



Modellalapú szoftverfejlesztés

IX. előadás

Gráfmentaillesztés, Gráftranszformáció

Semeráth Oszkár

Gráfmintaillesztés, Gráftranszformáció

Alapfogalmak

Gráfmintaillesztés

Modelltranszformációk

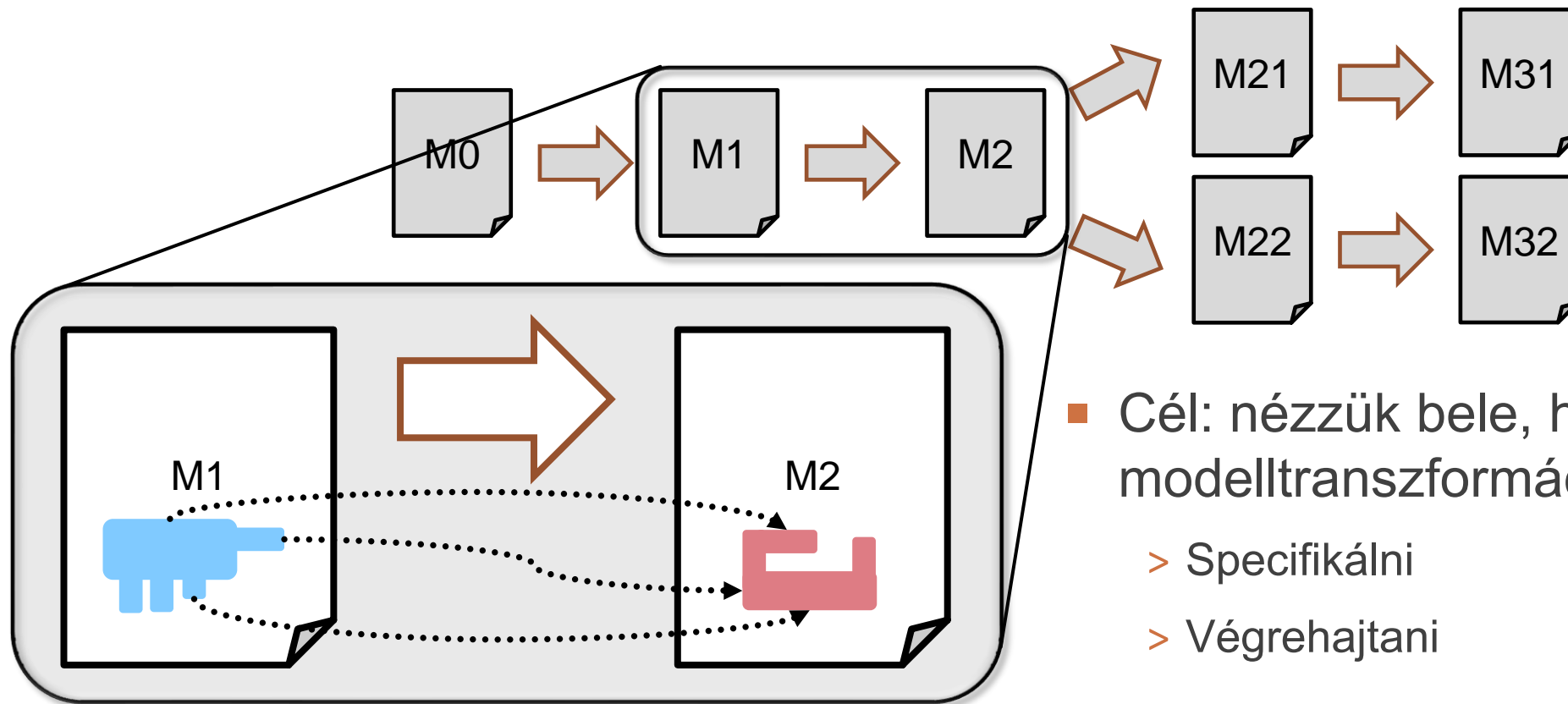
Inkrementális transzformációk

Tervezésítér bejárás



Motiváció: Modellek transzformációja

- **Modellalapú fejlesztés:** Modellek az elsődleges dokumentumok
- Modelleket fejlesztünk, automatizáljuk a modellfeldolgozást

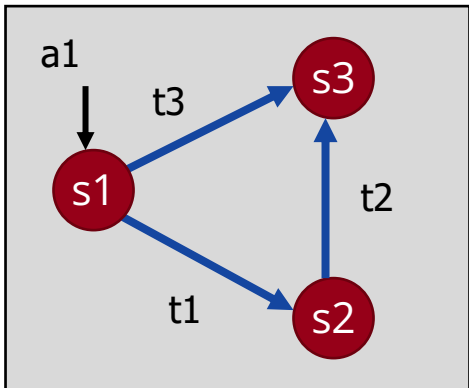


- Cél: nézzük bele, hogyan lehet modelltranszformációkat
 - > Specifikálni
 - > Végrehajtani

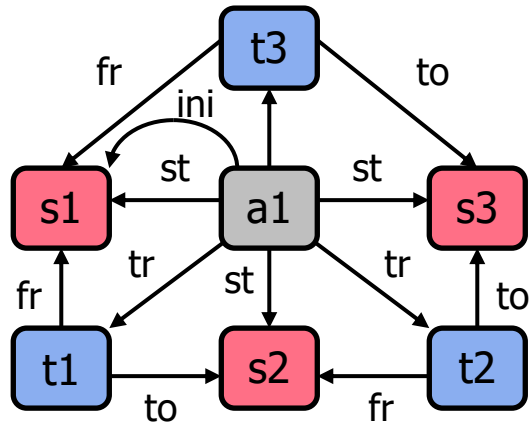
Absztrakt szintaxis

- **Hogyan módosítsuk a modelleket?**
- **Ötlet:** módosítsuk modellek reprezentációját közvetlenül → **Absztrakt szintaxis**

Konkrét



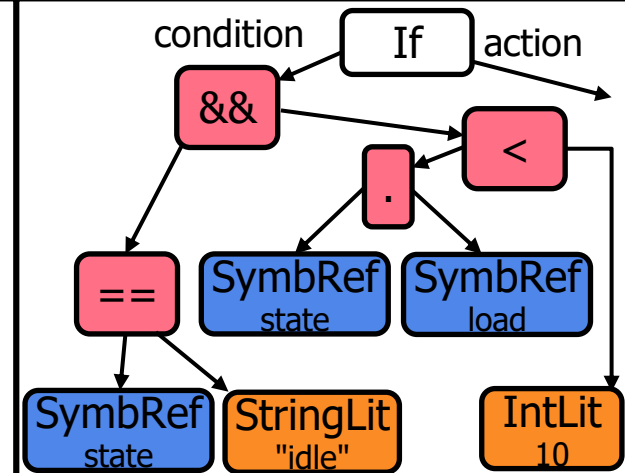
Absztrakt



Konkrét

```
if (  
    state ==  
    "idle" &&  
    this.load < 10)  
    ...
```

Absztrakt

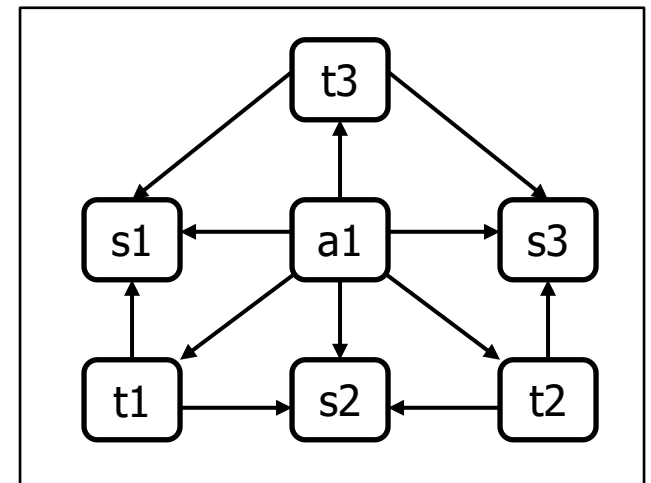


- **Feladat:** gráfok módosítására módszer!

Hogyan írjuk le a modelleket gráffal?

Haladó matek a diasorban,
Szemléltetés demóval

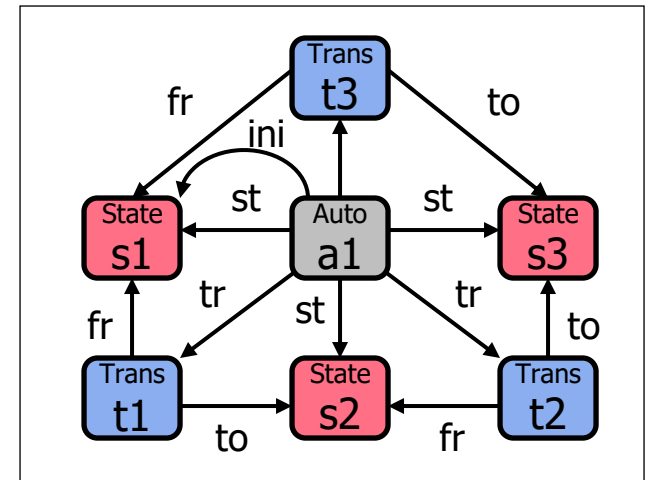
- Irányított gráf: $G = \langle V_G, E_G \rangle$, ahol:
 - > V_G : **csúcsok** halmaza (a G gráfban)
 - > $E_G: V_G \times V_G \rightarrow \{true, false\}$: **élek** a csúcsok között (a G gráfban)
 - > Hurokélek \checkmark , Párhuzamos élek \checkmark , Párhuzamos éle egy irányba \boxtimes
- Példa: állapotgép gráf S
 - > $V_S = \{a_1, s_1, s_2, s_3, t_1, t_2, t_3\}$
 - > $E_S(a_1, s_1) = true$, de
 - > $E_S(s_1, a_1) = false$
- Hogyan adjunk hozzá címkéket?



