Távközlési szoftverek

Tesztelés címkézett állapotátmeneti rendszerekkel

Kovács Gábor

Miről lesz szó?

- Absztrakt modellezés
- Processz algebra, a LOTOS (Language Of Temporal Ordering Specifications)
- Formális keretrendszer teszteléshez
- LTS (Labelled Transition System) alapú tesztelés

Felhasznált irodalom

- Salona A.A., Vives J.Q.,Gomez S.P.: "An introduction to LOTOS", DIT-Universidad Politecnica de Madrid 1993
- Tretmans J.: "Specification Based Testing with Formal Methods: A Theory", FORTE/PSTV 2000, Tutorial Notes, Pisa, 2000
- Nielsen B.: "LTS based testing – ioco", Aalborg University

Absztrakt modellezés 1

- A specifikáció a rendszer modellje egy adott absztrackiós szinten
- Az implementáció maga a rendszer
- Specifikálás=modellezés néhány relevánsnak kinevezett szempont absztrakttá tételével
- Implementálás=az absztrakt szempontok hozzáadása a modellhez

Absztrakt modellezés 2

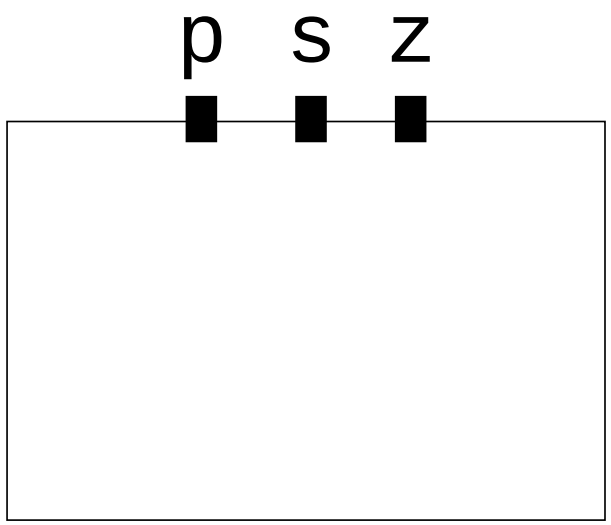
- A viselkedés leírása események időbeli rendezésével
- Mi az az esemény (event)? A kommunikáció darabja.
  - Megfigyelhető:
    - egy függvényhívás
    - visszatérés egy függvényhívásból
    - üzenet=(küldő, címzett, név, paramétererek)
  - Nem megfigyelhető:
    - egy értékadás
    - rekurzió
- Az esemény mindig atomi, azonnali és soros
- Rendszer=minden lehetséges eseménysor

Absztrakt modellezés 3

- Egy processz egy fekete doboz, amely néhány *kapun* (gate) keresztül kommunikál a külvilággal

- Mindek kapuhoz események tartoznak

- Kommunikáció=a kapu aktiválása=megfigyelhető esemény vagy akció



Processz algebra

- Több processzből álló rendszerben a processzek viszonyát definiálja
- Processz műveletek: alternatív kompozíció ($+$), szekvenciális kompozíció ($;$), párhuzamos kompozíció ($||$)
- Az operátorok precedenciája: $; > + > ||$
- Azonosságok:

```
x + y = y + x
x + ( y + z ) = ( x + y ) + z
x + x = x
( x + y ) ; z = ( x ; z ) + ( y ; z)
( x ; y ) ; z = x ; ( y ; z )
x || y = y || x
x || ( y || z ) = ( x || y ) || z
```

LOTOS

- A LOTOS processz fogalma
- Szekvenciális viselkedés: input, output, választás, feltétel, nem megfigyelhető akció, processz vége
- Párhuzamos viselkedés: átfedés, részleges szinkronizáció, teljes szinkronizáció
- Nem lesz szó az adattípusokról

Processz

- A processz definíció egy rendszer egy része viselkedésének a leírása
- Akciók szekvenciális végrehajtás
- `akcio ;`
- Lehetővé teszi rekurzió vagy iteráció absztrakt leírását
- `akcio ; P`, ahol `P` egy processz
- Választás (choice) = a környezet legalább két alternatívát kínál fel, tehát a rendszer többféleképpen viselkedhet.
- `Processz1 [felteteles_kifejezes] Processz2`
- Őrfeltétel: `[orfeltetel] -> Processz`
- Hierarchikus leírás
- Átmenetek elrejtése

Be-, illetve kimeneti esemény

- Megfigyelhető esemény: input vagy output egy kapun
- Input: érték elfogadása a környezettől egy kapun
- Az érték ismert típusú a fogadó számára
- Például: függvényhívás a hívott szempontjából, vagy visszatérés a hívó szempontjából
- `kapu ? valtozo : tipus`

- Output: érték felkínálása a környezet számára egy kapun
- Az érték egy kifejezés a küldőben
- Például: egy függvényhívás a hívó szempontjából, vagy visszatérés a hívott szempontjából
- `kapu ! kifejezes`

- Esemény hiánya (inaction): `i` vagy `stop`

Processz definíció

- Processz $\sim$ függvény a programozási nyelvekben, mert egyszer definiáljuk, névvel rendelkezik és többször végrehajthatjuk
- Processz végrehajtása = a processzdefiníció példányosítása
- Processzdefiníció:

```
PROCESS nev [kapuk] (parameterek) : visszateres :=
viselkedes
WHERE
lokalis definiciok
ENDPROC
```

- Processzhívás:

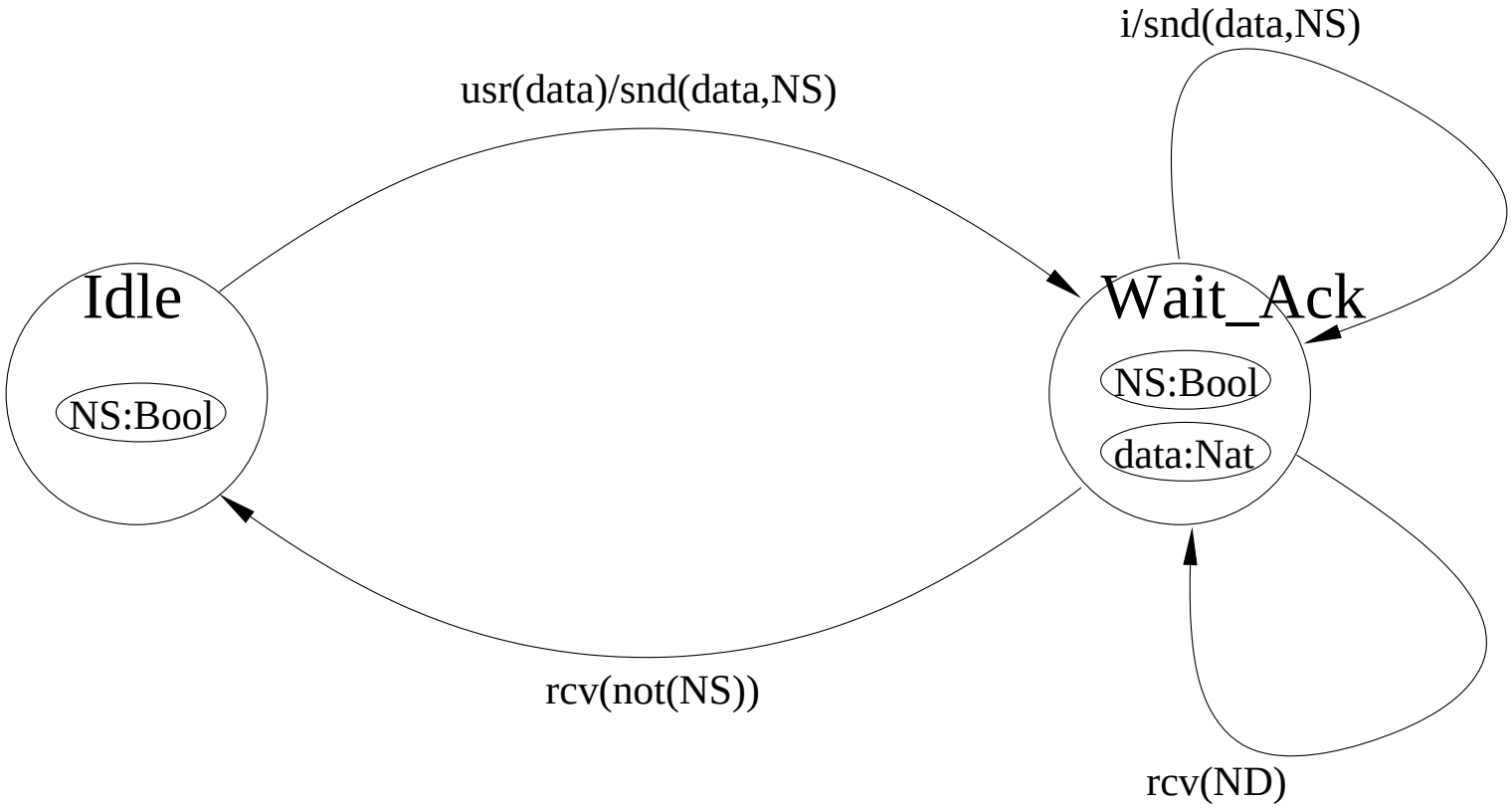
```
nev [aktualis_kapuk] (parameterek)
```

Kiterjesztett véges állapotgépek

- Extended Finite State Machine
- Véges állapotgép változókkal, értékadással, üzenet paraméterekkel, döntésekkel
- $EFSM = (S, V, I, O, T)$
- $T \subseteq (S \times V) \times (I \times P(V)*) \times (A(V) * \times O*) \times (S \times V)$
- Vö. FSM állapotátmeneti függvény: $h : S \times I \rightarrow O \times S$
- LOTOS-ban: egy állapot egy processz, a változók paraméterek, az átmenetek választás, következő állapot egy új processz példányosítása

Példa 1

- Az alábbi ábra egy EFSM-et mutat:



