

Részletes szoftver tervek ellenőrzése

Majzik István

Budapesti Műszaki és Gazdaságtudományi Egyetem

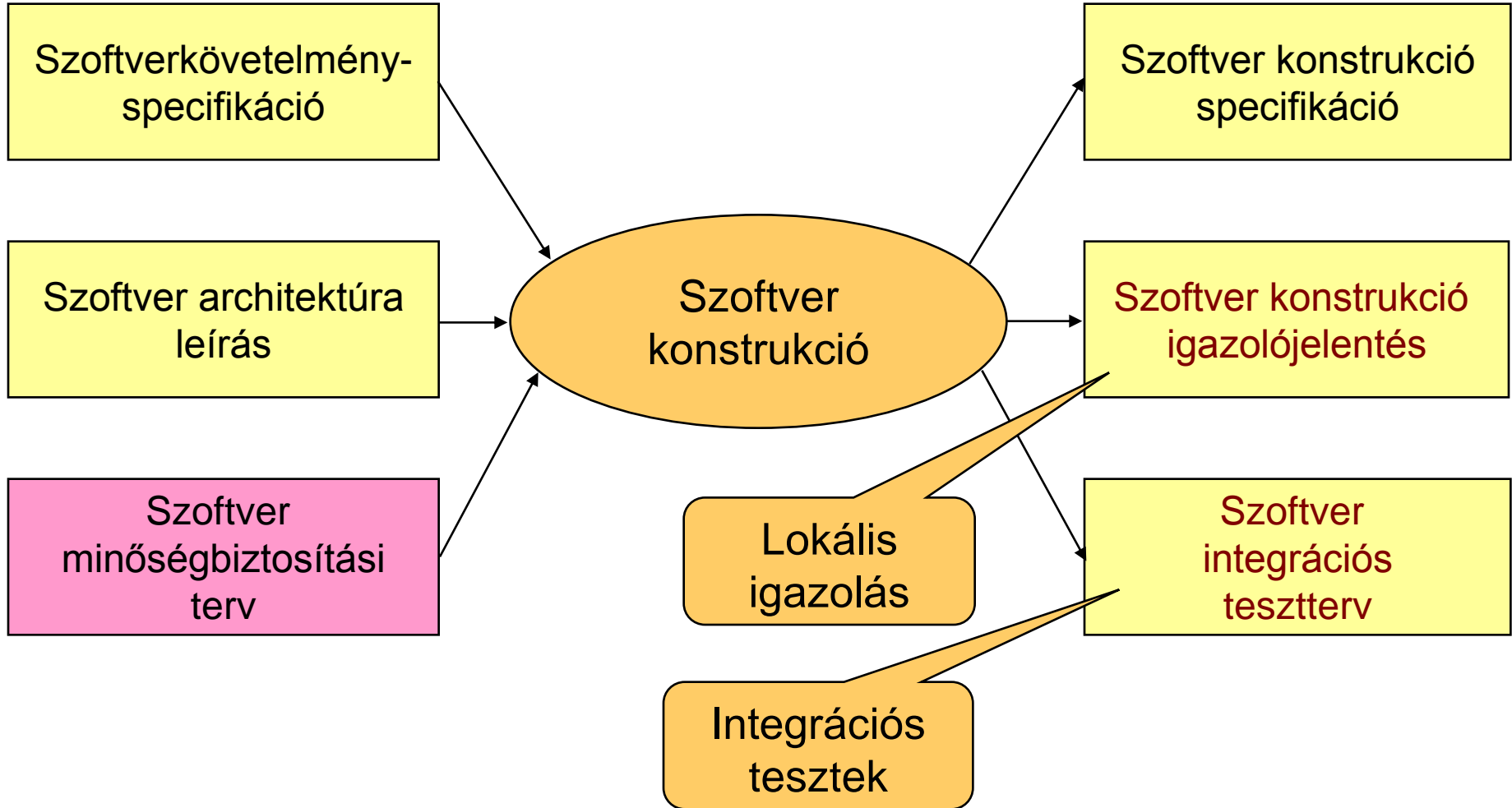
Méréstechnika és Információs Rendszerek Tanszék