Címkézett gráf: címkekészlet

Haladó matek a diasorban,
Szemléltetés demóval

- Szótár: $\langle \Sigma, \alpha \rangle$
- Σ : Címkék halmaza
 $\Sigma = \{Automaton, \textcolor{red}{State}, \textcolor{blue}{Transition}, states, transitions, from, to, init\}$
- Aritás $\alpha: \Sigma \rightarrow \mathbb{N}$ meghatározza a címke szerepét
 - > **Csúcs címke** $\alpha: Automaton, \textcolor{red}{State}, \textcolor{blue}{Transition} \mapsto 1$
Típusok, osztályok leírására
 - > **Élcímke**: $\alpha: states, transitions, from, to, init \mapsto 2$
Attribútumok, referenciák leírására
 - > Lehet több is (3+), most nekünk nem fontos



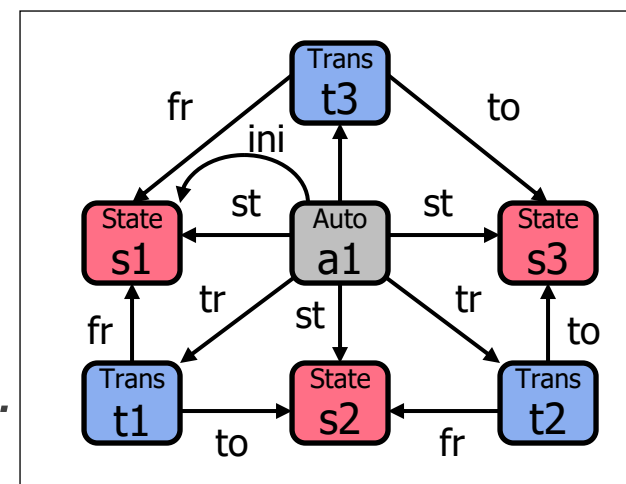
Címkekészlettel címkézett gráf

Haladó matek a diasorban,
Szemléltetés demóval

- Adott $\langle \Sigma, \alpha \rangle$ szótárhoz
- **Címkézett gráf (modell):** $M = \langle O_M, I_M \rangle$, ahol:
- O_M : **csúcsok** vagy **objektumok** halmaza (az M modellben)
- $I_M(s): O_M^{\alpha(s)} \rightarrow \{true, false\}$: **interpretációs függvény**
minden $s \in \Sigma$ szimbólumhoz

I_M meghatározza, hogy:

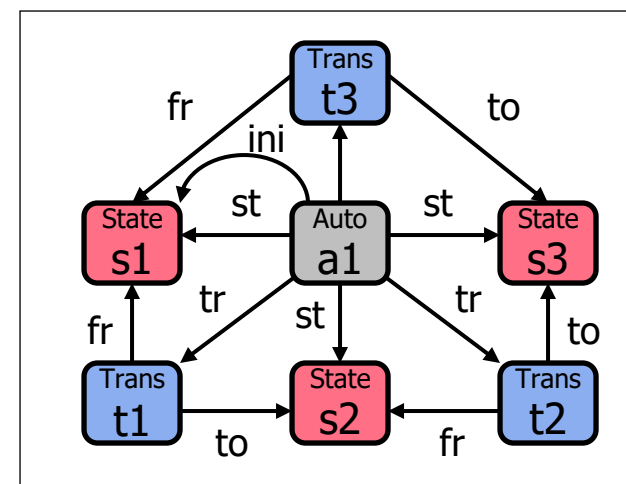
- *mely csúcsokon van egy csúcscímke, és*
- *hogy mely csomópont pár között vezet adott címkéjű él.*



Példa címkézett gráf (modell)

Haladó matek a diasorban,
Szemléltetés demóval

- Példák az S állapotgépben
- $O_S = \{a_1, s_1, s_2, s_3, t_1, t_2, t_3\}$
- Példa típusok: $I_S(\textit{State})(s_1) = \textit{true}$, vagy csak egyszerűen:
 - > $\textit{State}(s_1) = \textit{true}$, $\textit{Transition}(t_1) = \textit{true}$,
 - > $\textit{State}(t_1) = \textit{false}$, $\textit{Transition}(s_1) = \textit{false}$
- Példa élekre:
 - > $\textit{from}(t_1, s_1) = \textit{true}$, $\textit{to}(t_1, s_2) = \textit{true}$
 - > $\textit{from}(s_1, s_1) = \textit{false}$



Gráfmintaillesztés, Gráftranszformáció

Alapfogalmak

Gráfmintaillesztés

Modelltranszformációk

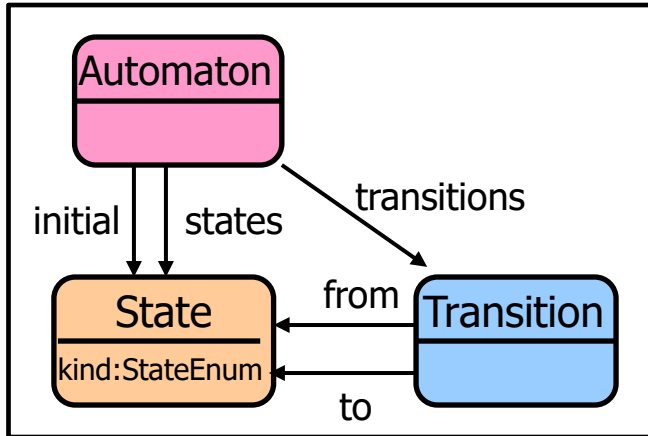
Inkrementális transzformációk

Tervezésítér bejárás

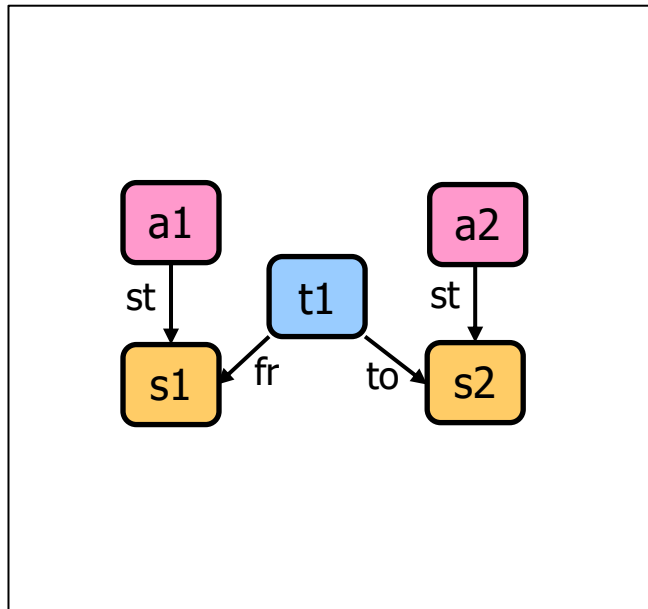


Egy egyszerű példa

Metamodell



Szabálysértés példa



■ Jólformáltsági kényszer:

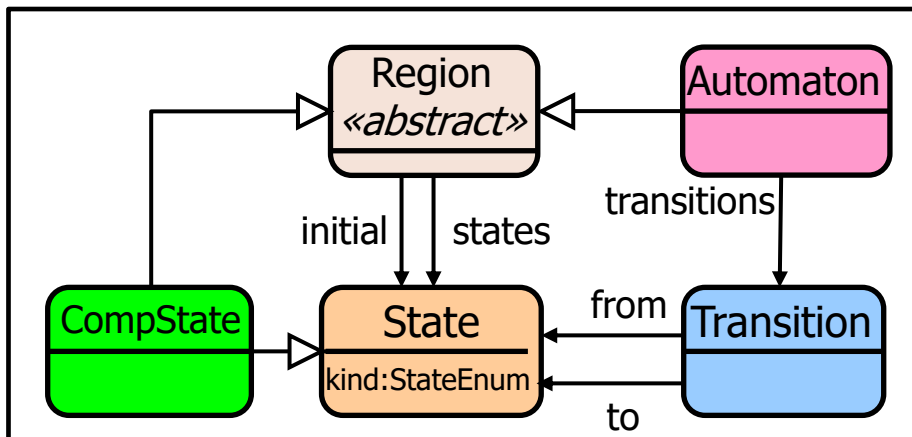
- > Tranzíció forrás- és célállapotának ugyanabban az állapotgépben kell lennie

■ Cél: szabálysértések megtalálása...

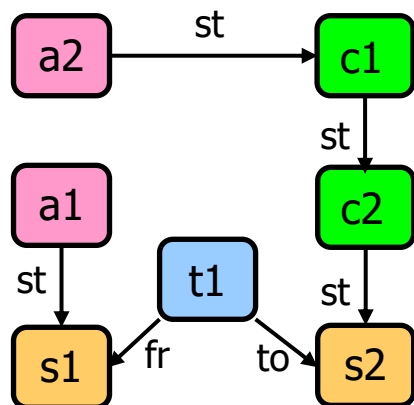
- > A szabályszegés egy *Tranzíció*, aminek „*from*” éle egy *s1 State*-re mutat, és „*to*” éle egy *s2 State*-re mutat, ahol *s1 állapotgépe* nem *s2 állapotgépe*

Egy összetettebb példa

Metamodell



Szabálysértés példa



■ Jólformáltsági kényszer:

- > Tranzíció forrás- és célállapotának ugyanabban az állapotgépben kell lennie

■ Cél: szabálysértések megtalálása...

- > A szabályszegés egy *Tranzíció*, aminek „*from*” éle egy *s1 State*-re mutat, és „*to*” éle egy *s2 State*-re mutat, ahol *s1* állapotgépe nem *s2* állapotgépe

Programozott bejárás vs. lekérdezések

- **Cél:** kényszer megsértéseinek megtalálása a modellben

A modell bejárása általános célú nyelven

```
for (Automaton automaton : automations) {  
    for (Transition transition : automaton.getTransitions()) {  
        State sourceState = transition.from;  
        // melyik automaton definiálja ezt az állapotot?  
        Automaton sourceAutomaton = null;  
        for (Automaton candidate : automations) {  
            if (candidate.getStates().contains(sourceState)) {  
                sourceAutomaton = candidate;  
                break;  
            }  
        }  
        // ... ugyanezt a targetState esetében, majd  
        if (sourceAutomaton != targetAutomaton)  
            // szabálysértés jelentése  
    }  
}
```

„egyszerű
példa”

Programozott bejárás vs. lekérdezések

- Cél: kényszer megsértéseinek megtalálása a modellben
 - > A modell bejárása általános célú nyelven
 - > Használjunk egy **lekérdezési DSL-t**
 - Tömörebb
 - A lekérdezés **deklaratív** funkcionális specifikációja
 - Szabadon értelmezhető a **lekérdezőmotor** (query engine) által (pl. optimalizálás)
 - Platformfüggetlen lehet
- A validálás csak egy felhasználási módja a **modell-lekérdezéseknek**
 - > Származtatott tulajdonságok
 - > M2M/M2T transzformáció, Szimuláció
 - > ...