Példa 2

```
SPECIFICATION Alternating_bit_protocol [usr,snd,rcv] : NOEXIT
  LIBRARY Boolean, NaturalNumber ENDLIB
BEHAVIOUR Idle [usr,snd,rcv] (true) WHERE

PROCESS Idle [usr, snd, rcv] (NS:bool) : NOEXIT:=
usr?data:nat; snd!Data_Frame(data,NS);
Wait_Ack [usr,snd,rcv] (data,NS)
ENDPROC

PROCESS Wait_Ack [usr,snd,rcv] (data:nat,NS:bool) : NOEXIT :=
rcv?NR:bool;
( [NR ne NS] -> Idle [usr,snd,rcv] (not(NS))
[]
[NR eq NS] -> Wait_Ack [usr,snd,rcv] (data,NS) )
[]
i; snd!Data_Frame(data,NS); Wait_Ack [usr,snd,rcv] (data,NS)
ENDPROC
ENDSPEC
```

Párhuzamos viselkedés 1

- Interleaving: |||
- Két processz párhuzamos kapcsolása átfedés nélkül, a processzek függetlenül viselkednek, nem kommunikálnak egymással
- A kompozit rendszer állapotot vált, ha az egyik processz állapotot vált

- Szinkronizáció: a két processz egymással kommunikál legalább egy közös kapun keresztül
- A szoftvervilágban: függvényhívás
- A küldő processz kimeneti értéke elfogadható a fogadó processz számára bemenetként, ha annak típusa megegyezik a várttal
- A két processz egyszerre hajtja végre az átmenetét, a környezet számára csak egy esemény látszik

Párhuzamos viselkedés 2

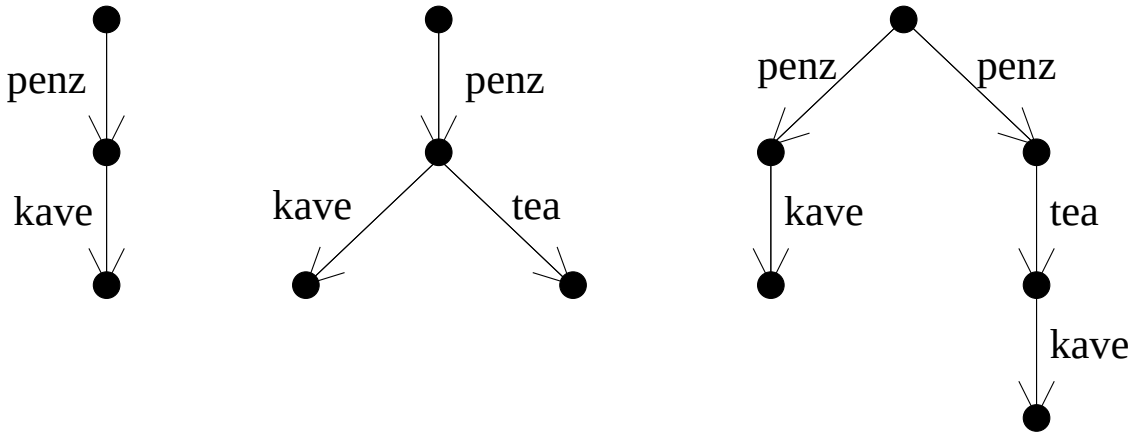
- Részleges szinkronizáció: bizonyos kapuk szinkronizáltak, mások függetlenek
- `Processz1[kapuk1] |[szinkronizalt_kapuk]| Processz2[kapuk2]`
- Teljes szinkronizáció: minden kapu szinkronizált
- `Processz1||Process2`
- A kompozit rendszer állapotot vált, ha a két processz egyszerre állapotot vált
- A rendszer holtpontra jut, ha nincs megfigyelhető akciója
- Akkor következik be, ha nincs átfedő esemény, és a szinkroniált események nem kompatibilisek
- A rendszereket holtpontmentesre kell tervezni!

Címkézett állapotátmeneti rendszerek 1

- Egy LOTOS processz ábrázolható egy fával
- A csomópontok állapotok, amelyek nem címkézettek
- Szekvenciális végrehajtás $\rightarrow$ 1 kimenő fokszám az aktuális csomópontból
- Elágazás $\rightarrow$ $+1$ kimenő fokszám egy csomópontból alternatívánként
- Élek címkéje $\rightarrow$ megfigyelhető akció neve (input vagy output)
- $LTS = (S, L, T, s_0)$
- Ahol $T \subseteq S \times L \times S$

Címkézett állapotátmeneti rendszerek 2

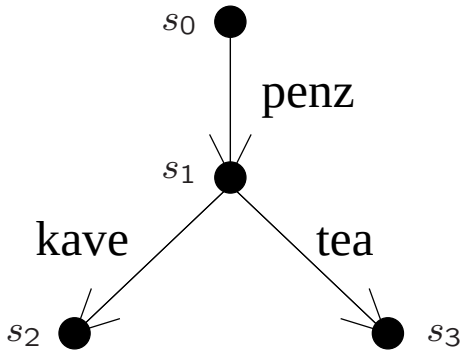
- Modellezzünk egy kávéautomatát
- Példa: legyen $L = \{penz, kave, tea\}$



Címkézett állapotátmeneti rendszerek 3

Definíciók, jelölések:

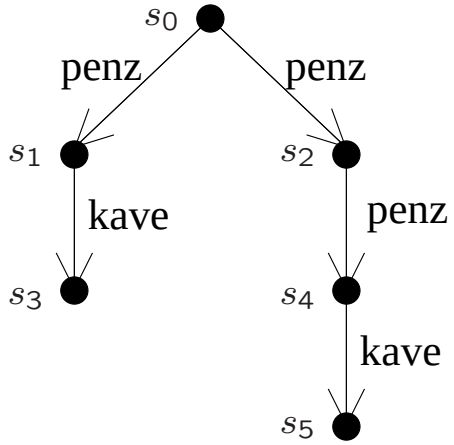
- Állapotátmenet: $s_1 \xrightarrow{a} s_2$, ahol $a \in L$
- Állapotátmenet-sorozat: $s_1 \stackrel{\sigma}{\Rightarrow} s_2$, ahol $\sigma = a_1 \ldots a_n \in L*$
- Végrehajtható akciók: $s_1 \stackrel{\sigma}{\Rightarrow}$
- Nem végrehajtható akciók: $s_1 \stackrel{\sigma}{\not\Rightarrow}$



Itt $s_0 \xrightarrow{penz} s_1$, $s_0 \stackrel{penz\ tea}{\Longrightarrow}$, $s_0 \stackrel{penz\ kave}{\Longrightarrow} s_2$, $s_0 \stackrel{penz}{\Longrightarrow} / \stackrel{penz}{\Longrightarrow}$

Címkézett állapotátmeneti rendszerek 4

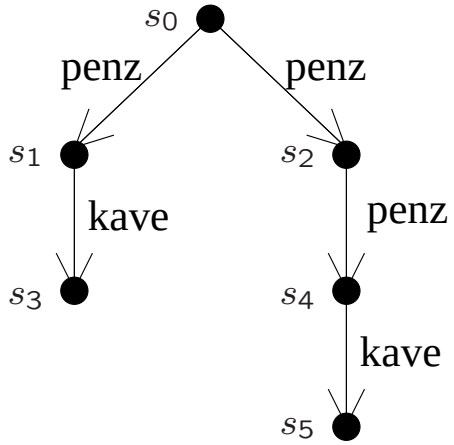
- Nyomvonal: megfigyelhető események sorozata :
- $traces(p) = \{\sigma \in L* | p \stackrel{\sigma}{\Rightarrow}\}$
- Elérhető állapotok halmaza egy nyomvonal után:
- p **after** $\sigma = \{p' | p \stackrel{\sigma}{\Rightarrow} p'\}$



Itt $traces(s_0) = \{\varepsilon, penz, penz\ kave, penz\ penz, penz\ penz\ kave\}$
s_0 **after** $penz = \{s_1, s_2\}$
s_0 **after** $penz\ penz = \{s_4\}$

Címkézett állapotátmeneti rendszerek 5

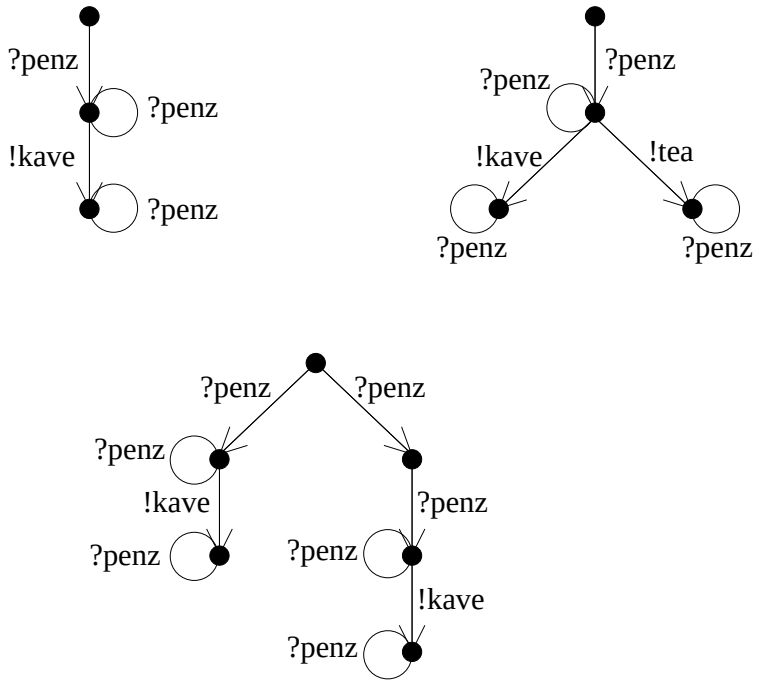
- Elutasító halmaz $A \subseteq L$
- $p \xrightarrow{A} p \iff \forall a \in A : p \overset{a}{\not\rightarrow}$
- Párhuzamos kompozíció:
- $\dfrac{p \xrightarrow{a} p', q \xrightarrow{a} q', a \in L}{p||q \xrightarrow{a} p'||q'}$



