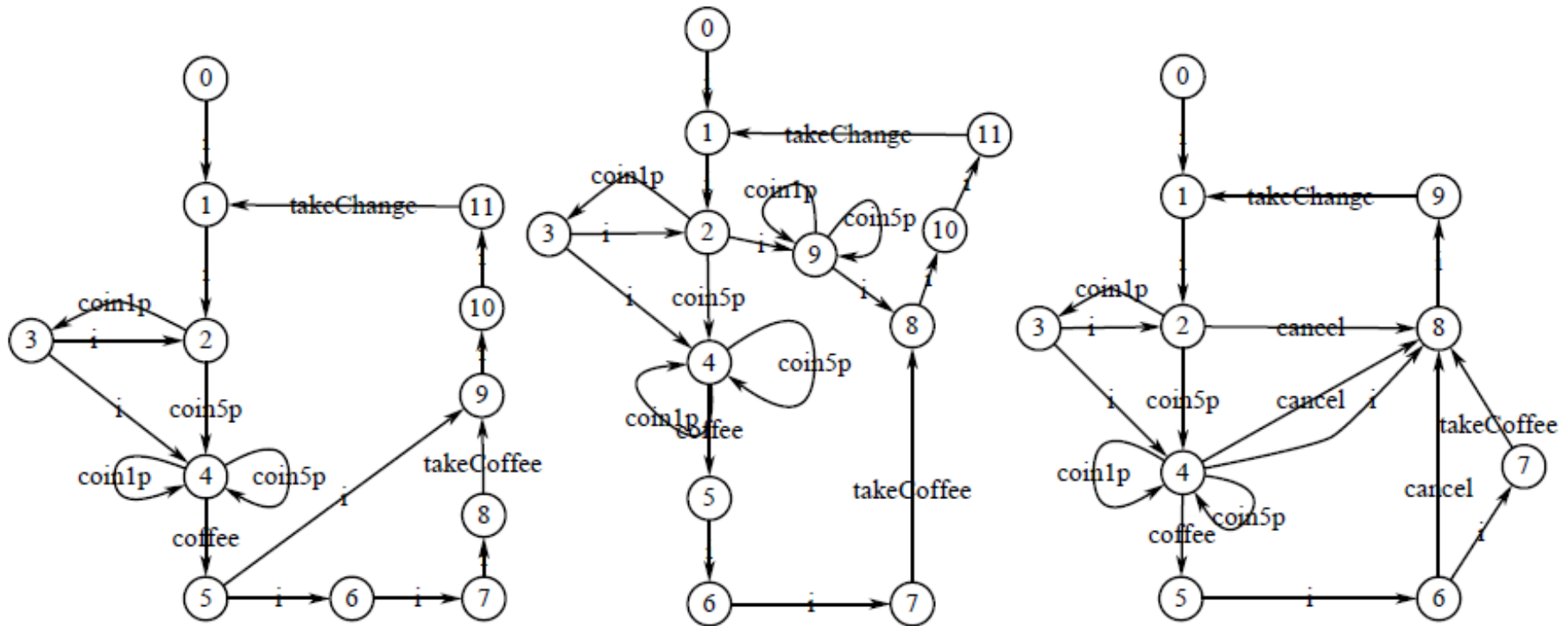


Példa: Mit várunk egy finomítás relációtól?

- Reflexív és tranzitív reláció
- Élő tulajdonság megtartása: A finomított modell tudja nyújtani azt az viselkedést, amit az eredeti modellnek is tudnia kellett
 - Méltányosság (fairness) feltételezés: Az élő tulajdonság megtartása méltányos viselkedés mellett történik (azaz választás esetén minden viselkedésnek van esélye)
- Komponálhatóság:
 - Egymás utáni finomítások finomítást eredményeznek
 - Finomítás és kiterjesztés együtt kiterjesztést eredményez
- ...

Pontos definíció szükséges!