Lekérdezési nyelvi stílusok

■ SQL-szerű (relációs algebra)

- > Példa: EMF Query
- > 😊 Jó az attribútum korlátozásokhoz
- > ☹ Nem túl tömör a kapcsolatokra (sok join)

■ Funkcionális stílus

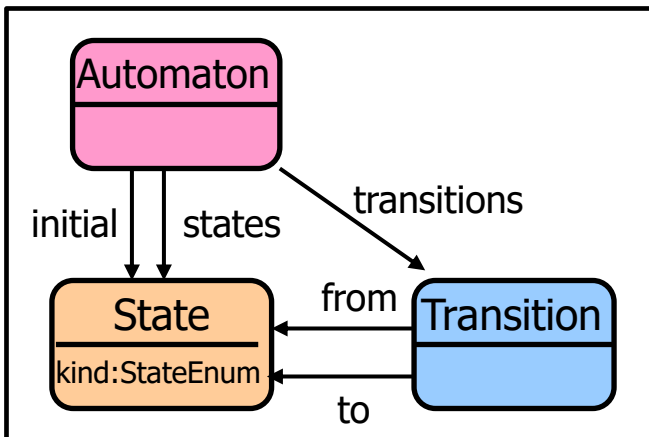
- > Példa: OCL
- > Valamelyest deklaratív

```
context Transition inv:  
  Automaton.allInstances()->forAll(a |  
    a.states->includes(self.from) =  
    a.states->includes(self.to)  
  );
```

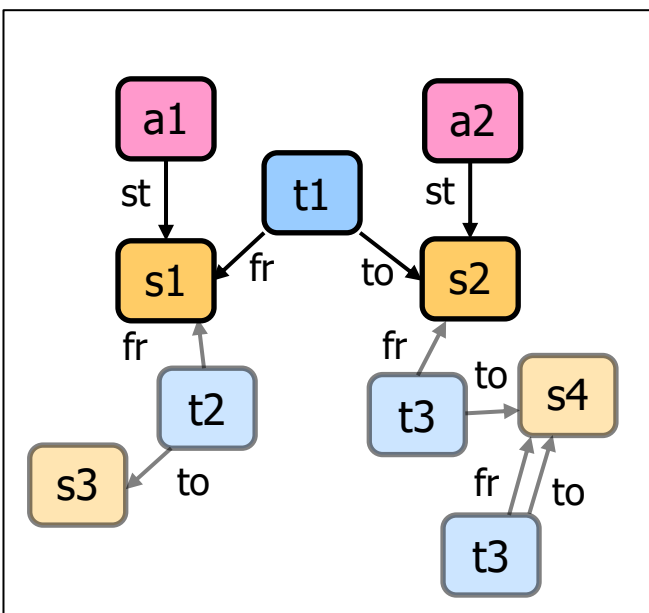
■ Logikai stílus

- > Domain relációs kalkulus / gráfminták / Datalog
- > Még inkább deklaratív

Metamodell

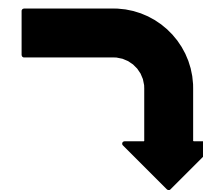


Szabálysértés példa



A szabályszegés egy Tranzíció, aminek „from” éle egy s1 State-re mutat, és „to” éle egy s2 State-re mutat, ahol s1 állapotgépe nem s2 állapotgépe

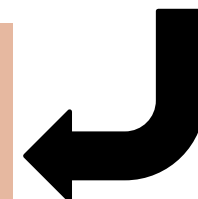
Formális logikával
(Domain
Relációs
Kalkulus)



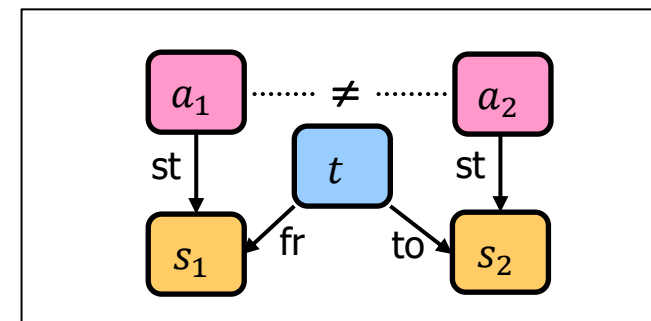
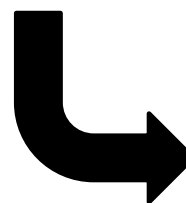
$$\{t \mid \exists s_1, s_2, a_1, a_2: Transition(t) \wedge from(t, s_1) \wedge to(t, s_2) \wedge states(a_1, s_1) \wedge states(a_2, s_2) \wedge a_1 \neq a_2\}$$

`violates(t) <->`
`Transition(t), from(t, s1), to(t, s2),`
`states(a1, s1), states(a2, s2), a1 != a2`

Datalog-szerű
lekérdezési
nyelvek



Lekérdezőmotor



Minta

Mi az a modell-lekérdezés?

- Egy programozó számára:
kód, amely a modell részeinek keresésére szolgál
- A tudós / mérnök számára:
 - > **Lekérdezés** = teljesítendő kényszerek halmaza ($Q(t)$)

$$Q(t) := \exists s_1, s_2, a_1, a_2: \mathbf{Trans}(t) \wedge \mathbf{fr}(t, s_1) \wedge \mathbf{to}(t, s_2) \wedge \mathbf{st}(a_1, s_1) \wedge \mathbf{st}(a_2, s_2) \wedge a_1 \neq a_2$$

- > **Modell** = a lekérdezés kiértékelésének célpontja (\mathbf{V})
- > **Lekötés** = kényszerváltozókat a modell elemeihez köti

$$Z_1: t \mapsto \mathbf{t1}, Z_2: t \mapsto \mathbf{t2}, \dots$$

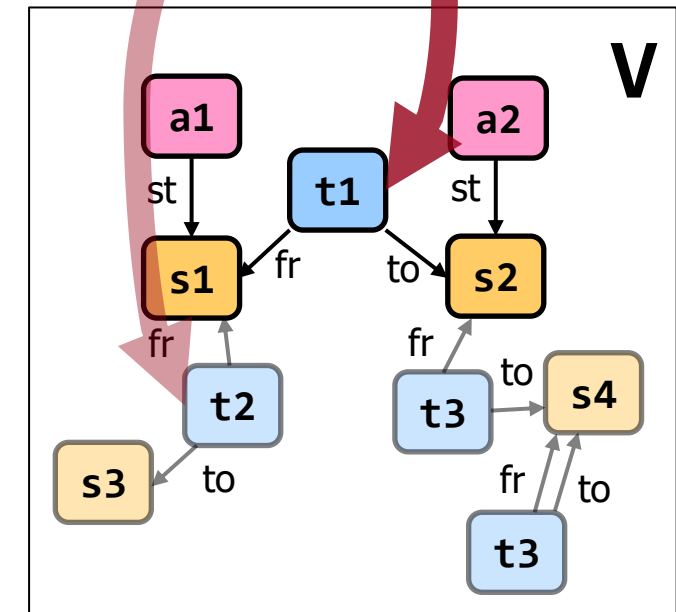
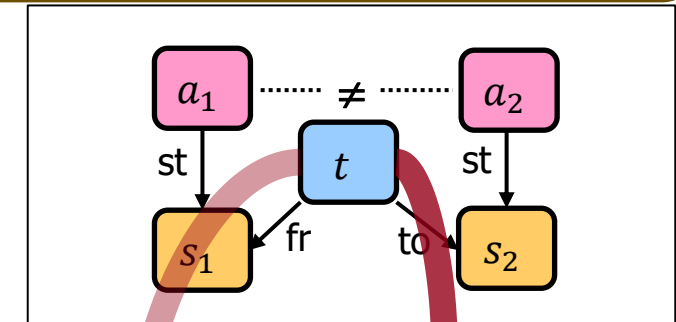
- > **Eredmény** = a kiértékelés igazságértéke (igaz/hamis)

$$\llbracket Q(t) \rrbracket_{Z_1}^{\mathbf{V}} = \mathbf{true}$$

← modell
← Eredmény
← lekötés

lekérdezés

Haladó matek a diasorban,
Szemléltetés demóval



Lekérdezés

Szabálysértés példa

Lekérdezés kiértékelése

Haladó matek a diasorban,
Szemléltetés demóval

- **Szótár:** $\langle \Sigma, \alpha \rangle$, Σ : Címkék halmaza, α : típus/él
- **Modell** $M = \langle Obj_M, I_M \rangle$, Obj_M =objektumok: I_M = élek + címkék
- Egy Q **lekérdezés** egy logikai kifejezés, amely egy M modellel és egy Z lekötéssel van definiálva.