<http://www.mit.bme.hu/~majzik/>

Tartalomjegyzék

- A részletes tervek elkészítése
 - Szoftver konstrukció
 - Modul tervezés
- Ellenőrzések
 - Verifikációs lépések
 - Formális verifikáció
- Alapszintű formalizmusok
 - KS, LTS, KTS
 - Automaták

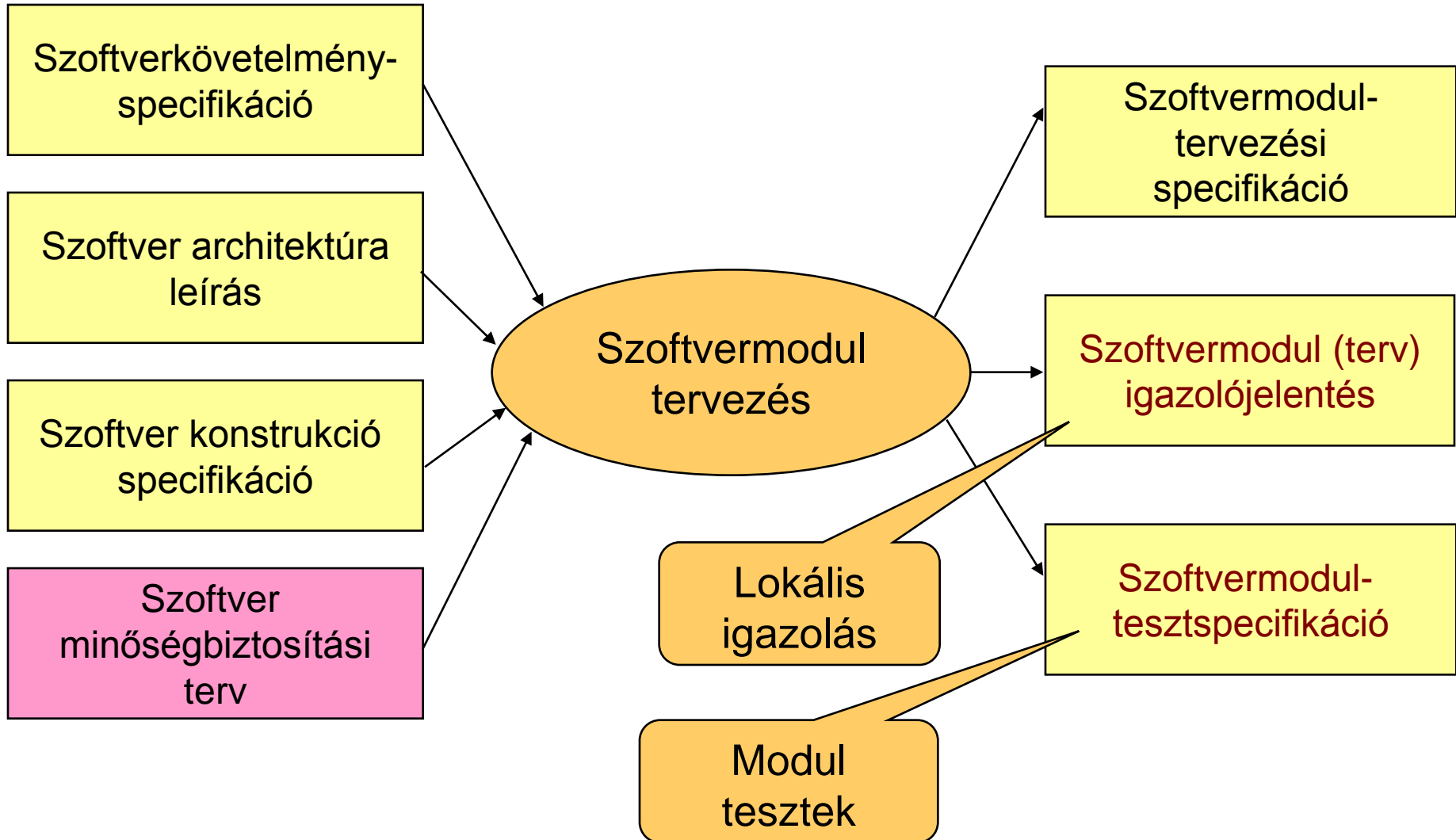
Szoftver konstrukció



Szoftver konstrukció

- Meghatározandó:
 - rendszerszintű algoritmusok a modulok együttműködéséhez
 - Globális adatstruktúrák
- Használt leíró nyelv:
 - Információáramlás leírása (sorrendiség, időbeliség)
 - Adatstruktúrák leírása
 - Absztrakció és modularitás, verifikálhatóság
- Választható módszerek:
 - Formális, félformális, strukturált módszerek
- Speciális előírások (biztonságkritikus rendszerekben):
 - Teljesen meghatározott interfészek
 - Modulméret korlátozás, információrejtés
 - Paraméter mennyiségi korlátozás

Szoftvermodul tervezés



Szoftvermodul-tervezési specifikáció

- Modulok belső tervezése
 - Algoritmusok
 - Adatstruktúrák
- Használt leíró nyelv:
 - Implementációközeli nyelvek
 - Pl. pszeudo-kód is
 - Absztrakt(abb) nyelvek
 - Formális, félformális, strukturált metodika
 - **Viselkedés leírás** is hangsúlyos

Tartalomjegyzék

- A részletes tervek elkészítése
 - Szoftver konstrukció
 - Modul tervezés
- **Ellenőrzések**
 - Verifikációs lépések
 - Formális verifikáció
- Alapszintű formalizmusok
 - KS, LTS, KTS
 - Automaták

Verifikációs lépések