- Elutasító átmenetek:
  $s_2 \xrightarrow{\{kave,\ tea\}} s_2$,
  $s_3 \xrightarrow{L} s_3$

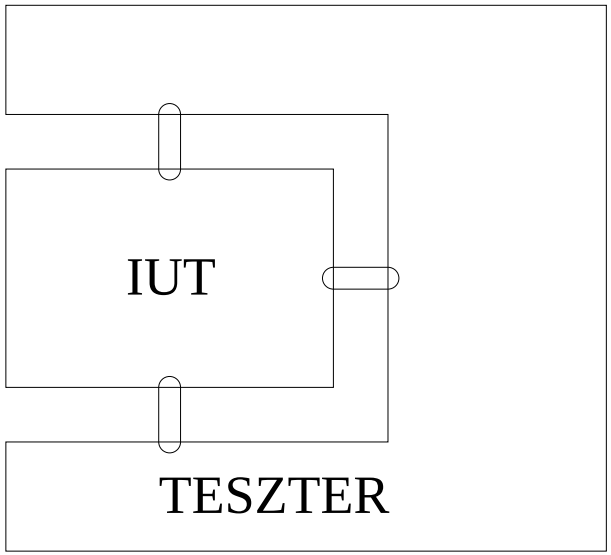
Input/output állapotátmeneti rendszer

- Az LTS speciális esete, $IOTS(L_I, L_O) \subseteq LTS(L_I \cup L_O)$
- Az input és output halmaz szétválasztása
- $IOTS = (S, L_I, L_O, T, s_0)$, ahol $L = L_I \cup L_O$ és $L_I \cap L_O = \emptyset$
- A bemenet mindig engedélyezett: $\forall s \in S : \forall ?a \in L_I : s \xrightarrow{?a}$
- $L_I = \{?penz\}$, $L_O = \{!kave, !tea\}$



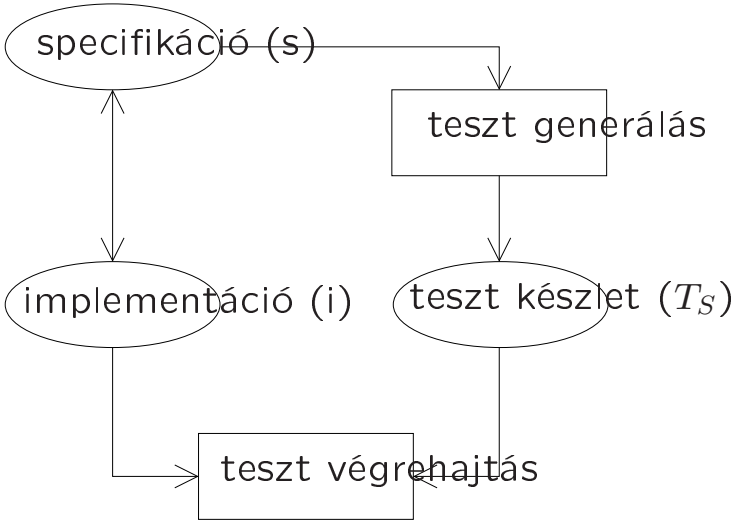
Tesztelés 1

- Egy objektum minőségének ellenőrzése kontrollált kísérletek végrehajtásával
- A tesztelés csak a hibák előfordulát deríti fel, azok hiányát nem
- A tesztelés két processz interakciója: IUT (Implementation Under Test) és teszter
- Három kapu: alsó és felső, valamint kontroll interfész



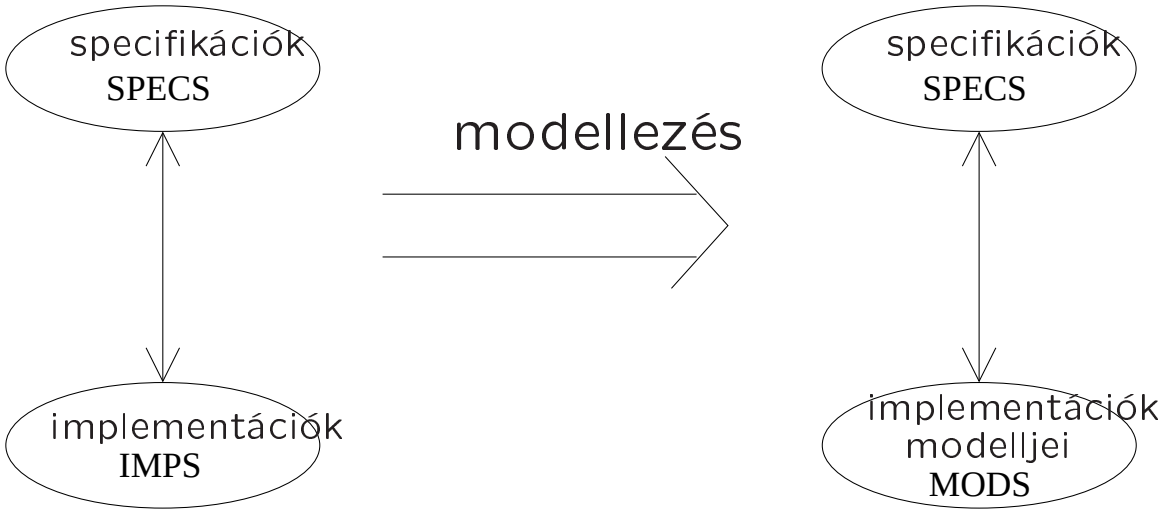
Tesztelés 2

- A követelmények alapján készül egy formális specifikáció
- A formális specifikációból származik az implementáció és a teszt készlet
- A teszt végrehajtás ítéletet hoz a specifikáció (közvetve a követelmények) és az implementáció relációjáról



Konformancia reláció 1

- Az s specifikáció a specifikációk $SPECS$ halmazának egy eleme
- Az IUT implementáció a implementációk $IMPS$ halmazának egy eleme
- A konformancia reláció (**conforms**): $SPECS \times IMPS \rightarrow \{\mathbf{pass}, \mathbf{fail}\}$
- Hipotézis: minden $IUT \in IMPS$ modellezhető egy $i_{IUT} \in MODS$ modellel
- Implementáció reláció (**imp**): $SPECS \times MODS \rightarrow \{\mathbf{pass}, \mathbf{fail}\}$



Megfigyelés és tesztelés 1

- A teszt eset az adott input/output ábécé feletti tesztek halmazának egy eleme: $t \in TESTS$
- A teszt készlet az adott input/output ábécé feletti tesztek halmazának részhalmaza: $T \subseteq TESTS$
- Teszt végrehajtás $EXEC : TESTS \times IMPS \rightarrow \{\mathbf{pass}, \mathbf{fail}\}$
- A megfigyelés (OBS) a teszt végrehajtások során észlelt akciószekvenciák halmaza
- A teszt végrehajtás modellezése $obs : TESTS \times MODS \rightarrow \{\mathbf{pass}, \mathbf{fail}\}$
- Az ítélet a megfigyelések leképezése egy kételemű halmazra: $v_t : (OBS) \rightarrow \{\mathbf{pass}, \mathbf{fail}\}$
- A hipotézis formálisan: $\forall IUT \in IMPS \; \exists i_{IUT} \in MODS : \forall t \in TESTS : EXEC(t, IUT) = obs(t, i_{IUT})$

Megfigyelés és tesztelés 2

- IUT **passes** $t \iff v_t(EXEC(t, IUT)) = \mathbf{pass}$
- IUT **passes** $T \iff \forall t \in T : IUT$ **passes** t
- IUT **fails** $T \iff \exists t \in T : v_t(EXEC(t, IUT)) = \mathbf{fail}$
- A gyakorlat szerint, ha IUT **passes** $T_S \iff IUT$ **conforms** s
- A hipotézist felhasználva: IUT **conforms** $s \iff i_{IUT}$ **imp** s
- Ez akkor igaz, ha T_S teljes, vagyis az összes az **imp**-relációnak megfelelő implementációt karakterizálja
- Ha az i **imp** s reláció fenállásából következik, hogy az i **passes** T_S megfigyelés, akkor a teszt készlet *alapos* (sound)
- Ha az i **passes** T_S megfigyelésből következik, hogy az i **imp** s reláció fennáll, akkor a teszt készlet *kimerítő* (exhaustive)

A keretrendszer és a processz algebra kapcsolata

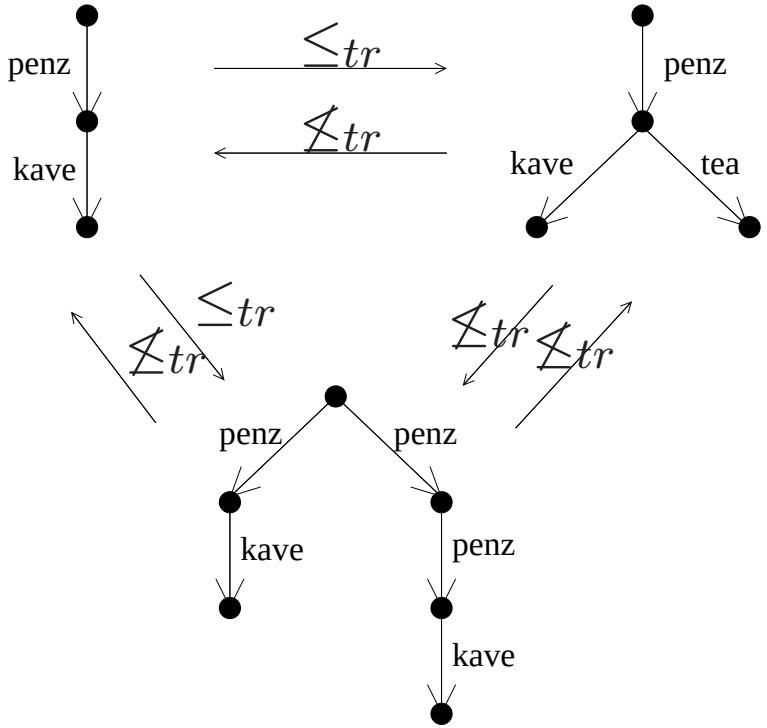
- Legyenek a $SPECS$, $MODS$ és $TESTS$ halmaz elemei LTS modellek!
- Ekkor az **imp** implementáció relációk két LTS relációi lesznek!
- A megfigyelés modellje (obs) nem más, mint a teszt eset és az implementáció teljes szinkronizációja
- Az OBS halmaz nyomvonalakat tartalmaz.
- Implementáció relációk erősség szerint: izomorfizmus, biszimuláció, elutasítás, hiba, nyomvonal szerinti előrendezés
- **ioco**