- Példa a $\llbracket Q \rrbracket_Z^M$ definíciójára

- A lekérdezőmotor támogatja
 - > a modell-lekérdezések definiálását,
 - > lekérdezés-illesztést:

*M modellre és Q lekérdezésre,
visszaad minden Z lekötést,
ahol $\llbracket Q \rrbracket_Z^M = true$.*

\approx DB query

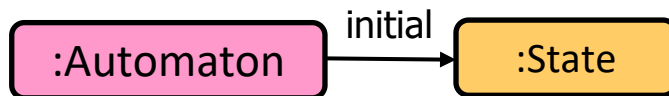
$$\begin{aligned}\llbracket \mathbf{C}(v) \rrbracket_Z^M &:= \mathcal{I}_M(\mathbf{C})(Z(v)) \\ \llbracket \mathbf{R}(v_1, v_2) \rrbracket_Z^M &:= \mathcal{I}_M(\mathbf{R})(Z(v_1), Z(v_2)) \\ \llbracket v_1 = v_2 \rrbracket_Z^M &:= Z(v_1) = Z(v_2) \\ \llbracket \varphi_1 \wedge \varphi_2 \rrbracket_Z^M &:= \llbracket \varphi_1 \rrbracket_Z^M \wedge \llbracket \varphi_2 \rrbracket_Z^M \\ \llbracket \varphi_1 \vee \varphi_2 \rrbracket_Z^M &:= \llbracket \varphi_1 \rrbracket_Z^M \vee \llbracket \varphi_2 \rrbracket_Z^M \\ \llbracket \neg \varphi \rrbracket_Z^M &:= \neg \llbracket \varphi \rrbracket_Z^M \\ \llbracket \forall v : \varphi \rrbracket_Z^M &:= \bigwedge_{x \in Obj_M} \llbracket \varphi \rrbracket_{Z, v \mapsto x}^M \\ \llbracket \exists v : \varphi \rrbracket_Z^M &:= \bigvee_{x \in Obj_M} \llbracket \varphi \rrbracket_{Z, v \mapsto x}^M\end{aligned}$$

Minta anatómiája

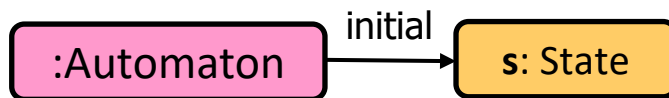
- A mintában szereplő típusok és referenciák egy metamodellben vannak definiálva
- Egy minta bevezet néhány csomópont változót

Pl. kettő csomópont:  

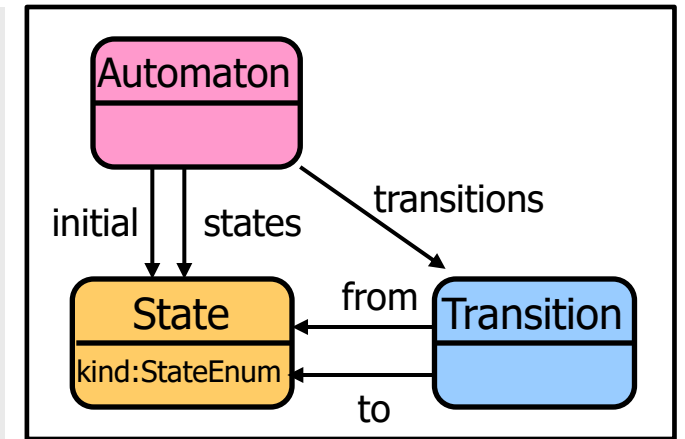
- És definiál köztük viszonyokat és állításokat



- Némely változót megnevezünk

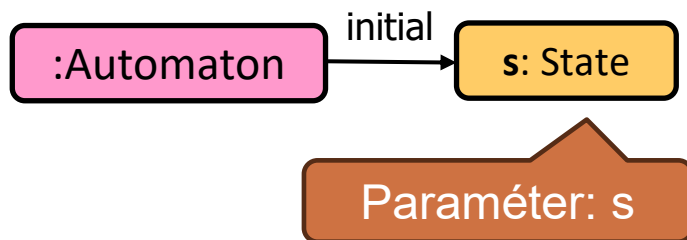


Metamodell



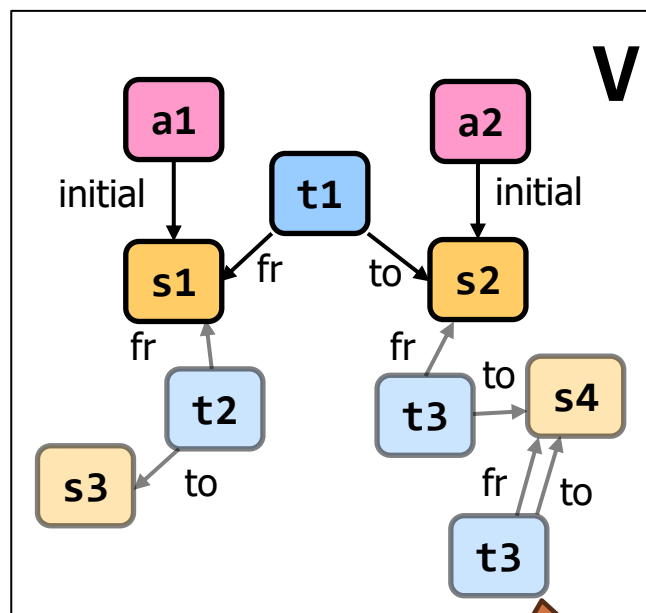
Minta illesztése

Minta



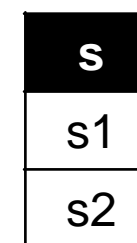
+

Példa modell



Modellelemek

Illeszkedés

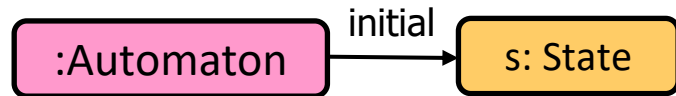


Paraméter: s

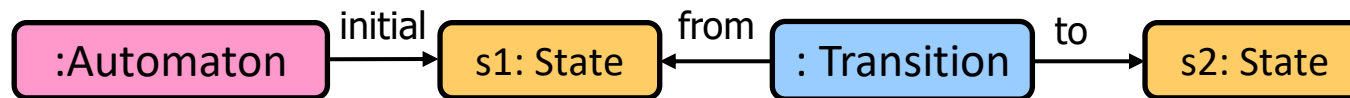
Modellelemek

Példák

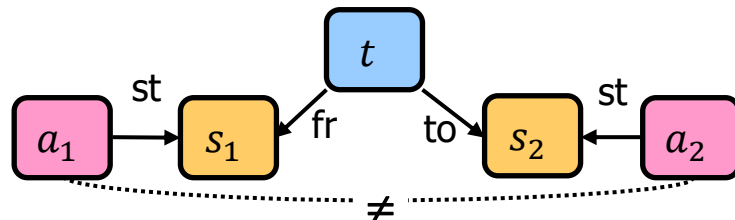
- **Egyszerű példa:** kezdőállapotok a modellben



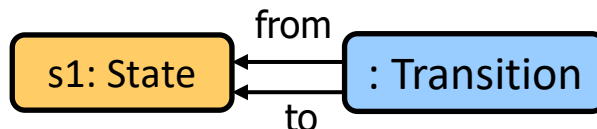
- **Láncolat (\wedge):** Második állapotok a modellben



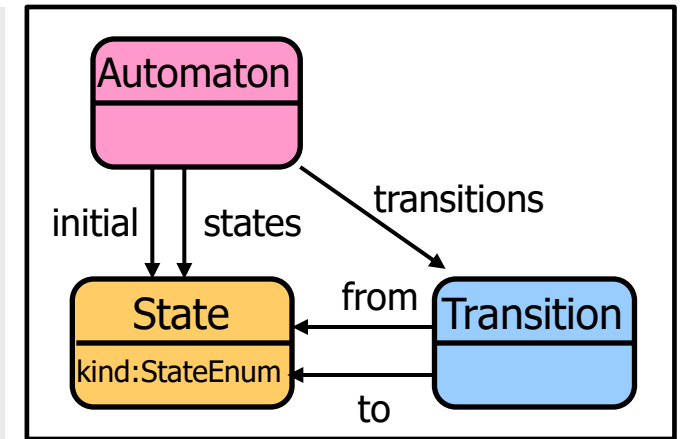
- \neq : Automatákon átívelő tranzíció



- $=$: Hurokél



Metamodell

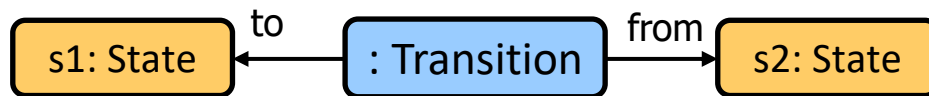


Példák 2

- (v): Két állapot össze van kötve

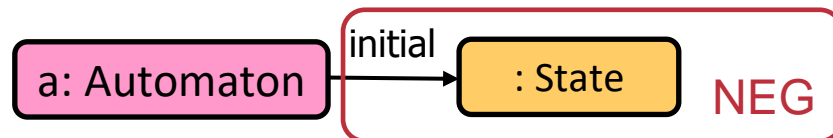


OR

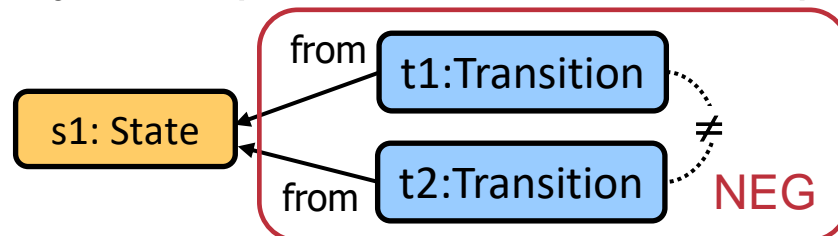


- (\neg , Negative Application Condition):

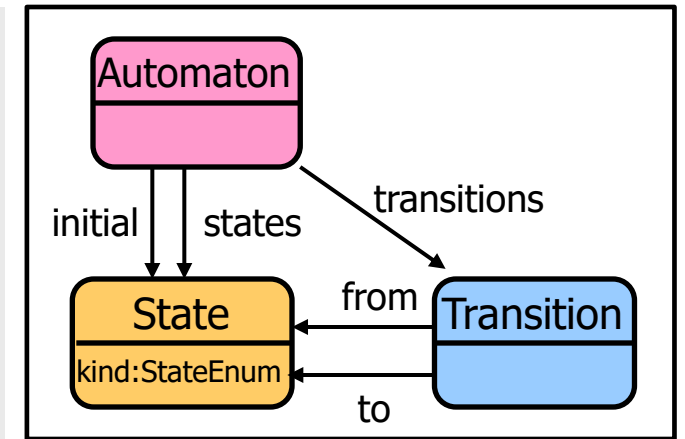
> kezdőállapot nélküli automata



> olyan állapot, aminek a kezdőállapotából nem megy ki két tranzakció (determinisztikus)