Ellenőrizendők az igazolási fázisokban:

- A részletes terv tulajdonságai és megfelelőségei
 - Teljesség, ellentmondás-mentesség, megvalósíthatóság, ellenőrizhetőség
 - Megfelelőségek: Előző fázisokban rögzített követelmények alapján
- Teszt tervek teljessége

Verifikációs módszerek:

- Statikus elemzés
 - Ellenőrzőlisták, hibabecslés
 - Vezérlési folyamat elemzése (pl. strukturáltság)
 - Adatáramlás elemzése (pl. hozzáférési sorrend, inicializálás, felhasználás)
 - Határérték elemzés (pl. viselkedés határértékek esetén)
 - Nemkívánatos információáramlás („alattomos komponensek”)
 - Szimbolikus végrehajtás
- Dinamikus elemzés
 - Prototípus készítés és animáció
 - Szimuláció
- Formális verifikáció

Formális verifikáció

- Matematikai eszközök használata
 - Diszkrét matematika, matematikai logika
 - Formális nyelv: Formális szintaxis és szemantika
 - Tulajdonság leírás (követelmények: tulajdonság specifikáció)
 - Viselkedés leírás (terv, implementáció)
 - Matematikai algoritmus: analízis (verifikáció), szintézis
- Algoritmusok a verifikációhoz
 - „Önmagában való” vizsgálat (pl. ellentmondás-mentesség)
 - „Változások” vizsgálata (pl. tervezői döntések)
 - Tulajdonság leírás és viselkedés leírás összevetése
- Kritikus pont: A valós probléma formalizálása
 - Nem automatizálható
 - Egyszerűsítés, absztrakció gyakran szükséges (ezt validálni kell)

Formális szintaxis (áttekintés)

- Matematikai definíciók:

$KS = (S, R, L)$ és AP, ahol

$AP = \{P, Q, R, \dots\}$

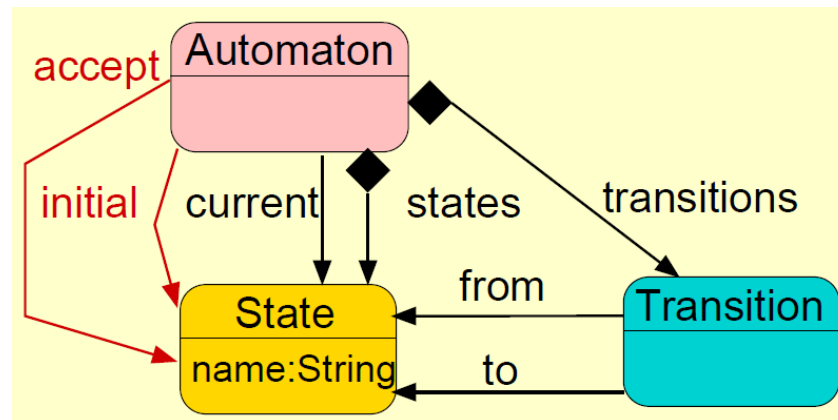
$S = \{s_1, s_2, s_3, \dots, s_n\}$

$R \subseteq S \times S$

$L: S \rightarrow 2^{AP}$

- BNF: $HML ::= true \mid false \mid p \wedge q \mid p \vee q \mid [a]p \mid \langle a \rangle p$

- Metamodell:



- Absztrakt szintaxis (nyelvtani szabályok)
és konkrét szintaxis (megjelenítés)

Formális szemantika (áttekintés)

- **Műveleti (operációs) szemantika: „Programozóknak”**
 - Megadja, mi történik a számítások során
 - Egyszerűbb formalizmusra épít: pl. állapotok, akciók
- **Axiomatikus szemantika: „Helyességbizonyításhoz”**
 - Állítás nyelv + axiómakészlet + következtetési szabályok
 - Elő- és utófeltételek adják meg a jelentést
 - Pl. automatikus tételbizonyító rendszerekhez
- **Denotációs szemantika: „Fordítóprogramokhoz”**
 - Szintaxis által meghatározott „jel \rightarrow jelölt dolog” leképezés
 - A „jelölt dolog” tipikusan mint matematikai domén adott
 - Számítási szekvencia, vezérlési gráf, állapothalmaz, ...
és ezeken definiált műveletek (összefűzés, unió, ...)
 - A modellek vizsgálata a mögöttes matematikai objektum vizsgálatára vezethető le
 - Komponálható szemantika: $|P \text{ op } Q| = |\text{op}| (|P|, |Q|)$