Példa: A relációk értelmezhetők az állapottérkép modellekből származó LTS-eken



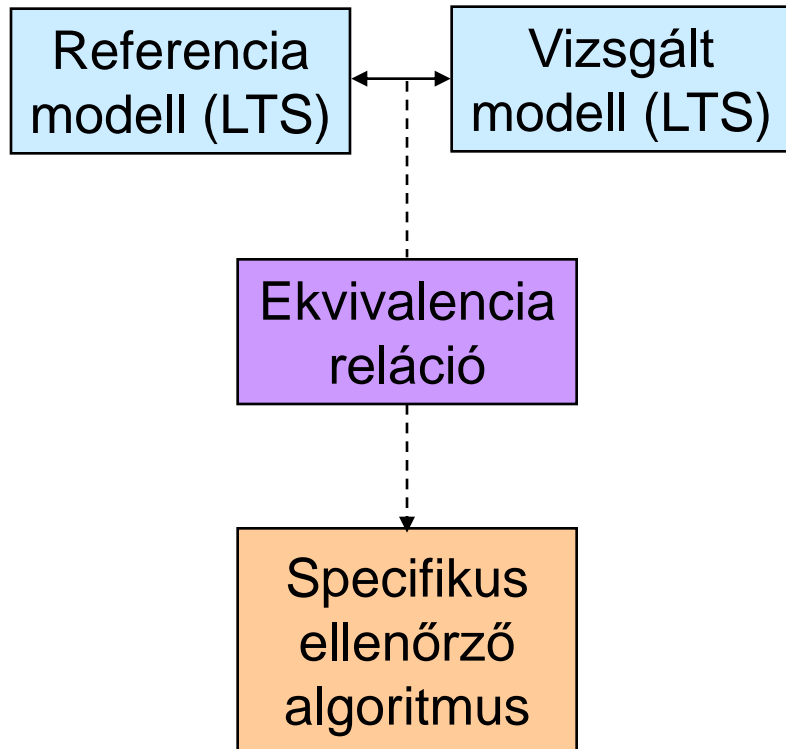
a. LTS₃: CoffeeMachine

b. LTS₄: AnticipatingMachine