Metamodell



Gráfmintaillesztés, Gráftranszformáció

Alapfogalmak

Gráfmintaillesztés

Modelltranszformációk

Inkrementális transzformációk

Tervezésítér bejárás



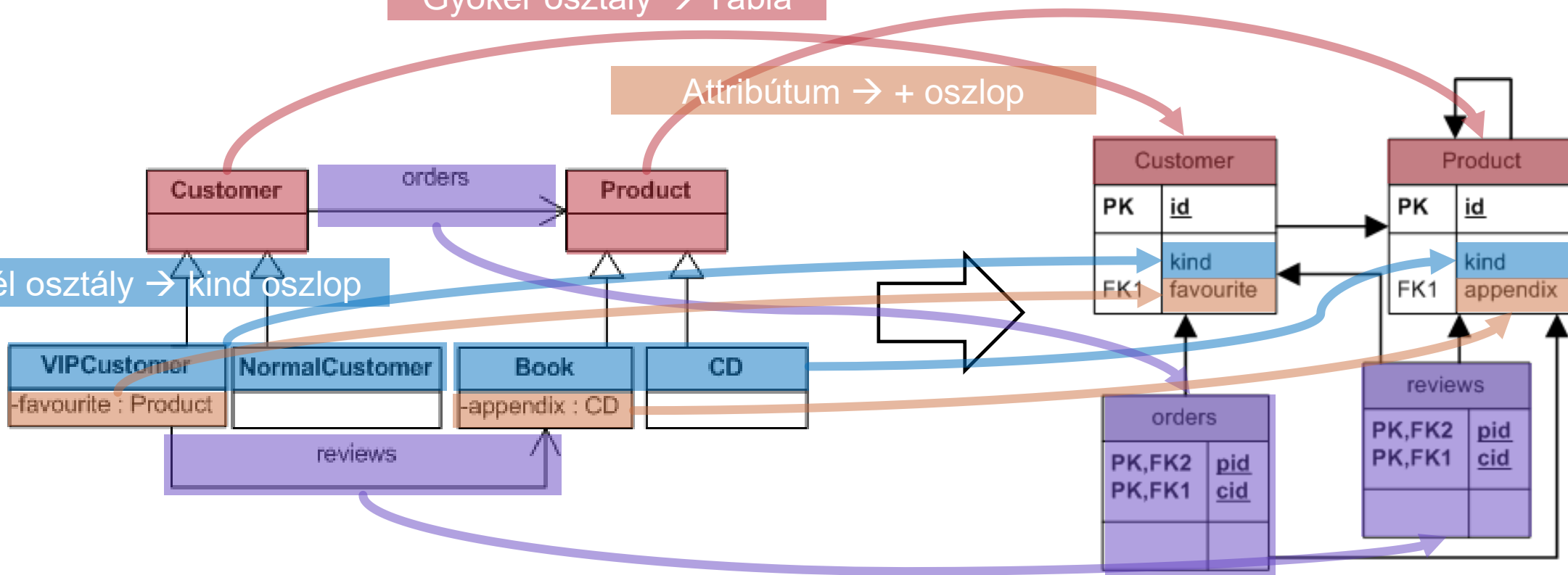
Példa Transzformáció

- Tipikus példa: képezzünk le egy osztálydiagramot adatbázis táblákra!

Gyökér osztály → Tábla

Attribútum → + oszlop

Levél osztály → kind oszlop



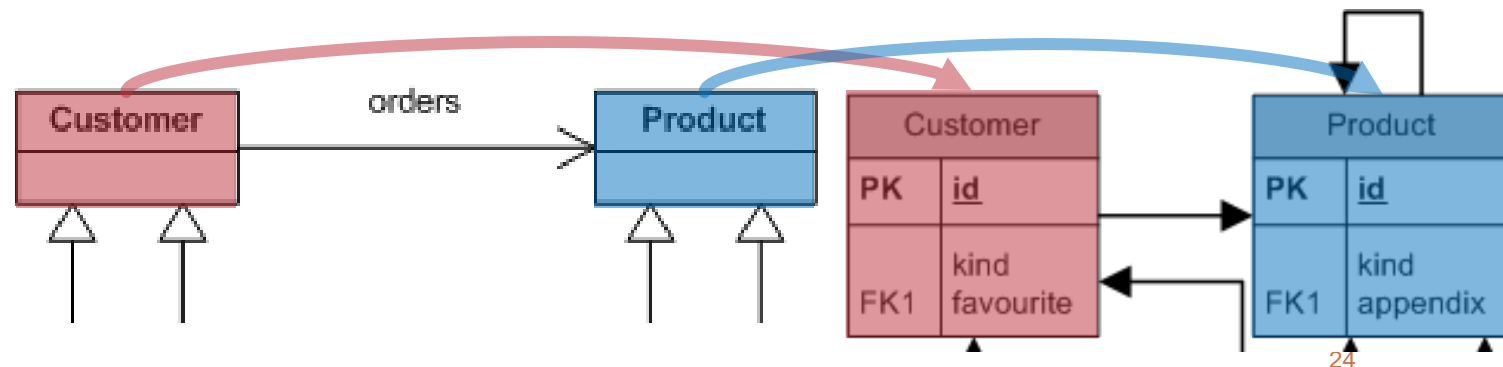
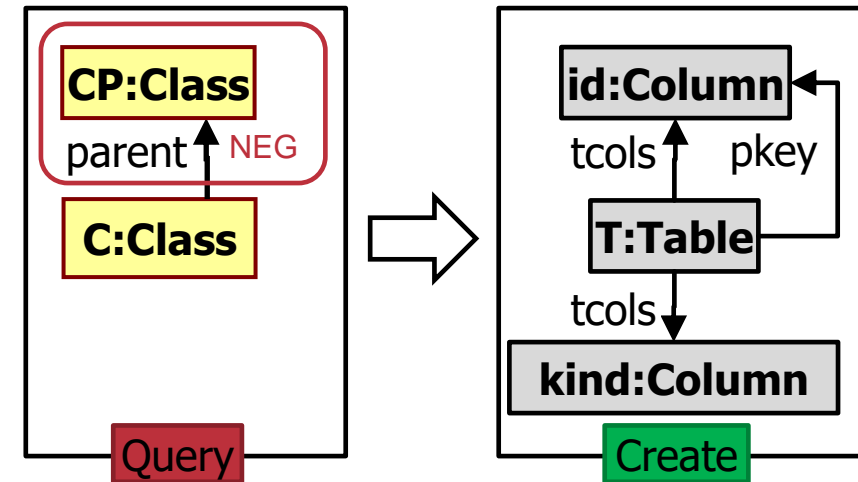
Referencia → Tábla + idegen kulcs

Példa Transzformáció

- Hogyan oldanánk a gyöker osztályokat reprezentáló táblák létrehozását?

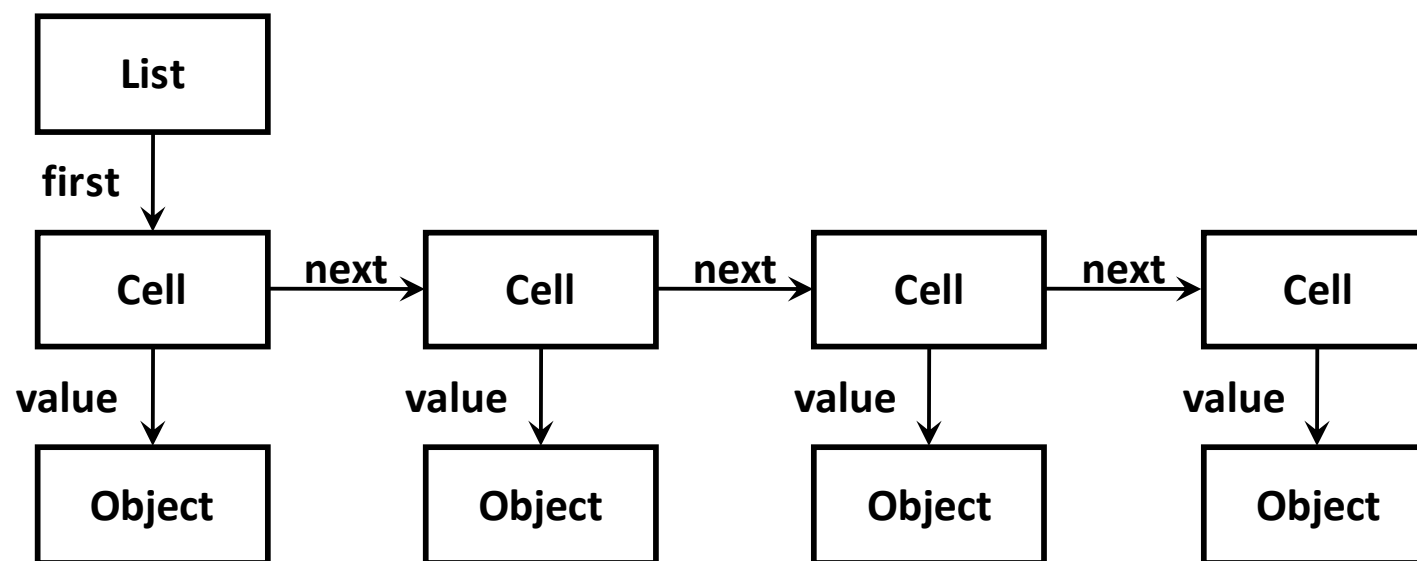
1. Lekérdeznénk a gyöker osztályokat (osztály, aminek nincs őse)
2. Létrehoznánk a táblákat, és velük a szükséges oszlopokat
3. Ismételnénk amíg tudjuk