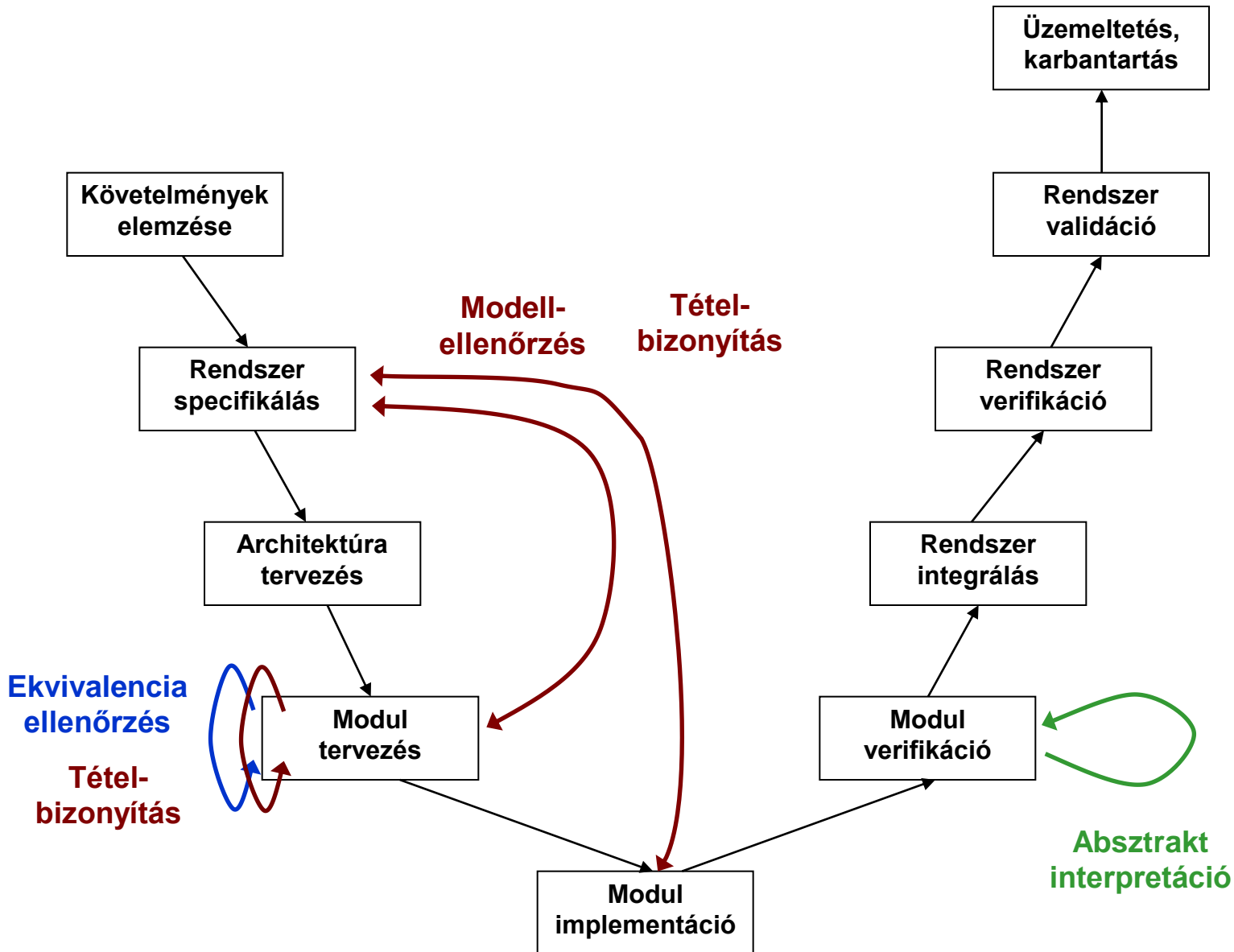
A legelterjedtebb formális verifikációs technikák

| Modell / technika | Viselkedés leírás (alapszintű) | Tulajdonság leírás (alapszintű) |
|--|---|--|
| Modellellenőrzés | Kripke struktúra, Kripke tranzíciós rendszer | Temporális logika, elsőrendű logika |
| Ekvivalencia ellenőrzés | LTS (Labeled Transition System), automata | LTS, automata (mint referencia viselkedés) |
| Tételbizonyítás | Dedukciós rendszer | Bizonyítandó tétel (elsőrendű logika) |
| Statikus analízis (absztrakt interpretáció) | Kripke struktúra (programból absztrakcióval) | Elsőrendű logika, assertion |

A technikák előnyei és korlátai

- **Modellellenőrzés, ekvivalencia ellenőrzés**
 - ☺ Teljesen automatikus, explicit kimerítő (teljes) vizsgálat
 - ☺ Ellenpélda generálás (debuggoláshoz)
 - ☹ Állapottér robbanás (részben kezelhető)
- **Tételbizonyítás**
 - ☺ Skálázható nagy rendszerekre (pl. indukció)
 - ☺ Nagy kifejezőerő
 - ☹ Interaktív (segítséget igényel, pl. ciklus invariánsok megtalálása, bizonyítási stratégia megadása)
 - ☹ Nincs ellenpélda
- **Statikus analízis (absztrakt interpretáció)**
 - ☺ Állapottér csökkentés
 - ☹ Absztrakció nehezen automatizálható

Verifikációs technikák szerepe



Tervek és modellek szerepe a formális verifikációban

- Viselkedés leírás (modell)

- 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 (követelmény)

- Alapszintű:

- Elsőrendű logika, temporális logika, referencia automata

- Magasabb szintű:

- Scenario: MSC, LSC, PSC, ...