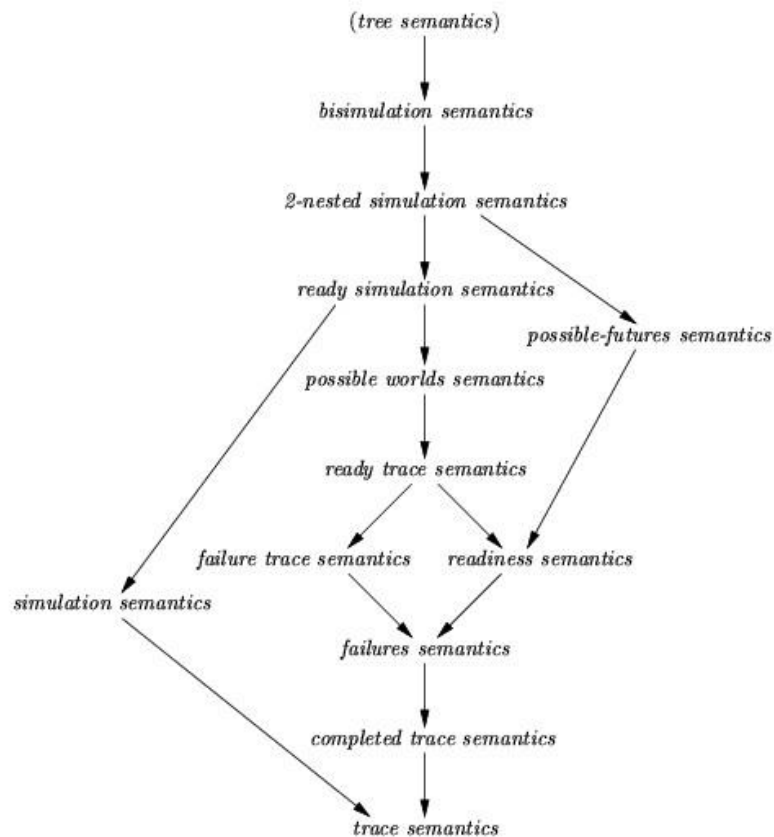
c. LTS₅: CancellableMachine

Az ekvivalencia ellenőrzés formalizálása

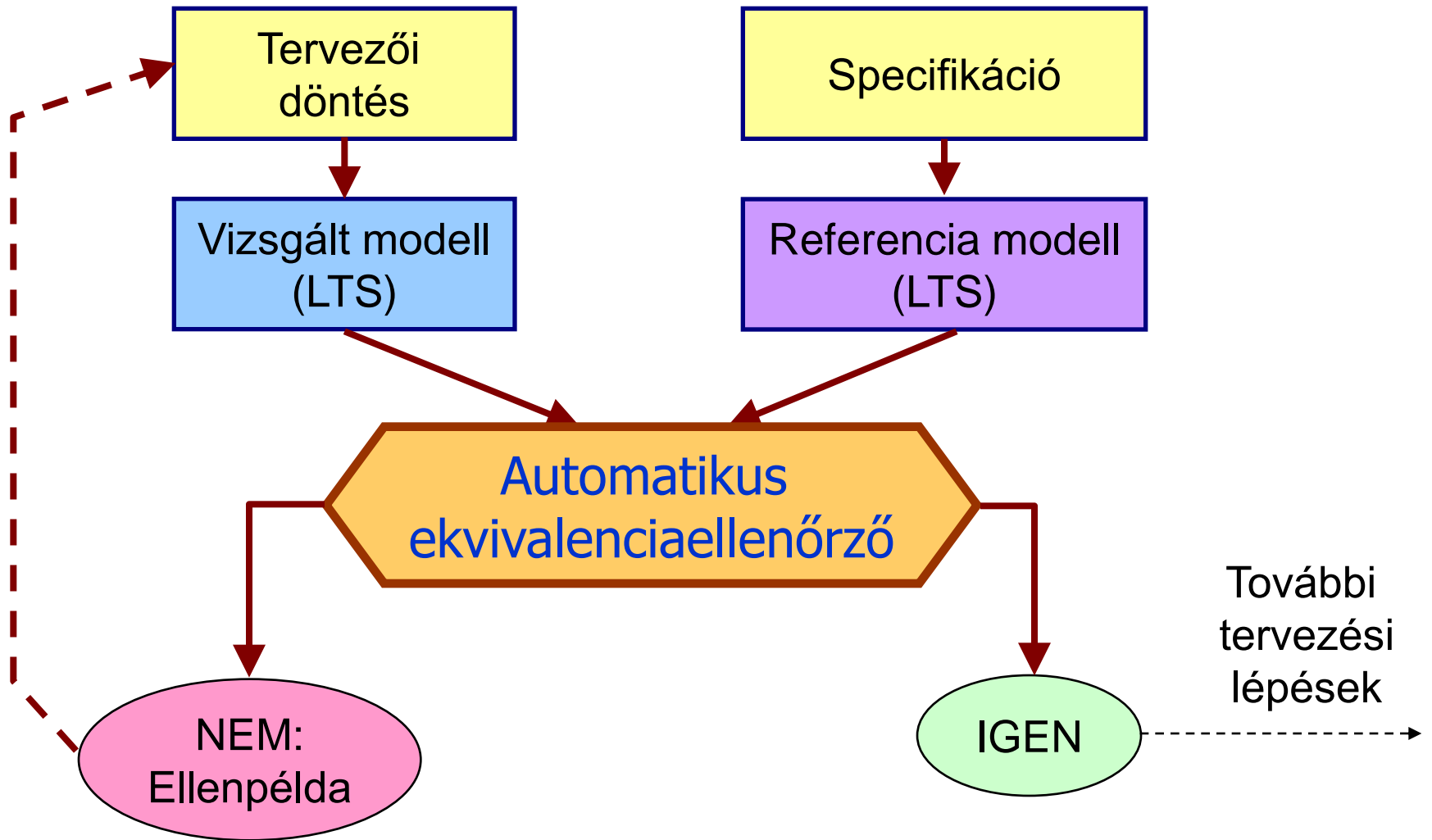
Ekvivalencia ellenőrzés:



Ekvivalencia relációk:



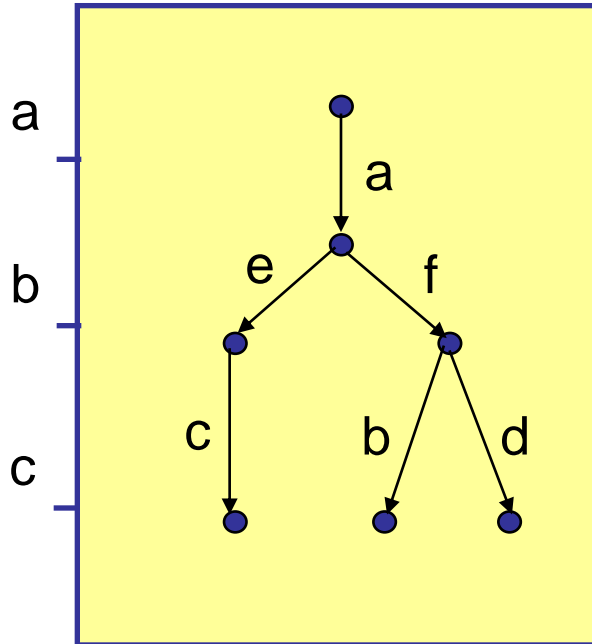
Automatikus ekvivalencia ellenőrzés



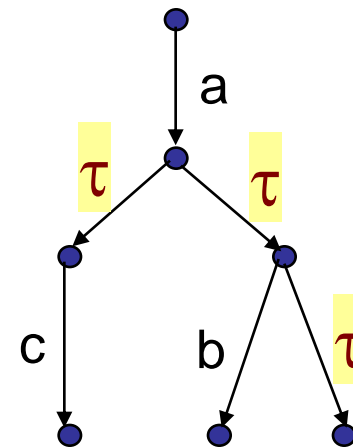
A komponensek „fekete doboz” modellje

- **Megfigyelhető akciók**
 - A vizsgált komponens (modul) **interfészen megjelenő**, a környezet számára érdekes („releváns”) viselkedés
 - Metódus hívása, metódushívás fogadása
 - Üzenet küldése, üzenet fogadása
- **Nem megfigyelhető akciók (i , τ)**
 - Az interfészen nem megjelenő, vagy a környezet számára nem érdekes („don't care”) viselkedés
 - Belső működés (metódusok)
 - Figyelman kívül hagyható hívások, üzenetek
- **Nemdeterminizmus**
 - A környezet szempontjából nem megfigyelhető belső működés is okozhatja (ezt eltakarja a fekete doboz modell)

Megfigyelhető viselkedés



Komponens
belső viselkedés:
a,b,c,d,e,f akciók



Megfigyelhető
viselkedés:
a,b,c akciók

Trace ekvivalencia: Jelölések

- Minta: Automaták ekvivalenciája

$$A_1 = A_2 \text{ ha } L(A_1) = L(A_2)$$

- LTS-ek esetén analógia:

- Minden állapot „elfogadó állapot”
- Nyelv: Minden akciószekvencia (trace)

- Jelölések:

$\alpha = a_1 a_2 a_3 a_4 \dots a_n \in Act^*$ véges akciószekvencia (ε az üres)

$s \xrightarrow{\alpha} s'$ ha $\exists s_0 s_1 \dots s_n$ állapotsorozat ahol $s_0 = s$, $s_n = s'$, $s_i \xrightarrow{a_{i+1}} s_{i+1}$