- Cél: Hasonló szabályokkal megfogalmazni az egész transzformációt



Gráftranszformáció

- Modell = Címkézett gráf

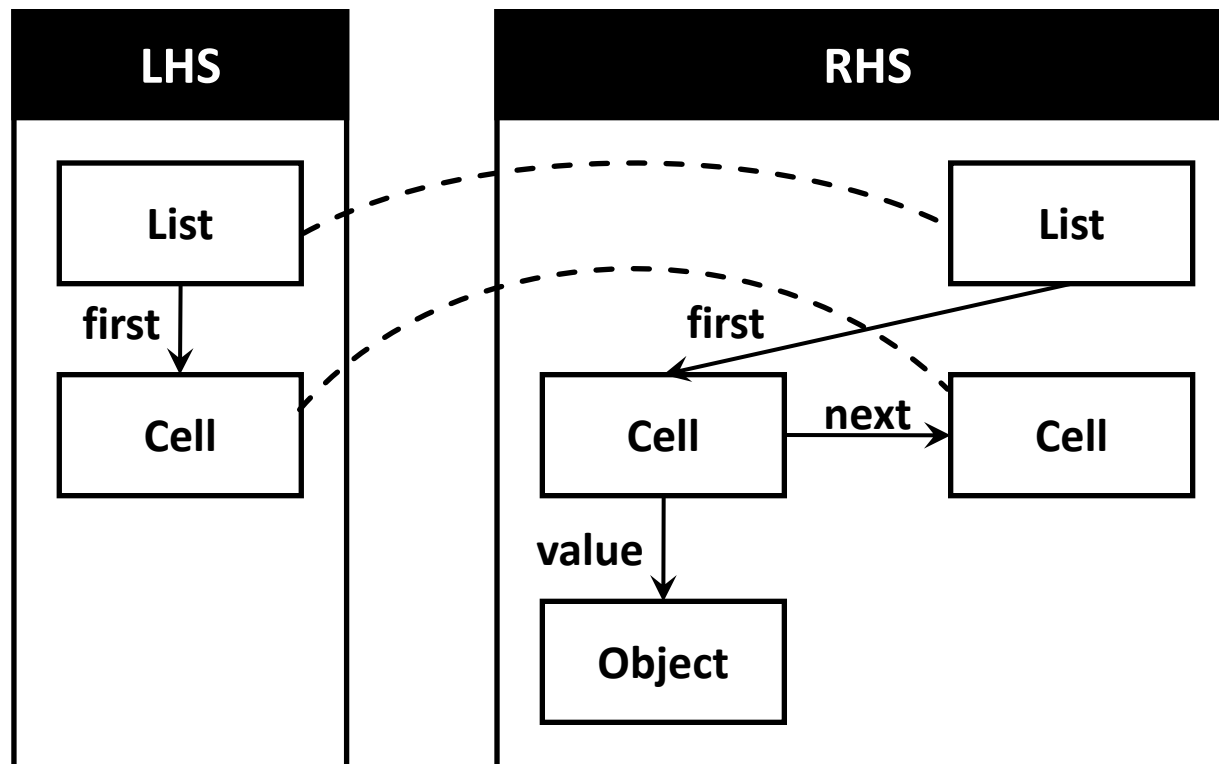


Gráftranszformációs szabály

- Gráf átírási szabály, két gráffal van megfogalmazva

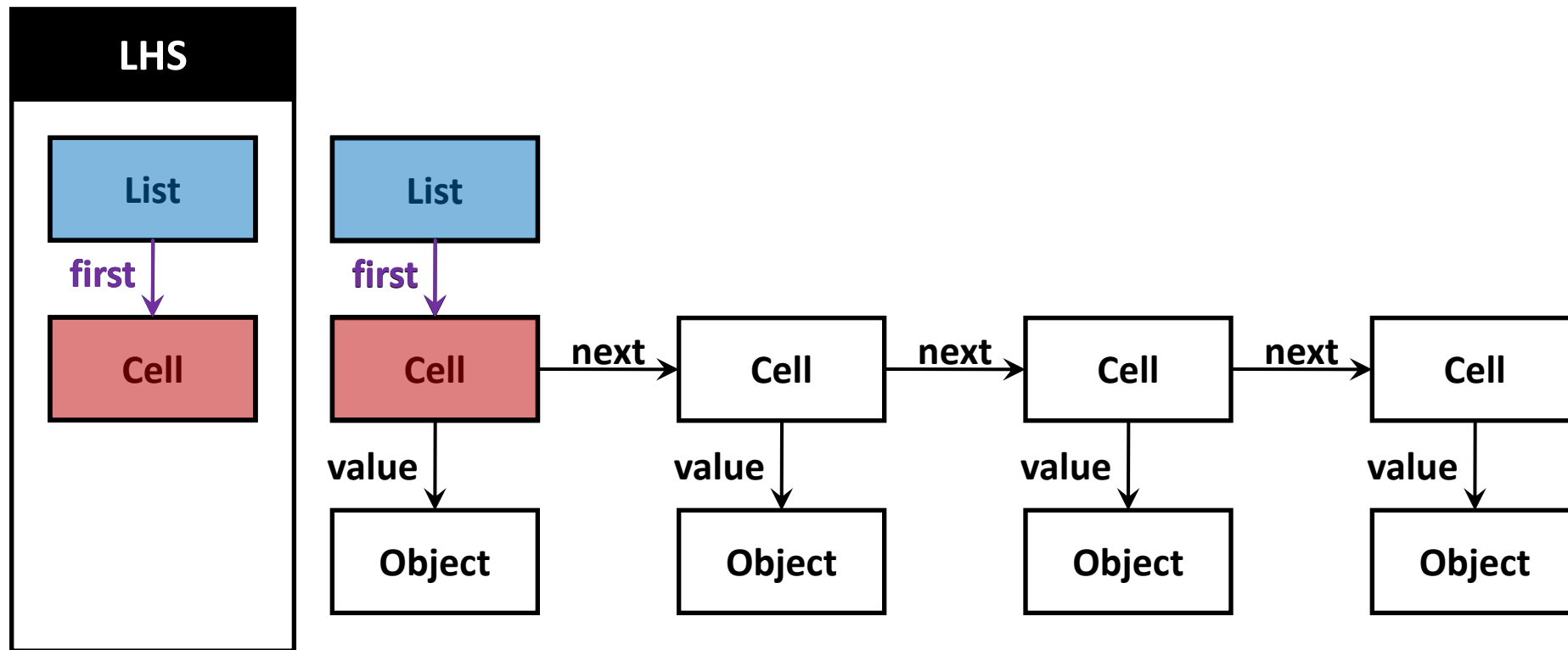
LHS = bal oldal,

RHS = jobb oldal



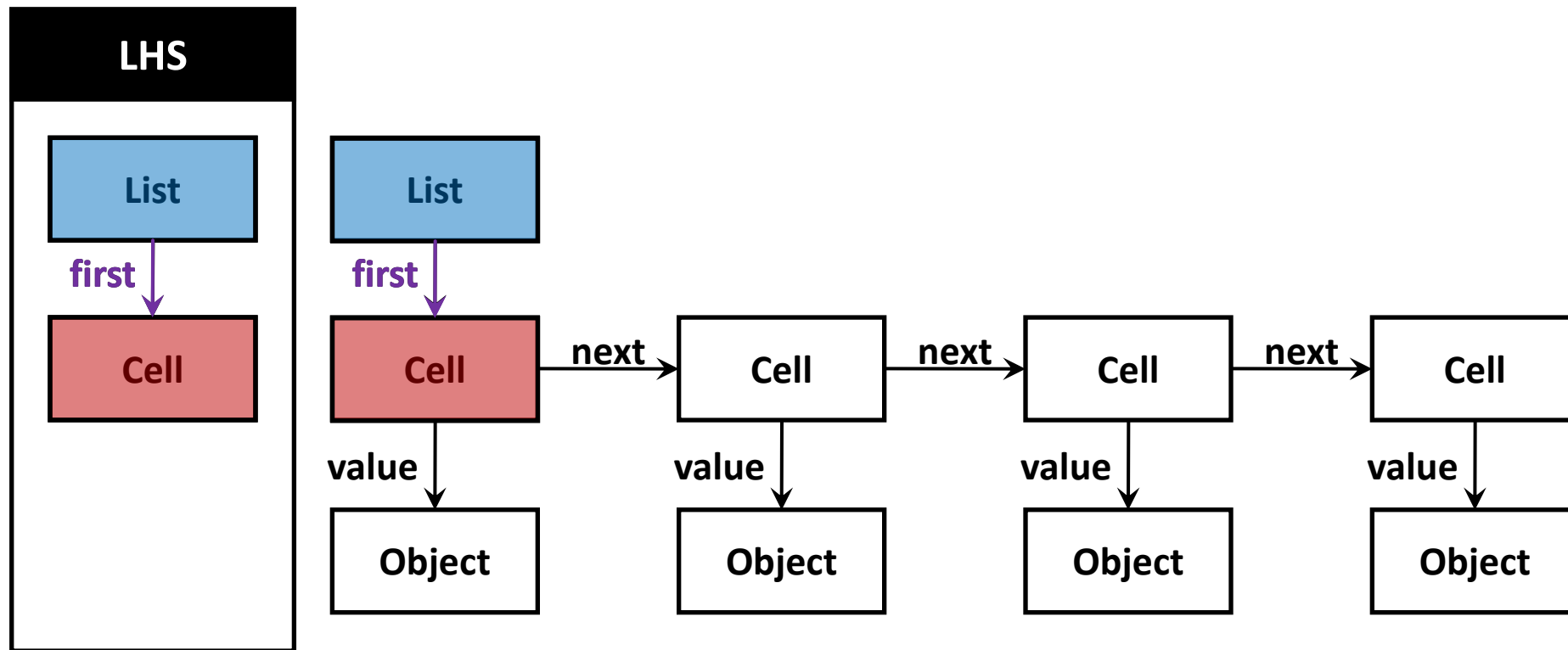
Gráftranszformáció: Mintaillesztés

- **Illesztés:** megkeressük a LHS-t tartalmazó részgráfokat a forrás gráfban



Gráftranszformáció: Mintaillesztés

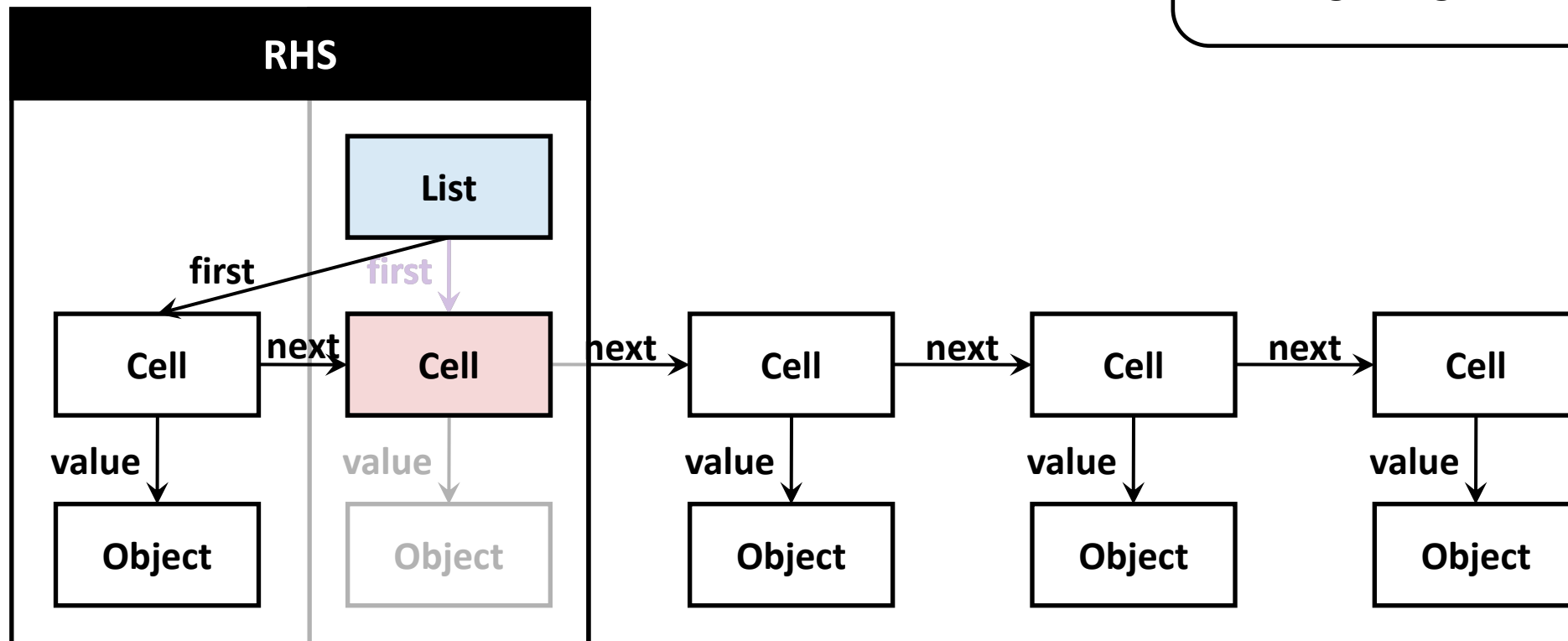
- **Illesztés:** megkeressük a LHS-t tartalmazó részgráfokat a forrás gráfban