Tartalomjegyzék

- A részletes tervek elkészítése
 - Szoftver konstrukció
 - Modul tervezés
- Ellenőrzések
 - Verifikációs lépések
 - Formális verifikáció
- **Alapszintű formalizmusok**
 - **KS, LTS, KTS**
 - **Automaták**

Alapszintű formalizmusok (áttekintés)

- Kripke-struktúrák (KS)
 - Állapotok, állapotátmenetek
 - Állapotok lokális tulajdonságai mint címkék
- Címkézett tranzíciós rendszerek (LTS)
 - Állapotok, állapotátmenetek
 - Állapotok lokális tulajdonságai mint címkék
- Kripke tranzíciós rendszerek (KTS)
 - Állapotok, állapotátmenetek
 - Állapotok és lokális tulajdonságai mint címkék
- Automaták
 - (Elfogadó) automaták véges hosszúságú szavakon
 - Büchi automaták végtelen hosszúságú szavakon

1. Kripke-struktúra

KS, Kripke-structure:

- **Állapotok** tulajdonságait fejezzük ki:
címkézés atomi kijelentésekkel
- Egy állapothoz sok címke rendelhető

Alkalmazás: Algoritmusok leírása

$KS = (S, R, L)$ és AP , ahol

$AP = \{P, Q, R, \dots\}$ atomi kijelentések halmaza (specifikus)

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

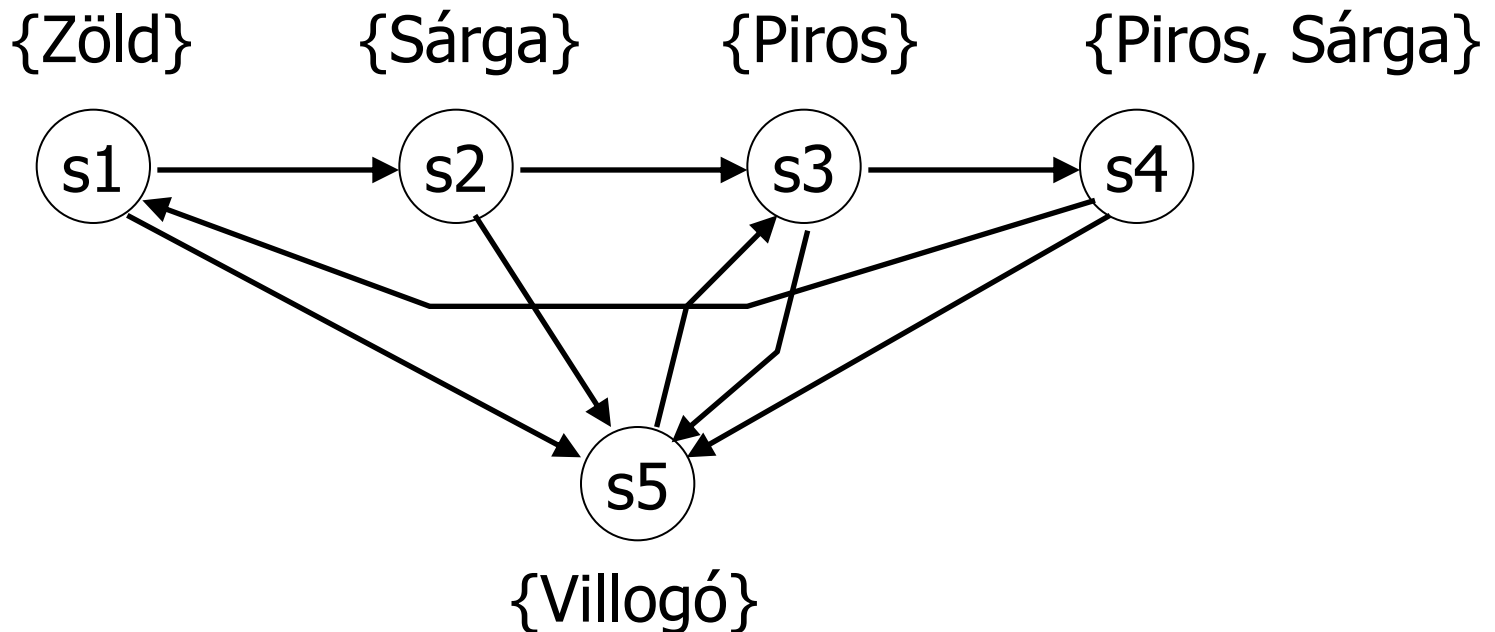
$R \subseteq S \times S$: állapotátmeneti reláció