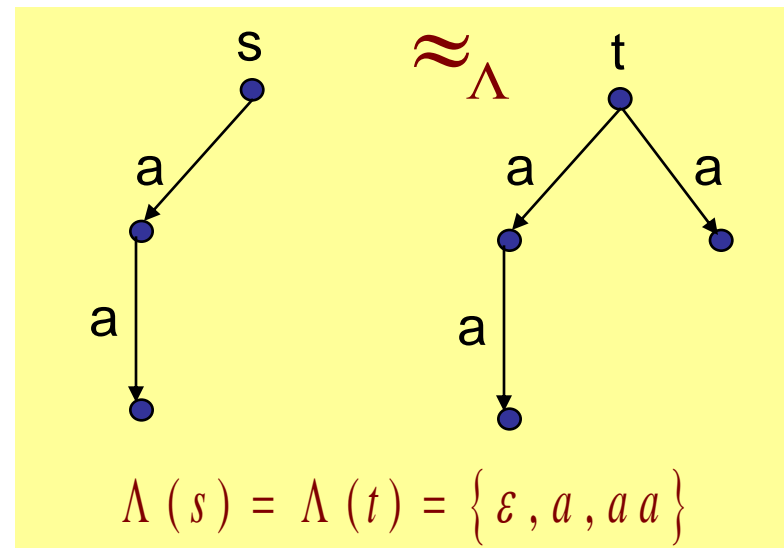
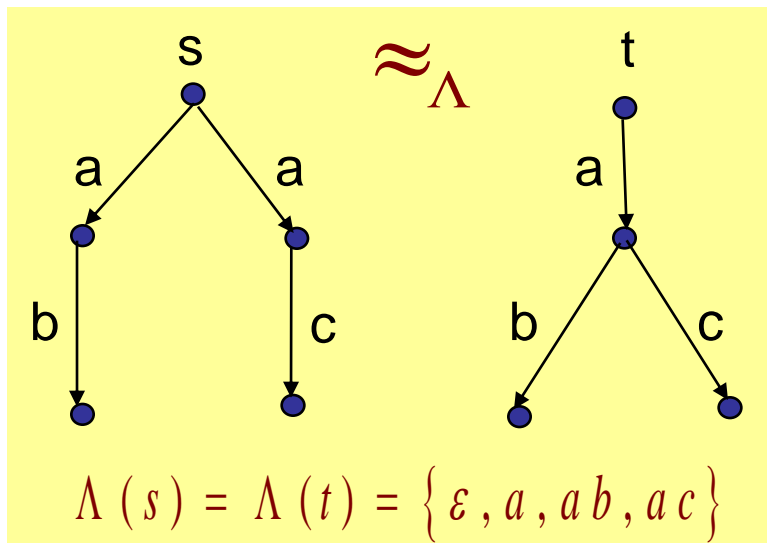
$\Lambda(s)$ legyen s trace-einek halmaza: $\Lambda(s) = \left\{ \alpha \mid \exists s' : s \xrightarrow{\alpha} s' \right\}$

Trace ekvivalencia: Definíció és példák

- Legyen T_1 és T_2 két LTS, s_1 és s_2 kezdőállapottal
- Definíció:

$$T_1 \approx_{\Lambda} T_2 \text{ a.c.s.a. } \Lambda(s_1) = \Lambda(s_2)$$

- Példák:



- Problémák:

- Ekvivalens LTS-ek eltérő viselkedése, pl. megállás szempontjából
- Azonos trace-ek különböző állapotokon keresztül

Megfigyelési ekvivalencia (gyenge biszimuláció)

- Jellegzetességek

- Lényege: Két megfigyelhető viselkedés ekvivalens, ha egymást szimulálni tudják:
 - Azonosak a megfigyelhető akciószekvenciák
 - Ekvivalens állapotokon keresztül figyelhető meg
- Nem érzékeny a hatás nélküli belső átmenetekre (azok megjelenésére, számára)

- Jelölések:

$\alpha \in Act^*$ véges akciószekvencia (ε az üres)

$\hat{\alpha} \in (Act - \tau)^*$ megfigyelhető akciószekvencia τ törlésével

ha $\alpha = \tau$ akkor $\hat{\alpha} = \varepsilon$

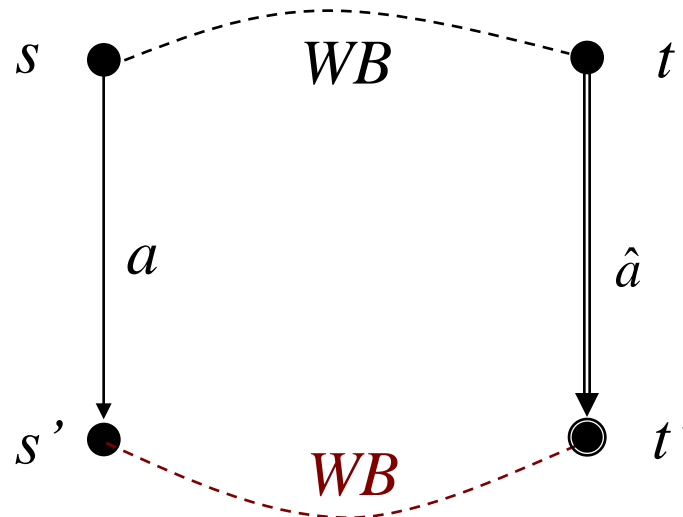
$s \stackrel{\beta}{\Rightarrow} s'$ ha $\exists \alpha: s \stackrel{\alpha}{\rightarrow} s'$ és $\beta = \hat{\alpha}$

Gyenge biszimuláció: Definíció

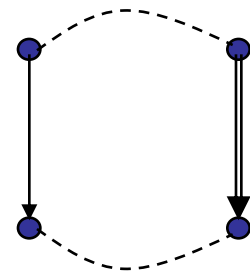
- Definíció:

$WB \subseteq S \times S$ gyenge biszimuláció, ha minden $(s, t) \in WB$ és bármely $a \in Act$, $s', t' \in S$ esetén fennáll:

- ha $s \xrightarrow{a} s'$ akkor $\exists t' : t \xRightarrow{\hat{a}} t'$ és $(s', t') \in WB$
- ha $t \xrightarrow{a} t'$ akkor $\exists s' : s \xRightarrow{\hat{a}} s'$ és $(s', t') \in WB$



Extrém eset:



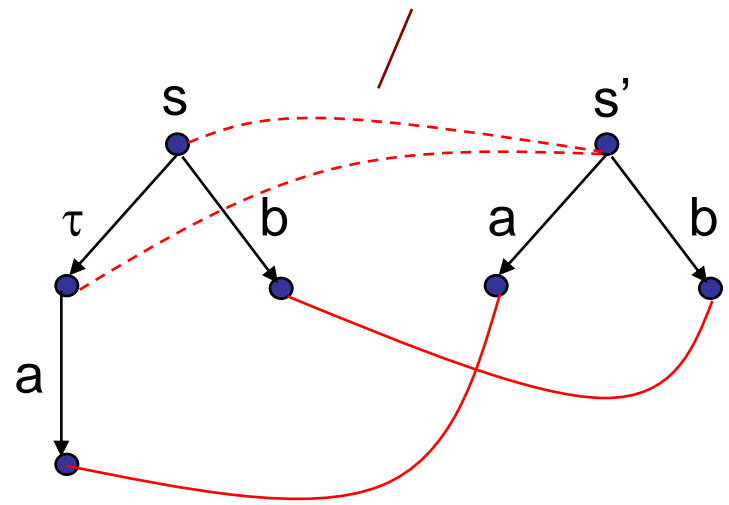
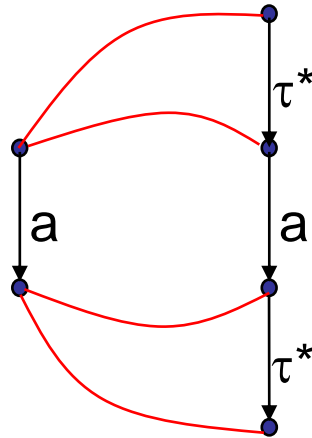
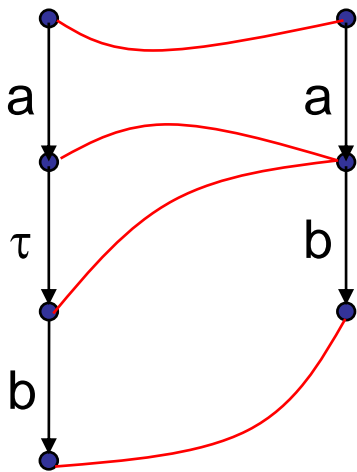
Megfigyelési ekvivalencia: Definíció és példák

- Megfigyelési ekvivalencia

= gyenge biszimuláció ekvivalencia (observation equivalence)

$$T_1 \approx T_2 \text{ a.c.s.a. } s_1 \approx s_2 \text{ azaz } \exists W B : (s_1, s_2) \in W B$$

- Példák



Ekvivalencia relációk számítási módszere

Partíció finomítás:

1. Kezdetben minden állapotpár eleme a relációnak
Egy partíciót (ekvivalencia osztályt) képeznek
2. Minden állapotpárra:
Ha az egyikből indulva van olyan átmenet, amit a másik nem tud a definíció szerint szimulálni, akkor
 - Az adott állapotpár kizárása a relációból (nem ekvivalensek)
 - Következmények végigvezetése a bejövő átmenetek végén lévő állapotokra
 - Nem ekvivalensek, ha nem ekvivalens állapotokba kerülnek
3. Ha már nincs változás:
Végleges ekvivalencia osztályok adódtak
Ha a kezdőállapotok azonos ekvivalencia osztályban vannak, akkor az LTS-ek ekvivalensek

Egy rendezési reláció: Lehetséges viselkedés szerint

- Jelölések:

$\beta \in (Act - \tau)^*$ megfigyelhető akciószekvencia τ törlésével

$s \xRightarrow{\beta} s'$ ha $\exists \alpha \in Act^*: s \xrightarrow{\alpha} s'$ és $\beta = \hat{\alpha}$

$\Delta(s)$ az s -ből induló megfigyelhető akciószekvenciák halmaza:

$$\Delta(s) = \left\{ \beta \mid \exists s' : s \xRightarrow{\beta} s' \right\}$$

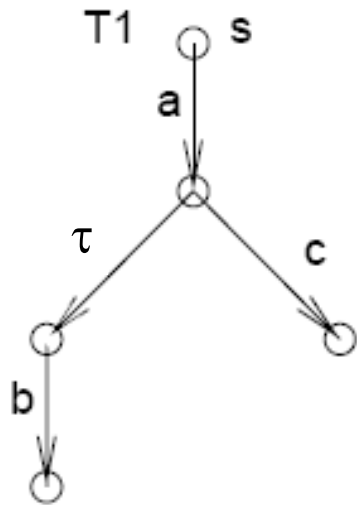
- Lehetséges viselkedés szerinti rendezés
 T_1 és T_2 LTS-ek esetén:

$$T_1 \leq_{\Delta} T_2 \text{ a.cs.a. } \Delta(s_1) \subseteq \Delta(s_2)$$

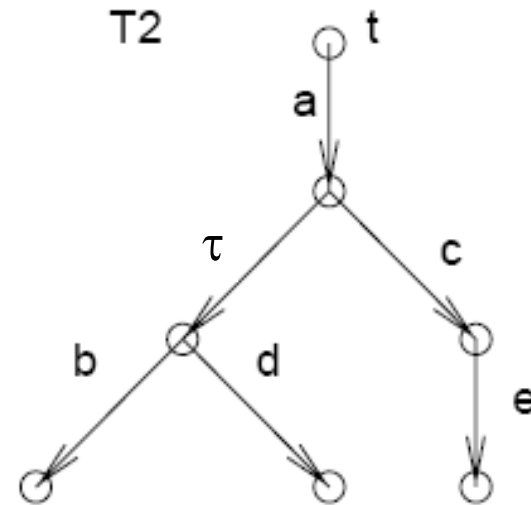
itt T_2 -ben több a megfigyelhető akciószekvencia

Lehetséges viselkedés szerinti rendezés: Példa

- Eredeti viselkedés bővítése:



$$\Delta(s) = \{a, ab, ac\}$$



$$\Delta(t) = \{a, ab, ac, ad, ace\}$$

Lehetséges viselkedés szerinti rendezés és a tesztelés

- **Kapcsolat a teszteléssel:** $T_1 \leq_{\Delta} T_2$ esetén:
 - Minden teszt, ami T_1 esetén sikeres lehet, T_2 esetén is sikeres lehet, de belső akciók miatt sikertelen is lehet (nemdeterminizmus)
 - Másképp: T_1 lehetséges sikeres tesztjei T_2 lehetséges sikeres tesztjei között vannak
- **Jellegzetességek:**
 - Olyan finomítást definiál, amelynek során nem „veszik el” lehetséges megfigyelhető viselkedés