Gráftranszformáció: Átírás végrehajtása

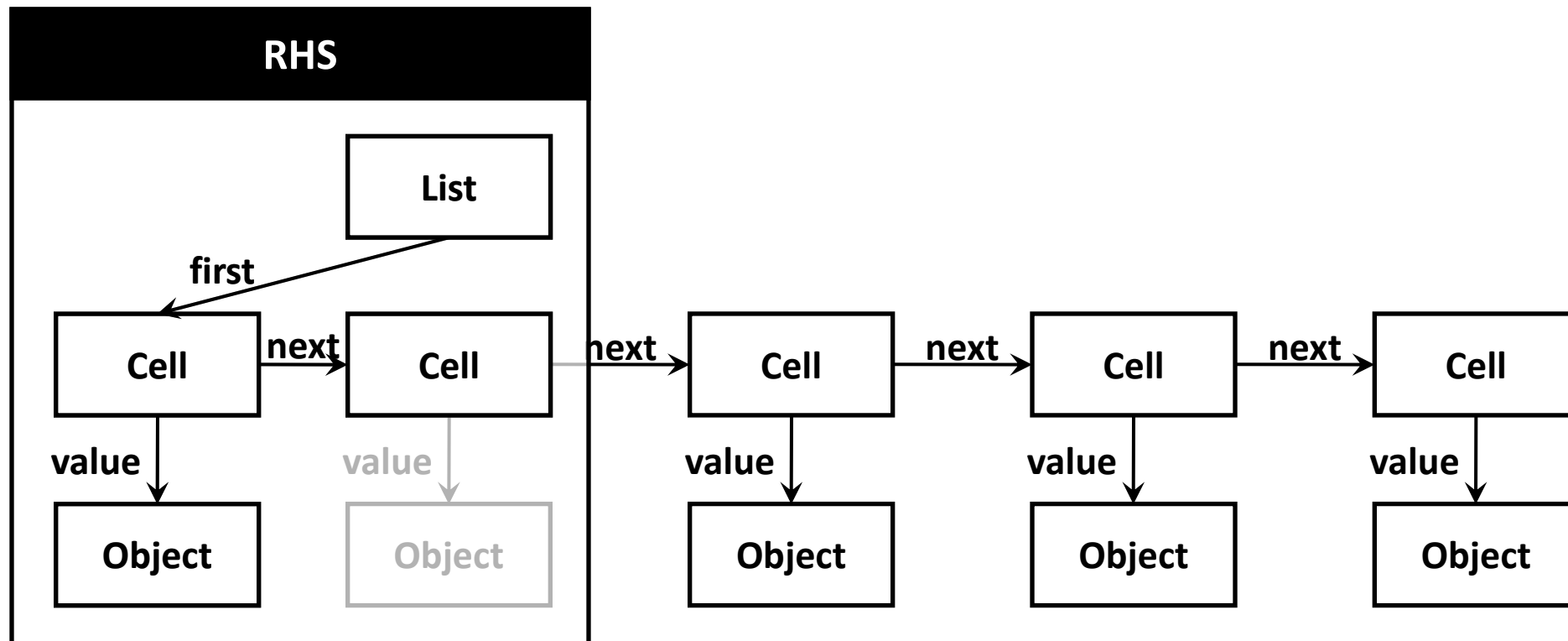
- Illesztés mentén lecseréljük az LHS-t RHS-re.

LHS\RHS → Töröl
RHS\LHS → Beszúr
RHS\LHS → Békénhagy



Gráftranszformáció: Átírás végrehajtása

- Új gráfot kapunk



Gráftranszformációk értelmezése

Kétféle értelmezést szoktak használni:

- Operacionalizált

- > Modellelemek törlése és beszúrása
- > Könnyen végrehajtható és implementálható

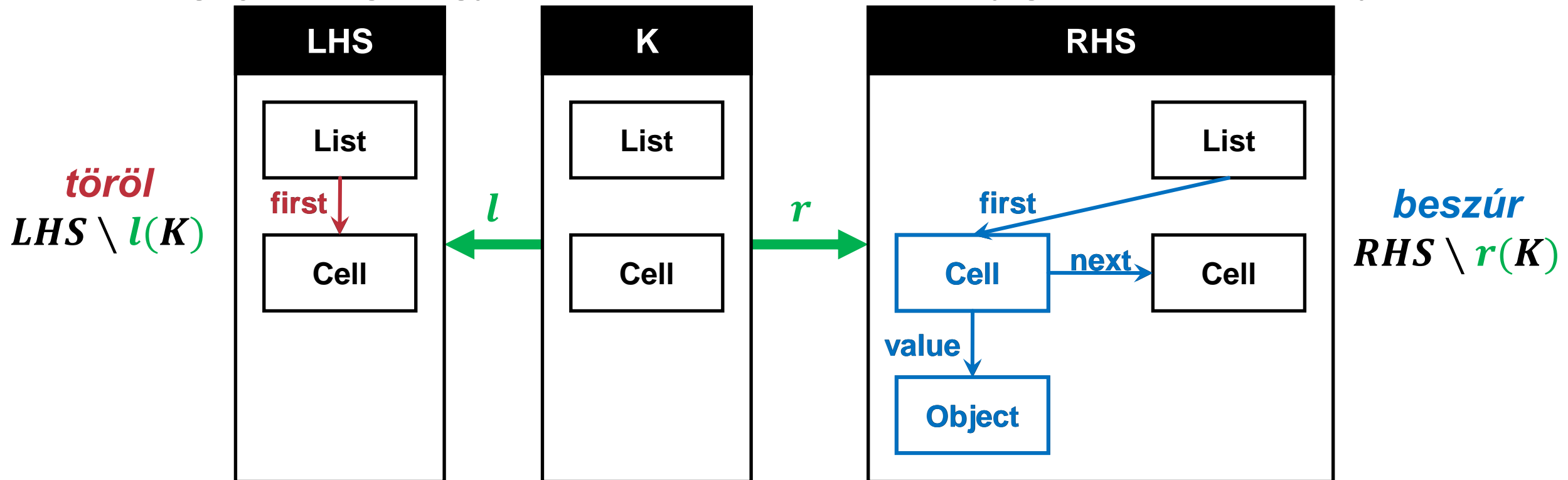
- Matematikai viszonyok vizsgálata

- > Leírja hogy a dokumentumoknak milyen viszonyokban kell lenniük egymással
- > Hasznos a bizonyításokhoz

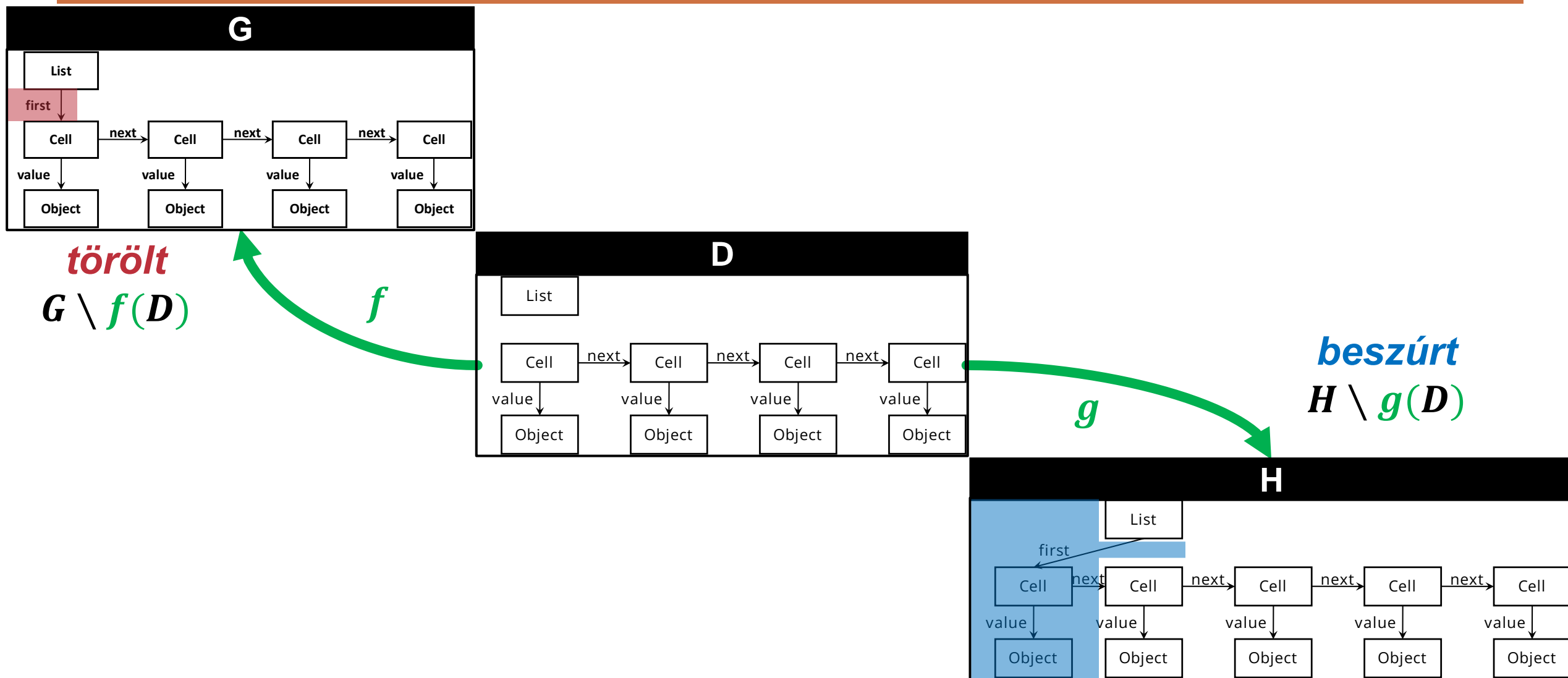
→ Matematikailag precíz és hatékonyan végrehajtható formalizmus