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

Kripke-struktúra példa

Közlekedési lámpa vezérlője

- $AP = \{\text{Zöld}, \text{Sárga}, \text{Piros}, \text{Villogó}\}$
- $S = \{s1, s2, s3, s4, s5\}$



2. Címkezett tranzíciós rendszer

LTS, Labeled Transition System:

- **Állapotátmenetek** tulajdonságait fejezzük ki: címkezés akciókkal
- Egy átmeneten csak egy akció szerepelhet

Alkalmazás: Kommunikáció, protokollok modellezése

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

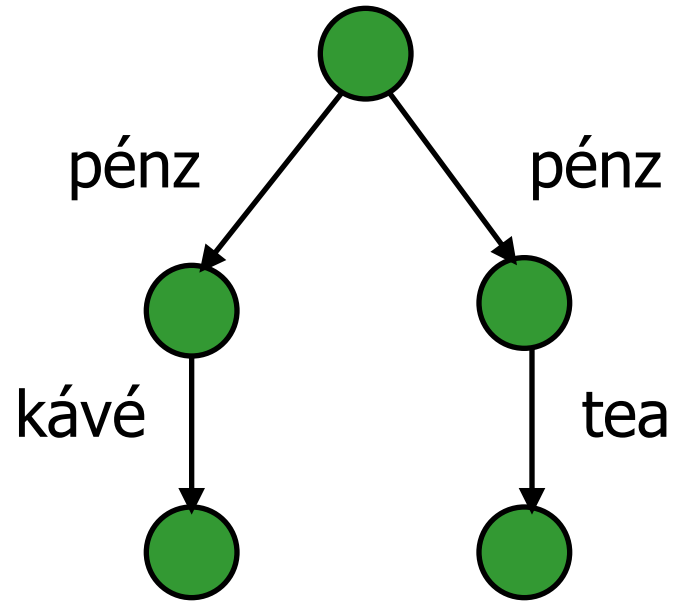
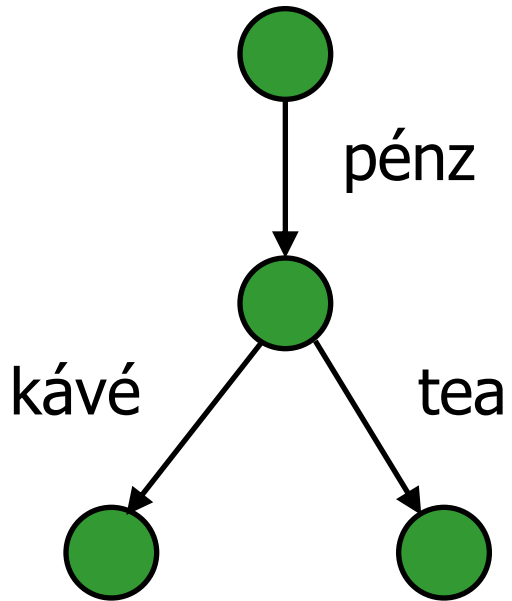
$S = \{s_1, s_2, \dots, s_n\}$ állapotok halmaza

$Act = \{a, b, c, \dots\}$ akciók (címkék) halmaza

$\rightarrow \subseteq S \times Act \times S$ címkezett tranzíciók.