Biszimuláció szerinti előrendezés

- A szimuláció egy reláció két állapot között: (S, L, T) esetén $R \subseteq S \times S$. Itt most $R \subseteq S \times S'$, tehát két LTS állapotai között keresünk leképezést.
- $\forall (p, q) \in R : \forall p' \in S : p \xrightarrow{a} p' \Rightarrow q \xrightarrow{a} q', q' \in S$, ahol $a \in L$, és $(p', q') \in R$
- Ha q szimulálja p-t (q **simulates** p), akkor azt mondjuk, hogy a két állapot hasonló, rendezési relációjuk: $p \leq q$
- Ha a reláció szimmetrikus, azaz $p \xrightarrow{a} p' \iff q \xrightarrow{a} q'$, akkor biszimulációról beszélünk
- A biszimuláció egy ekvivalencia reláció
- Gyenge biszimulációról beszélünk, ha a címkehalmaz tartalmaz egy $\delta \in L$ nem megfigyelhető akciót

Nyomvonal szerinti előrendezés

- Trace preorder $\leq_{tr}$
- $i \leq_{tr} s \iff traces(i) \subseteq traces(s)$



- Elutasító átmenetek:
  $s_2 \xrightarrow{\{kave,\ tea\}} s_2$,
  $s_3 \xrightarrow{L} s_3$

Nyomvonalak implicit átmenetekkel

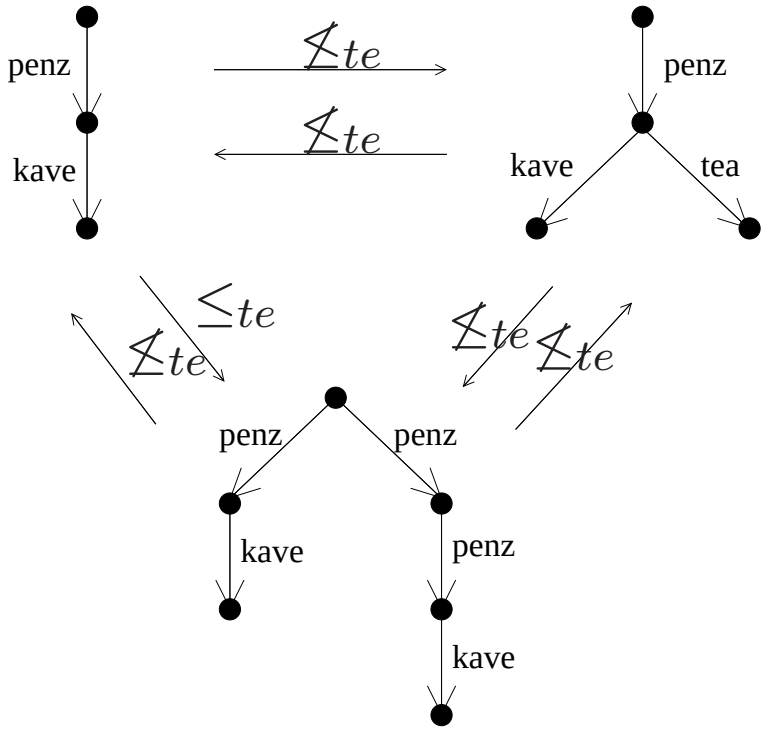
- Az implicit átmenetek engedélyezettek: $p \xrightarrow{\delta} p = p \xrightarrow{!x} p$, ha $!x \in L_O : p \overset{!x}{\not\rightarrow}$
- Az implicit átmenet nem megfigyelhető, de megtörténte időzítő használatával valószínűsíthető
- A kimeneti esemény függvény: $out : S \rightarrow L_O$, vagy egy kimeneti eseményt (címkét) ad vissza, vagy az implicit eseményt (δ): $out(p) = \{!x \in L_O | p \xrightarrow{!x}\} \cup \{\delta | p \xrightarrow{\delta} p\}$
- Vegyük be implicit átmeneteket a nyomvonalak közé (suspension traces): $Strace(p) = \{\sigma \in (L \cup \{\delta\}) * | p \overset{\sigma}{\Rightarrow}\}$
- Tehát, ha nincs lehetséges kimenet, akkor timeout (δ) történik a rendszerben

Tesztelési előrendezés 1

- A processzeket megfigyelt viselkedése alapján hasonlítjuk össze
- i **imp** $s \iff \forall u \in U : obs(u, i) \subseteq obs(u, s)$
- A s és u vagy képes minden eseményt szikronizáltan végrehajtani, vagy ugyanakkor terminálódnak.
- $obs(u, s) = \{\sigma \in L* | u || s \stackrel{\sigma}{\Rightarrow} \mathbf{nil}\} \cup \{\sigma \in L* | u || s \stackrel{\sigma}{\Rightarrow}\}$

Tesztelési előrendezés 2

Csak a következő elágazás nélküli fákat tekintve teszt esetnek:
$U = \{\varepsilon, penz, penz\ kave, penz\ tea, penz\ penz\ kave\}$



$u||s_1 = \{\varepsilon, penz, penz\ kave\}$,$u||s_2 = \{\varepsilon, penz, penz\ kave, penz\ tea\}$, $u||s_3 = \{\varepsilon, penz, penz\ kave, penz\ penz\ kave\}$

Tesztelési előrendezés 3

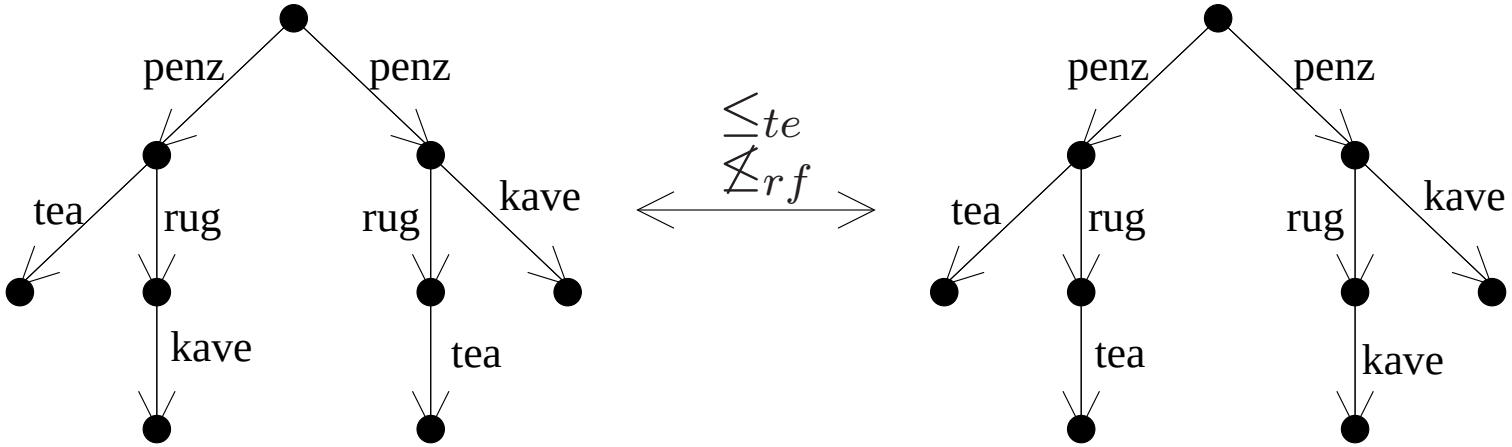
- A tesztelési előrendezés nem más, mint a hiba szerinti előrendezés, vagyis az egy-egy nyomvonal utáni elutasított akciók halmazai szerinti előrendezés
- $i \leq_{te} s \iff refusals(i) \subseteq refusals(s)$
- p **after** σ **refuses** $A \iff \exists p' \in p$ **after** $\sigma : \forall a \in A : p' \stackrel{a}{\not\Rightarrow}$
- $refusal(p) = \{\langle \sigma, A \rangle | A \subseteq L, \sigma \in traces(p), p$ **after** σ **refuses** $A\}$

Elutasítás szerinti előrendezés 1

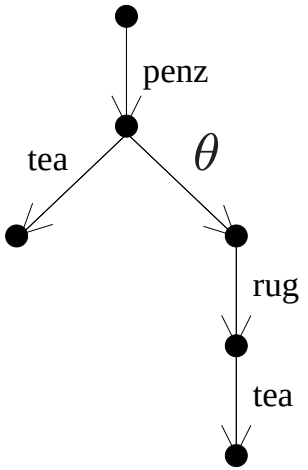
- A tesztelési előrendezés kiegészítése implicit eseményekkel
- Az implicit esemény itt teszter oldali timeout-ot jelent: $L_\theta = L \cup \{\theta\}$
- A szinkronizált viselkedés: $\frac{u \xrightarrow{\theta} u', \forall a \in L : u||s \overset{a}{\not\rightarrow}}{u||s \xrightarrow{\theta} u'||s}$, vagyis u és s nem tud szinkronizáltan kommunikálni, tehát $u||s$ holtpontra jutott. Ezt jelzi a θ timeout.
- $obs_\theta(u, s) = \{\sigma \in L_\theta * | u||s \overset{\sigma}{\Rightarrow} \mathbf{nil}\} \cup \{\sigma \in L_\theta * | u||s \overset{\sigma}{\Rightarrow}\}$
- $i \leq_{rf} s \iff Ftraces(i) \subseteq Ftraces(s)$, ahol $Ftrace$ a hiba nyomvonal
- azaz létezik egy $A \subseteq L$ eseményhalmaz: $p \overset{A}{\rightarrow} p = \forall a \in A : p \overset{a}{\not\rightarrow}$
- ahol $Ftraces(p) = \{\sigma \in (L \cup \mathcal{P}(L)) * | p \overset{\sigma}{\Rightarrow}\}$
- A s és u-ra igaz a tesztelési előrendezés, és a előrendezés az elutasított szekvenciákra is fennáll, tehát $i \leq_{rf} s \Rightarrow i \leq_{te} s$