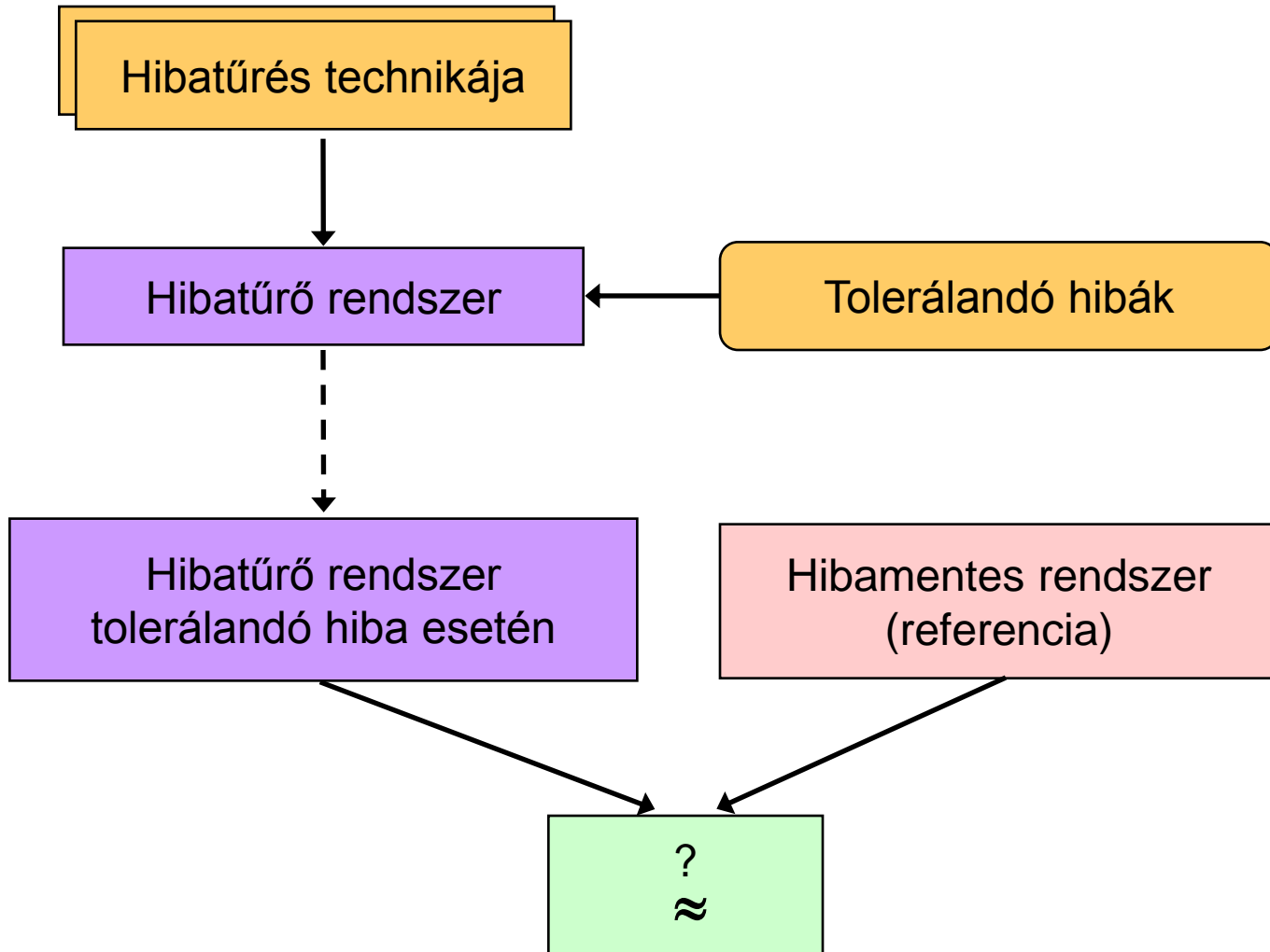
Egy másik rendezési reláció: Szükséges viselkedés

- Olyan finomítás, ami a mindig sikeres tesztekre („kötelezően” megfigyelhető viselkedésre) ad megkötést

Tartalomjegyzék

- **Áttekintés**
 - Milyen szerepe van a részletes terveknek?
 - Milyen ellenőrzési módszerek vannak?
- **Statikus ellenőrzés felülvizsgálattal**
- **Formális verifikáció modellellenőrzéssel**
 - Módszerek áttekintése
 - Modellellenőrzés mérnöki modellek alapján
- **Formális verifikáció ekvivalencia ellenőrzéssel**
 - Trace ekvivalencia
 - Megfigyelési ekvivalencia (gyenge biszimuláció)
 - Lehetséges viselkedés szerinti rendezés
 - **Mintapélda: Hibatűrés verifikációja**

Mintapélda: Hibatűrés verifikációja



Miről volt szó?

- **Áttekintés**
 - Milyen szerepe van a részletes terveknek?
 - Milyen ellenőrzési módszerek vannak?
- **Statikus ellenőrzés felülvizsgálattal**
- **Formális verifikáció modellellenőrzéssel**
 - Módszerek áttekintése
 - Modellellenőrzés mérnöki modellek alapján
- **Formális verifikáció ekvivalencia ellenőrzéssel**
 - Trace ekvivalencia
 - Megfigyelési ekvivalencia (gyenge biszimuláció)
 - Lehetséges viselkedés szerinti rendezés
 - Mintapélda: Hibatűrés verifikációja