Tranzíciók szokásos jelölése: $s_1 \xrightarrow{a} s_2$

LTS példák



3. Kripke tranzíciós rendszer

KTS, Kripke Transition System:

- Állapotok és átmenetek tulajdonságait is kifejezzük: címkézés atomi kijelentésekkel és akciókkal
- Egy állapothoz sok címke rendelhető, egy átmenethez egy címke rendelhető

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

$AP = \{P, Q, R, \dots\}$ atomi kijelentések halmaza (specifikus)

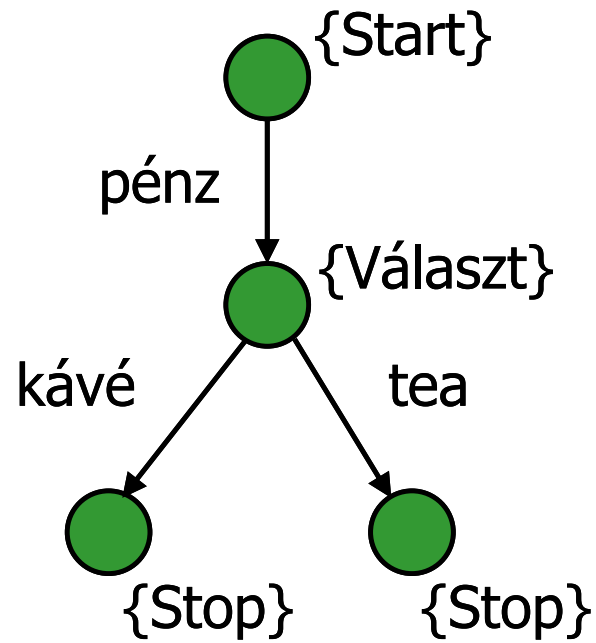
$Act = \{a, b, c, \dots\}$ akciók halmaza

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

$\rightarrow \subseteq S \times Act \times S$ tranzíció reláció

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

KTS példa



4. Automaták véges szavakon

- $A=(\Sigma, S, S_0, \rho, F)$ ahol
 - Σ az ábécé, S állapotok, S_0 kezdőállapotok
 - ρ az állapotátmeneti reláció, $\rho: S \times \Sigma \rightarrow 2^S$
 - F az elfogadó állapotok halmaza.
- Az automata futása:
 - Egy beérkező $w=(a_0, a_1, a_2, \dots a_n)$ betűsorozat hatására egy $r=(s_0, s_1, s_2, \dots s_n)$ állapotsorozat
 - r elfogadó futás: $s_n \in F$
 - Egy w szót elfogad az automata, ha létezik rá elfogadó futás
- $L(A)=\{ w \in \Sigma^* \mid w \text{ elfogadott} \}$
az automata által elfogadott nyelv

Automaták végtelen hosszúságú szavakon

- Nem ellenőrizhető, hogy a végállapot elfogadó-e
- Büchi elfogadási kritérium:
 - Egy beérkező $w=(a_0, a_1, a_2, \dots)$ betűsorozat hatására egy $r=(s_0, s_1, s_2, \dots)$ végtelen állapotsorozat
 - $\lim(r)=\{s \mid s \text{ előfordul végtelenül sokszor,}$
azaz nincs olyan j , hogy $\forall k>j:s \neq s_k\}$
 - Elfogadó futás: $\lim(r) \cap F \neq \emptyset$
 - Egy w szót elfogad az automata,
ha létezik rá elfogadó futás
(azaz végtelen sokszor érint elfogadó állapotot)
- $L(A)=\{w \in \Sigma^* \mid w \text{ elfogadott}\}$
az automata által elfogadott nyelv

Összefoglalás

- A részletes tervek elkészítése
 - Szoftver konstrukció
 - Modul tervezés
- Ellenőrzések
 - Verifikációs lépések
 - Formális verifikáció
- Alapszintű formalizmusok
 - KS, LTS, KTS
 - Automaták