Elutasítás szerinti előrendezés 2

- A tesztelési előrendezés kiegészítése implicit eseményekkel



- A megkülönböztető fa a következő:
- A megkülönböztető $Ftrace$ a következő: $penz\ \{tea\}\ rug\ kave$



I/O tesztelési előrendezés 1

- I/O rendszerek megfigyelése, ahol a rendszer kimeneteket küld a környezet felé, és bemeneteket fogad a környezet felől.
- i **imp** s, ahol $i \in IOTS, \mathbf{imp} \in IOTS \times LTS, s \in LTS$
- A tesztelési előrendezés kiterjesztése I/O rendszerekre
- $i \leq_{iot} s \iff \forall u \in U : obs(u, i) \subseteq obs(u, s)$, ahol $U = IOTS(L_I, L_O)$

I/O tesztelési előrendezés 2

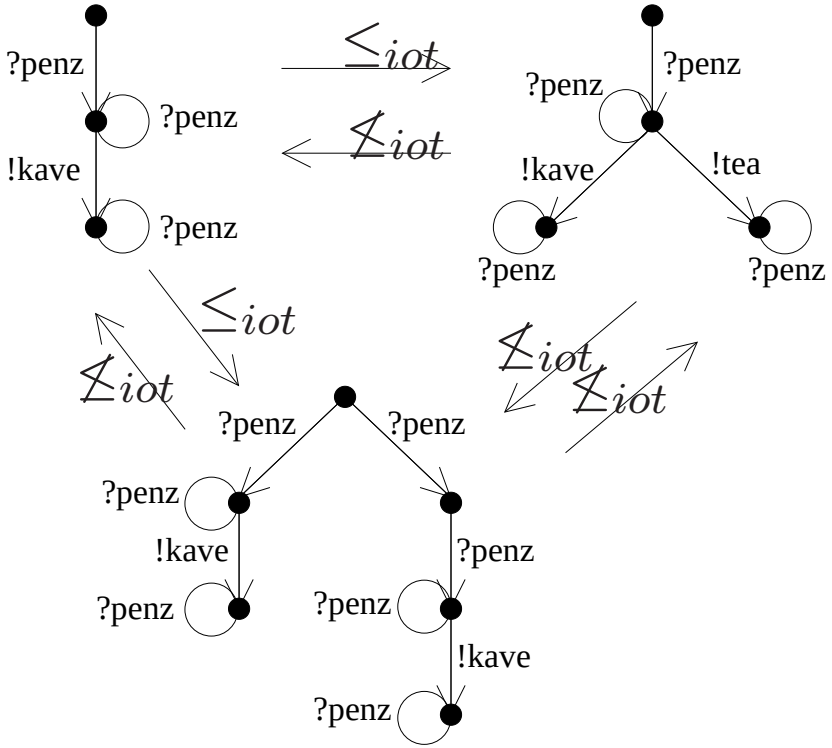
- Tudjuk, hogy $i \leq_{te} s \iff refusals(i) \subseteq refusals(s)$
- Most i nem utasíthat el inputot és u nem utasíthat el outputot: i **after** σ **refuses** $A \Rightarrow A = \emptyset$ vagy $A = L_O$
- Értelmezhetjük a nyugalmi nyomvonalakat: $Qtraces$, amiknek megfigyelt nincs kimenet a végén
- Az p állapot nyugalmi, ha $\forall !x \in L_O : p \overset{!x}{\not\rightarrow}$
- A σ nyomvonal nyugalmi, ha $p \overset{\sigma}{\Rightarrow} p'$ és p' nyugalmi állapot
- $Qtraces(p)$ a p állapotból induló nyugalmi nyomvonalak: $Qtraces(p) = \{\forall \sigma \in L*, !x \in L_O | p \overset{\sigma}{\Rightarrow} p' \overset{!x}{\not\rightarrow}\}$
- $i \leq_{iot} s \iff traces(i) \subseteq traces(s)$ és $Qtraces(i) \subseteq Qtraces(s)$

I/O tesztelési előrendezés 3

- Ekvivalens megfogalmazással:
- $i \leq_{iot} s \iff i$ **after** σ **refuses** $A \Rightarrow s$ **after** σ **refuses** A
- Ez két feltétel együttes fennállását feltételezi:
  - $\{\sigma | \sigma \in traces(i), i$ **after** σ **refuses** $\emptyset\} \subseteq$
    $\{\sigma | \sigma \in traces(s), s$ **after** σ **refuses** $\emptyset\}$
  - $\{\sigma | \sigma \in traces(i), i$ **after** σ **refuses** $L_O\} \subseteq$
    $\{\sigma | \sigma \in traces(s), s$ **after** σ **refuses** $L_O\}$

I/O tesztelési előrendezés 4

- Legyen $U = \{\varepsilon, ?penz, ?penz?penz, ?penz!kave, ?penz!tea, ?penz?penz!kave, ?penz?penz?penz, ?penz!kave?penz, ?penz!tea?penz, ?penz?penz!kave?penz\}$



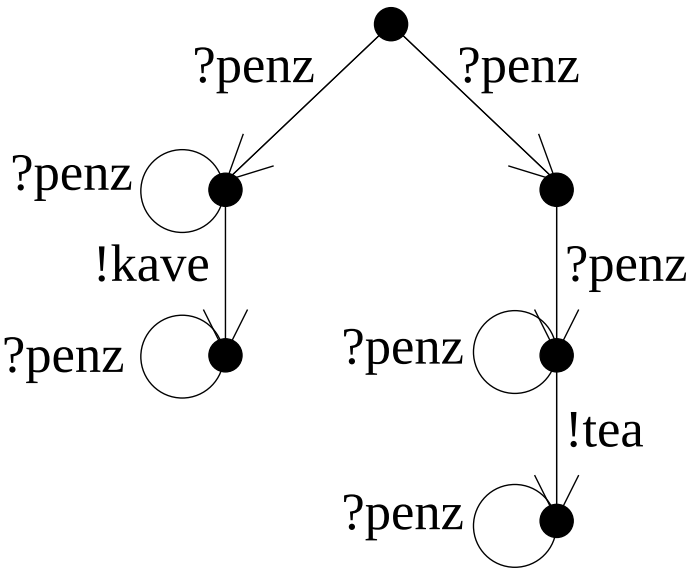
I/O elutasítási előrendezés 1

- Az elutasítási előrendezés általánosítása I/O esetre
- A tesztelési előrendezés kiegészítése timeout-tal
- Tehát $U = IOTS(L_I \cup \{\theta\}, L_O)$
- $i \leq_{ior} s \iff Straces(i) \subseteq Straces(s)$
- $p \xrightarrow{\delta} p \iff \forall !x \in L_O : p \overset{!x}{\not\rightarrow}$
- $L_O^\delta = L_O \cup \{\delta\}$,$L^\delta = L \cup \{\delta\}$
- $Straces(s) = \{\sigma \in (L^\delta) * | s \overset{\sigma}{\Rightarrow}\}$
- Más módon leírva:
- $i \leq_{ior} s \iff \forall \sigma \in (L^\delta)* : out(i \textbf{ after } \sigma) \subseteq out(s \textbf{ after } \sigma)$,
- ahol $out(p \textbf{ after } \sigma = \{!x \in L_O^\delta | p \overset{\sigma}{\Rightarrow} p' \xrightarrow{!x}\}$

I/O elutasítási előrendezés 2

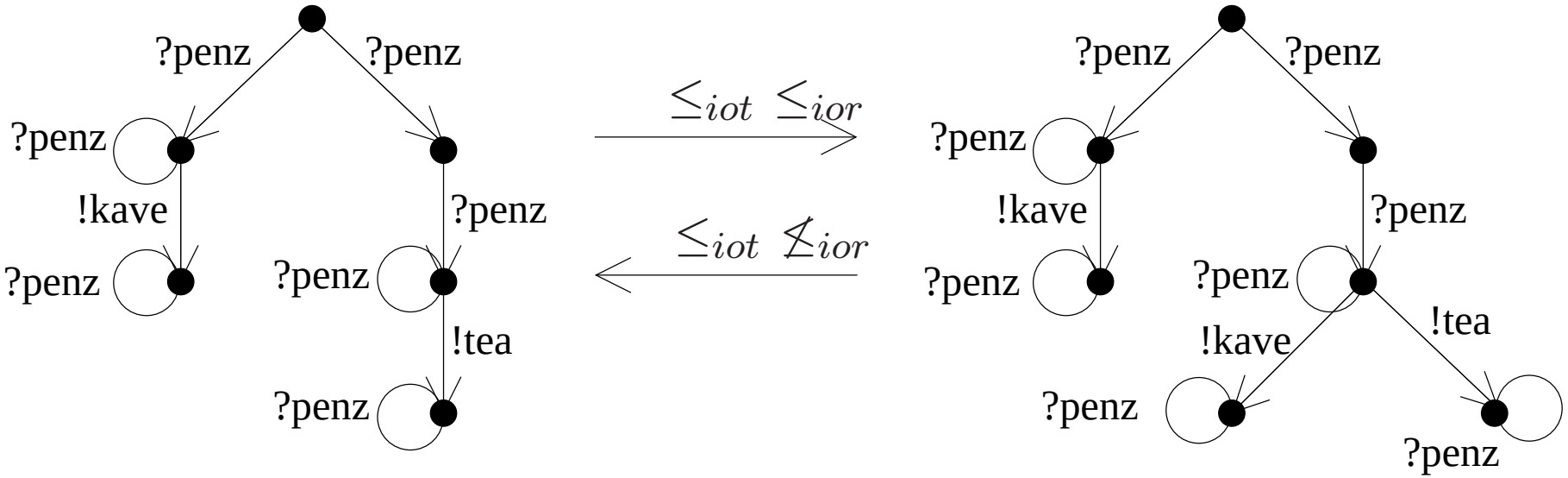
- $i \leq_{ior} s \iff$
  - ha i $!x$ kimenetet generál a σ szekvencia után, akkor s is képes kell, hogy legyen erre
  - ha i σ után minden kimenetet elutasít, akkor s is képes kell, hogy legyen erre
- $i \leq_{ior} s$ erősebb, mint $i \leq_{iot} s$

- $?penz\ \delta \in Qtraces(s) \subseteq Straces(s)$
- $?penz\ \delta\ ?penz\ !tea\ \delta \in Qtraces(s) \subseteq Straces(s)$
- $penz\ \delta\ ?penz\ !tea\ \delta \in Straces(s) \notin Qtraces(s)$



Input/output elutasítás szerinti előrendezés 3

- $out(r_1 \textbf{ after } penz\ \delta\ penz) = \{tea\}$
- $out(r_2 \textbf{ after } penz\ \delta\ penz) = \{kave, tea\}$



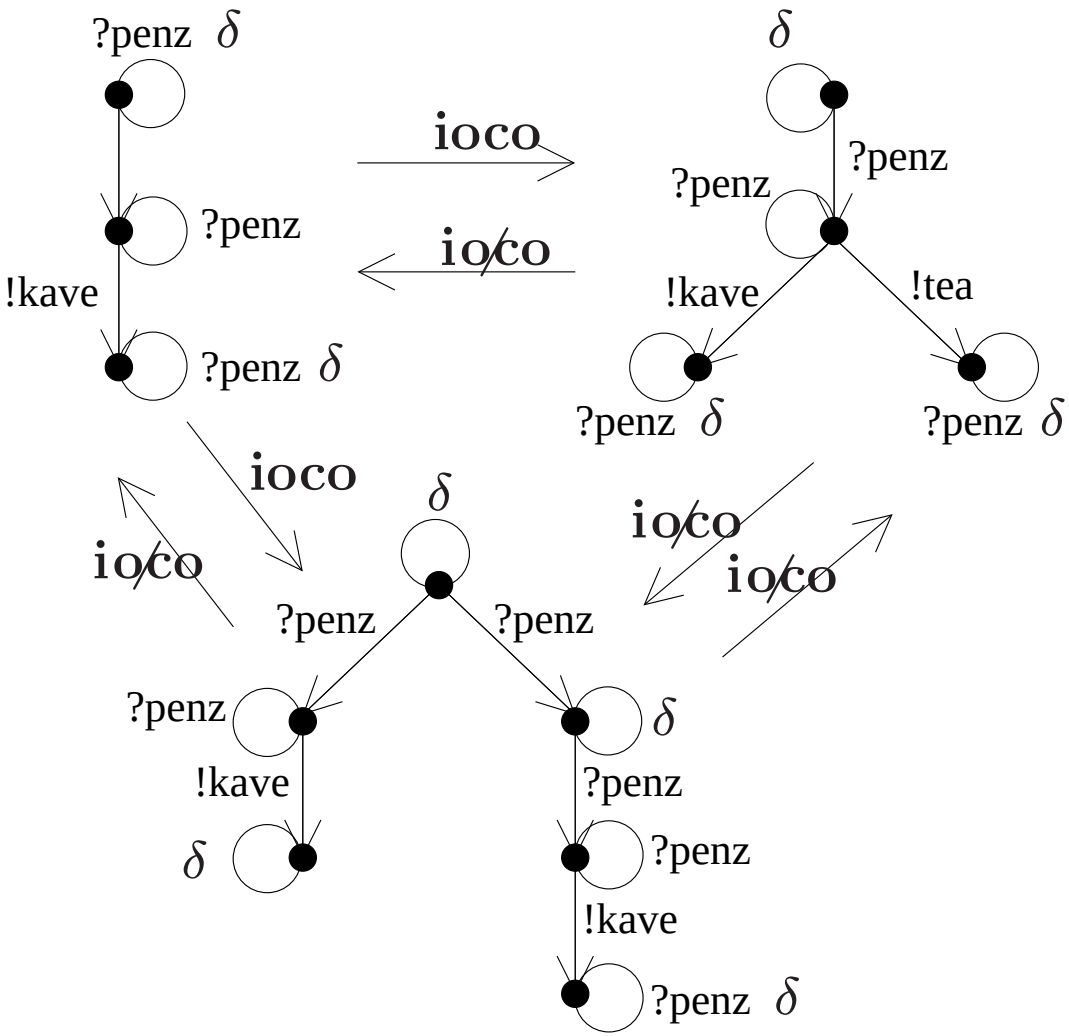
Konformancia 1

- A gyakorlatban az összes nyomvonalat nem nézhetjük végig, ezért
- $i \leq_{te} s \iff refusals(i) \subseteq refusals(s)$ helyett
- i **conf** $s \iff \forall \sigma \in traces(s) : \langle \sigma, A \rangle \in refusals(i) \Rightarrow \langle \sigma, A \rangle \in refusals(s)$
- illetve I/O esetben
- $i \leq_{ior} s \iff \forall \sigma \in (L^{\delta})*$ helyett
- i **ioco** $s \iff \forall \sigma \in Straces(s) : out(i \textbf{ after } \sigma) \subseteq out(i \textbf{ after } \sigma)$

Konformancia 2

- Legyen az események halmaza: $L = L_I \cup L_O$, ahol $L_I \cap L_O = \emptyset$
- Legyen $s \in SPECS = LTS(L_I \cup L_O)$ és $i_{IUT} \in MODS = IOTS(L_I, L_O)$
- Ekkor az implementáció reláció: $\mathbf{ioco} \subseteq IOTS \times LTS$
- Az i $\mathbf{ioco}$ s reláció feltételei:
  - ha i $!x$ kimenetet produkálja a σ nyomvonal után, akkor s is képes $!x$ kimenetet produkálni a σ nyomvonal után
  - ha i nem képes semmilyen kimenetet produkálni a σ nyomvonal után, akkor s minden kimenetet elutasíthat a σ nyomvonal után
- ahol σ egy $Strace$

Konformancia 3

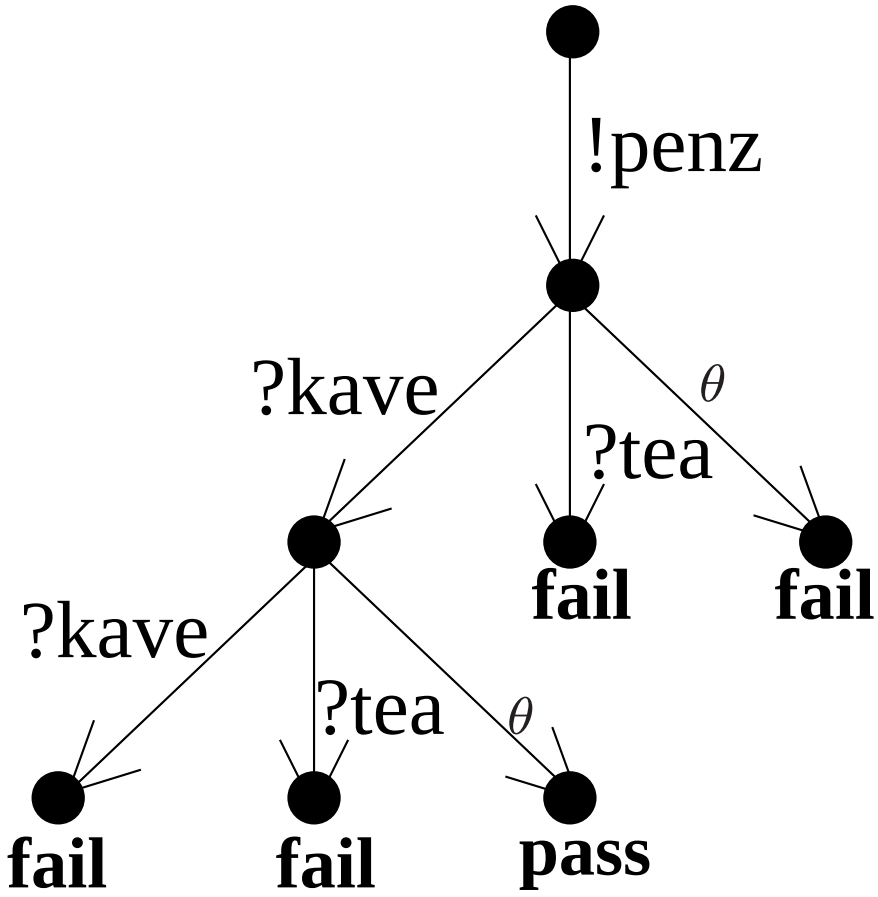


Teszt esetek 1

- A teszt esetek teszt állapotátmeneti rendszerrel (TTS) modellezhetők, ami egy speciális LTS
  - Gyökérrel rendelkező, véges, címkézett, determinisztikus fa
  - A címkehalmaz tartalmazza a teszter oldali timeout-ot: $L \cup \{\theta\}$
  - A végállapotok címkével rendelkeznek: $\{\mathbf{pass}, \mathbf{fail}\}$
  - Minden egyes állapotból, ami nem a végállapot vagy van egy input vagy megjelenik az összes output és a θ

Teszt esetek 2

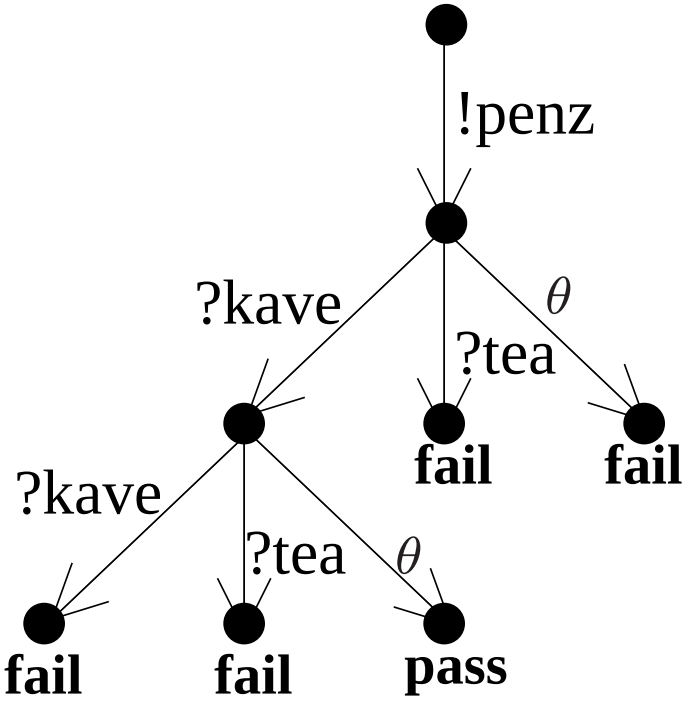
teszt eset t:	
!penz; start θ_1	
?kave; start θ_2	
$?\theta_2$	**pass**
?kave	**fail**
?tea	**fail**
?tea	**fail**
θ_1	**fail**

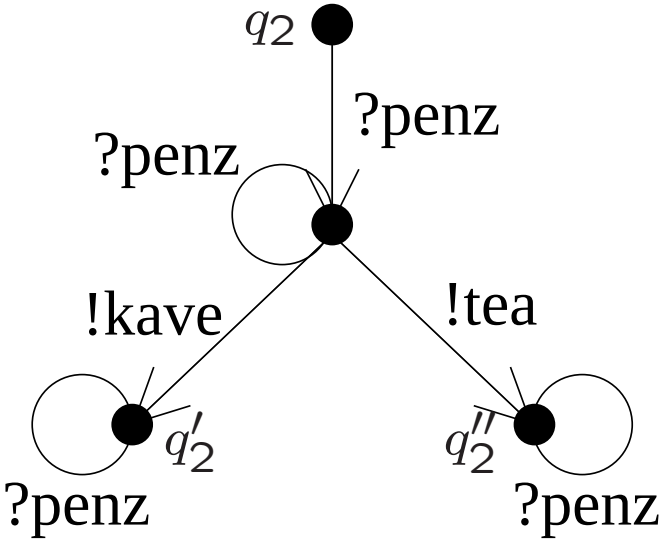


Teszt esetek végrehajtása 1

- A megfigyelhető nyomvonalak: $(L \cup \{\theta\})*$
- A megfigyelés: $obs : TTS \times IOTS \rightarrow \mathcal{P}((L \cup \{\theta\})*)$
- A megfigyelés a teszter és az implementáció teljes szinkronizációjából adódik: $obs(t, i) = \{\sigma \in (L \cup \{\theta\}) * |t||i \stackrel{\sigma}{\Rightarrow} \mathbf{pass}||i'$ vagy $t||i \stackrel{\sigma}{\Rightarrow} \mathbf{fail}||i'\}$
- Az ítélet: $v_t : \mathcal{P}((L \cup \{\theta\})*) \rightarrow \{\mathbf{pass}, \mathbf{fail}\}$
- ahol $v_t(O) = \begin{cases} \mathbf{pass} & \forall \sigma \in O : t \stackrel{\sigma}{\Rightarrow} \mathbf{pass} \\ \mathbf{fail} & \text{egyébként} \end{cases}$

Teszt esetek végrehajtása 3





$t||q_2 \xRightarrow{penz\ kave\ \theta} \mathbf{pass}||q_2'$

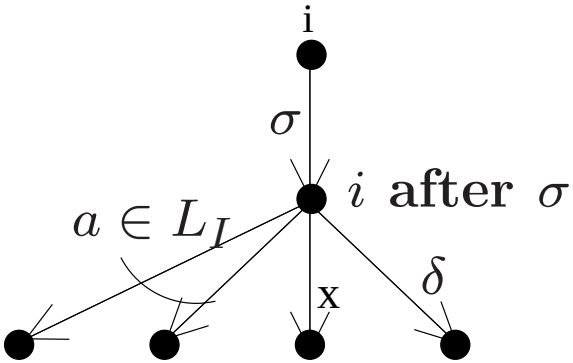
$t||q_2 \xRightarrow{penz\ tea} \mathbf{fail}||q_2''$

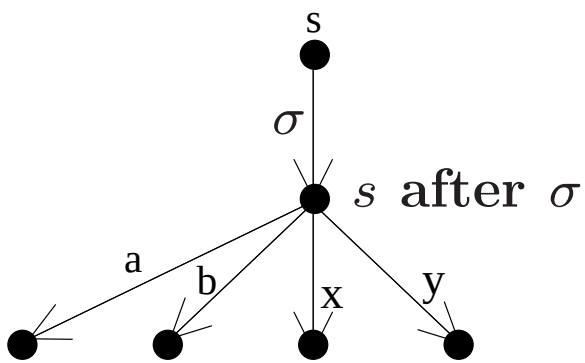
Tehát q_2 **fails** t

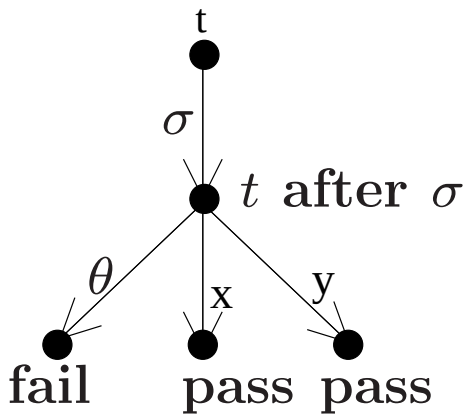
Teszt esetek generálása 1

- i **ioco** $s \iff \forall \sigma \in Straces(s) : out(i \textbf{ after } \sigma) \subseteq out(s \textbf{ after } \sigma)$

- $out(i \textbf{ after } \sigma) = \{\delta, x\} \not\subseteq out(s \textbf{ after } \sigma) = \{x, y\}$







Teszt esetek generálása 2

- Rekurzív nemdeterminisztikus algoritmus, amely az S állathalmaz alapján létrehozza a $t(S)$ teszt esetet, $S_0 = \{s_0\}$ kezdőállapottal:

  1. Legyen a teszt eset vége **pass**!

  2. Ajánljunk fel egy $!a$ eseményt az implementáció számára, ahol $?a \in L_I$, ha S **after** $?a = S' \neq \emptyset$!

  3. Vizsgáljuk meg az implementáció kimenetét! Ha $!x \notin out(S)$, $!x \in L_O \cup \{\delta\}$ vége az ágnak, és $v_t =$ **fail**. Ha $!x \in out(S)$, akkor hajtsuk végre az algoritmust az $S_x = S$ **after** $!x$ állapothalmazra. A δ rendszer oldali timeoutot a teszt esetben θ teszter oldali timeouttal helyettesítjük.

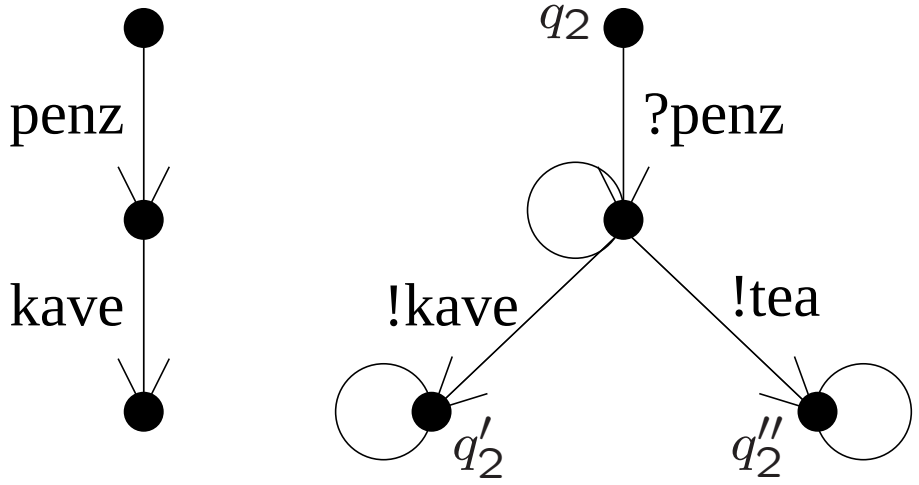
Teszt esetek generálása 3

- Legyen $S = \{s_0\}$, ahol s_0 a kezdeti állapot.

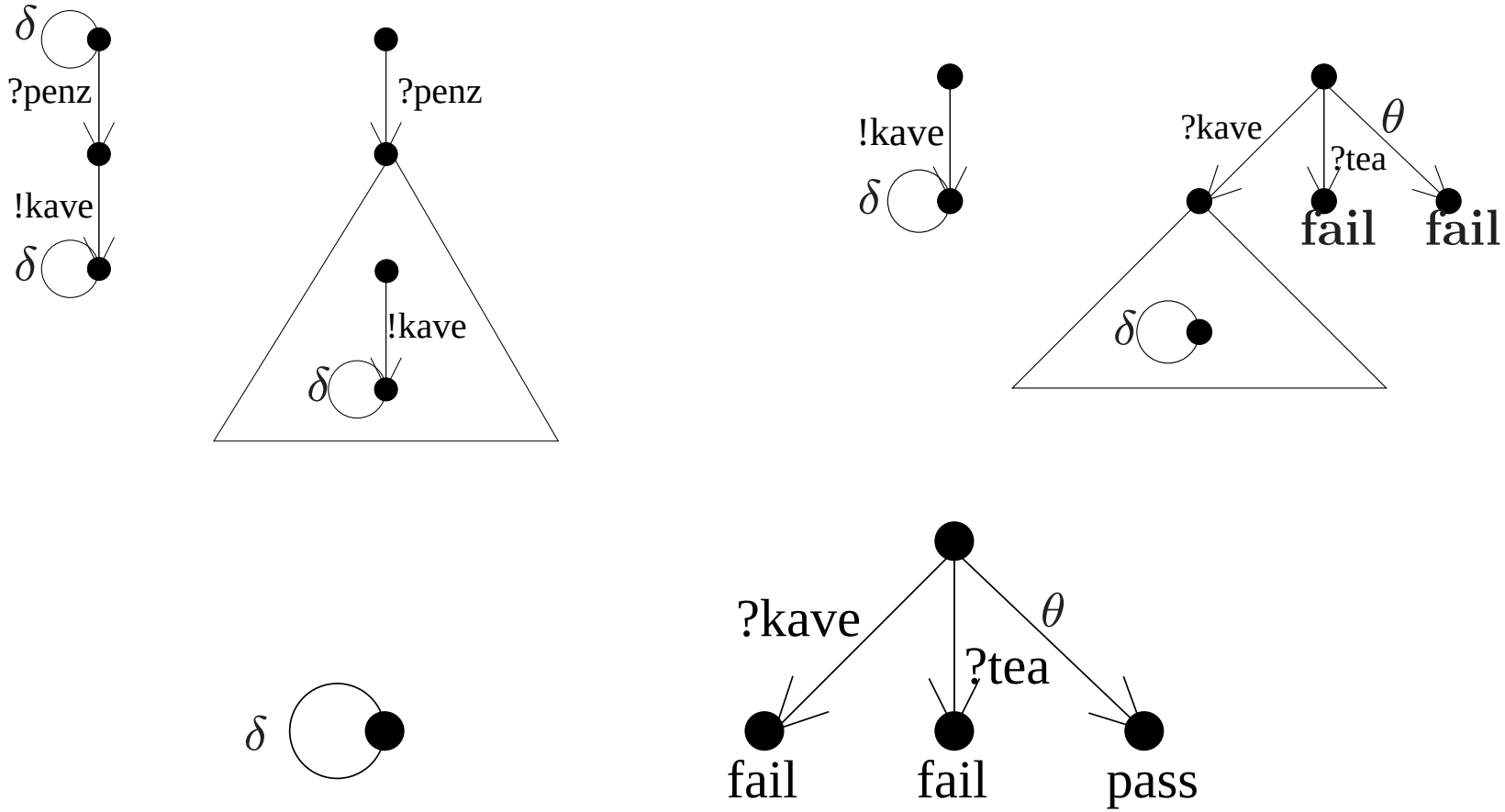
  1. $t(S)$ kimerítő: i **fails** $t(S) \Rightarrow i$ **ioco** s

  2. Az összes t(S) együtt kimerítő: i **ioco** $s \Rightarrow \exists t(S) : i$ **fails** $t(S)$

- Példa: a specifikáció és az implementáció

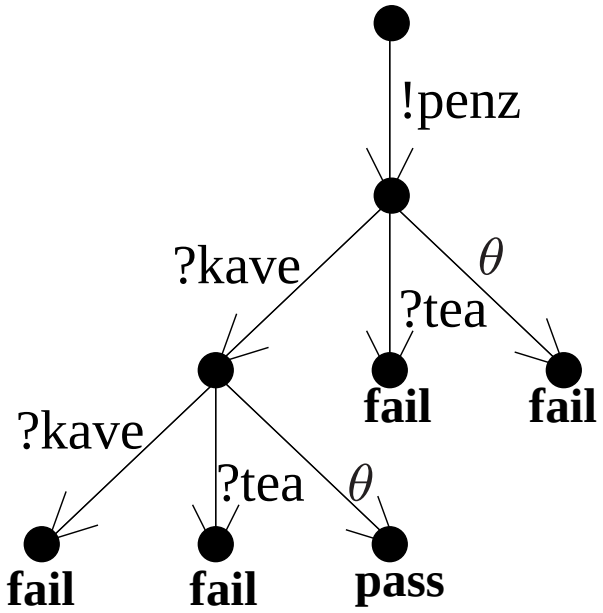


Teszt esetek generálása 4



Teszt esetek generálása 5

- Összerakva egy teszt esetté:

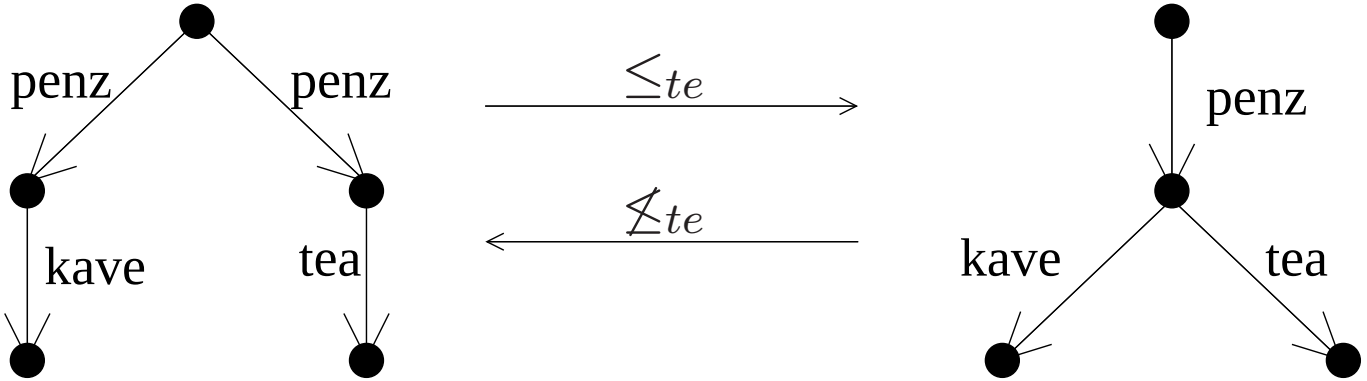


- Végrehajtva: q_2 $\mathbf{io}\not\mathbf{co}$ s

Teszterek

- Tekintsük a következő definíciót
- i **conf** $s \iff \forall u \in U : obs(u, i) \cap traces(s) \subseteq obs(u, s)$
- Ez a reláció egyetlen $T(s)$ tesztert definiál, amelyre i **conf** $s \iff$ i **passes** $T(s)$
- $T(s) \sim s$, tehát a $T(s)$ a specifikáció mintegy mása

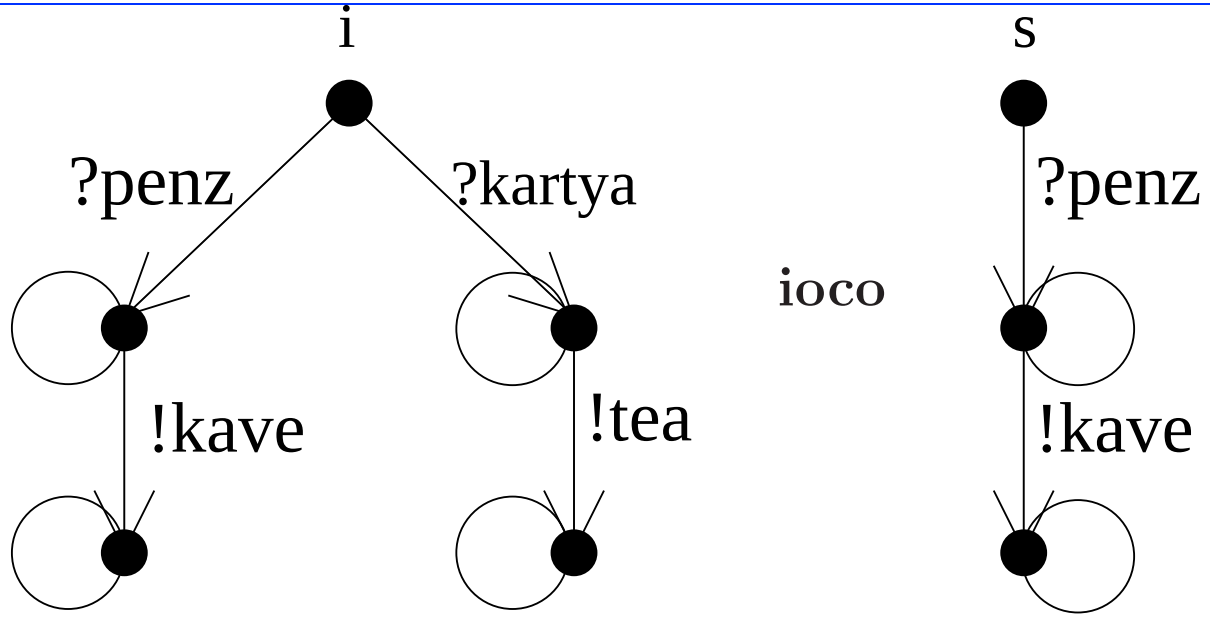
Példa a tesztelési előrendezésre



- **after** ε **refuses** $\{kave, tea\}$
- **after** $penz$ **refuses** $\{penz, kave\}$
- **after** $penz$ **refuses** $\{penz, tea\}$
- **after** $penz.tea$ **refuses** L
- **after** $penz.kave$ **refuses** L

- **after** ε **refuses** $\{kave, tea\}$
- **after** $penz$ **refuses** $\{penz\}$
- **after** $penz.tea$ **refuses** L
- **after** $penz.kave$ **refuses** L

Példa az ioco-ra



- $out(i \textbf{ after } ?penz) = \{!kave\}$
- $out(i \textbf{ after } ?kartya) = \{!tea\}$
- $out(s \textbf{ after } ?penz) = \{!kave\}$
- $out(s \textbf{ after } ?kartya) = \emptyset$

- A $?penz$ inputra fennáll a részhalmaz reláció, de a $?kartya$ inputra nem
- Mégis i **ioco** s, mert $?kartya \notin Straces(s)$.