Gráftranszformációk anatómiája

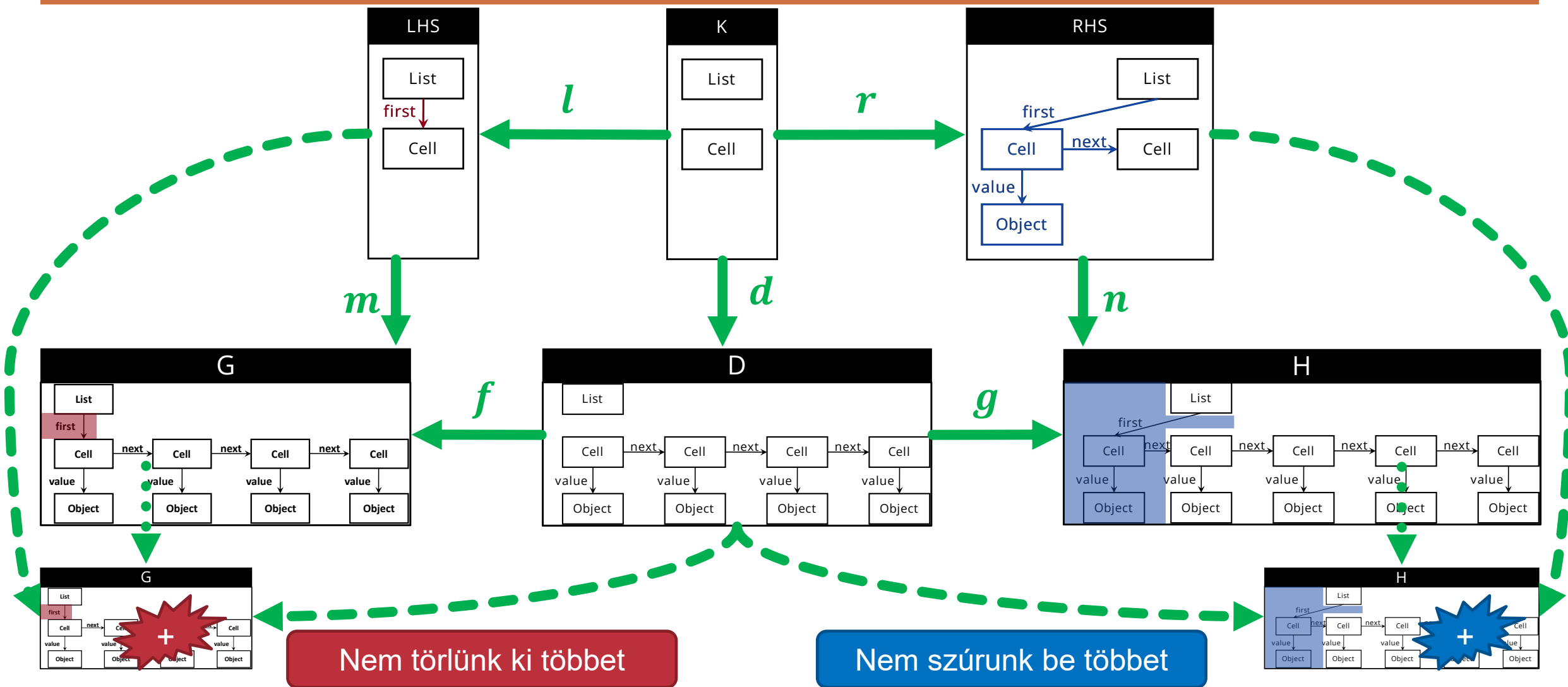
- Vizsgáljuk meg, hogy a transzformáció során mely gráf illeszthető melyikbe!



Gráftranszformációk anatómiája



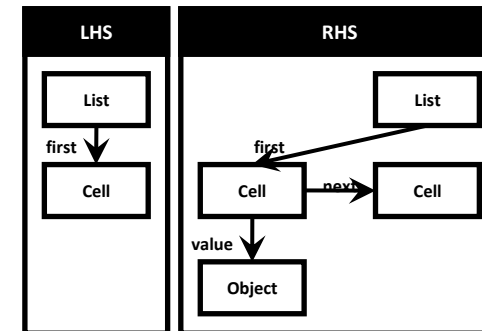
Teljes anatómia



Gráftranszformáció

- Szabályok megfogalmazása modellek átírására
- Nyelvtani szabályok kiterjesztése

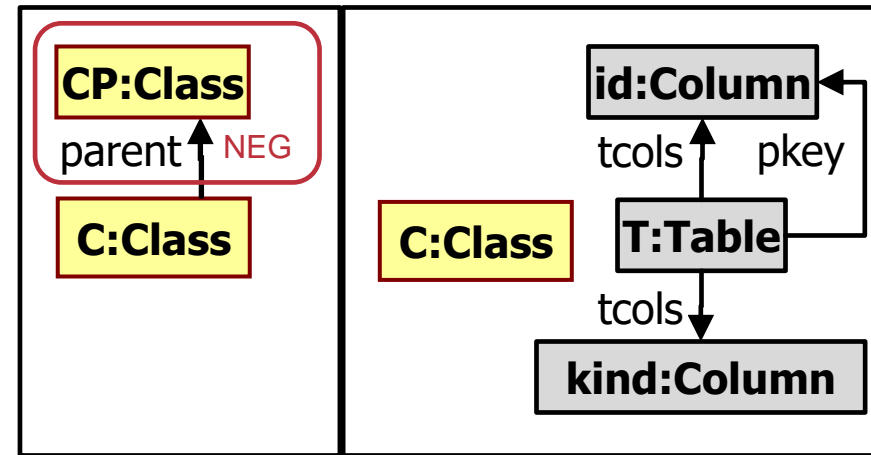
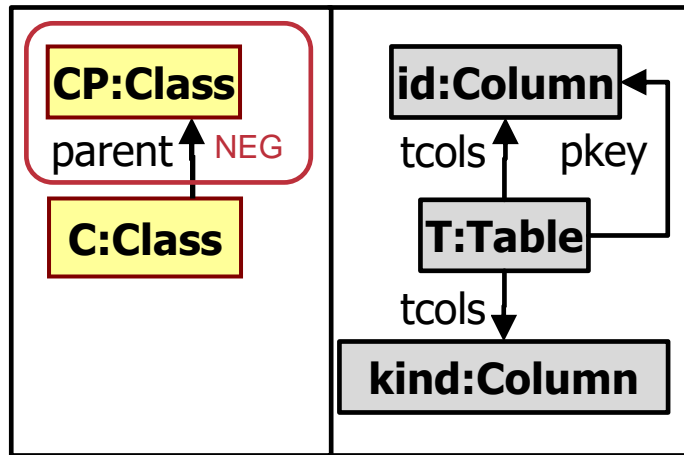
$List \rightarrow List, Cell$ **vs**



- Szemléletes, de matematikailag precíz formalizmus
(Terminálódás, Sorrendezés, Konfluencia, ...)
- Eszköztámogatás (lásd előző gyakorlat)

Példák

- Ősosztályok leképezése (törléssel és törlés nélkül)



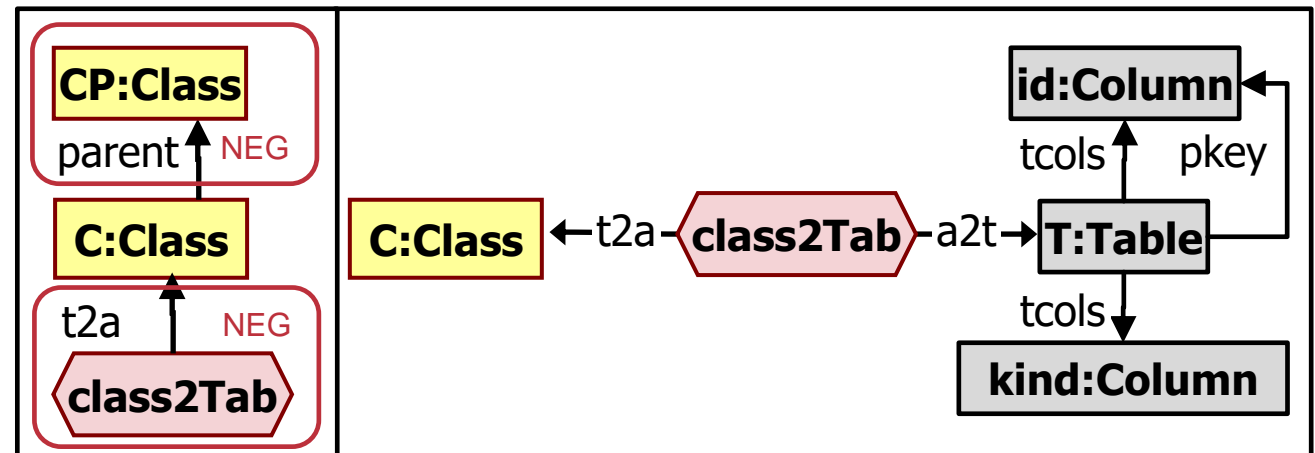
- Mi történik, ha kitörlünk egy elemet, amire mutat még él?



- „Lógó élek” megoldása: Töröljük az éleket / Visszavonjuk a transzformációt

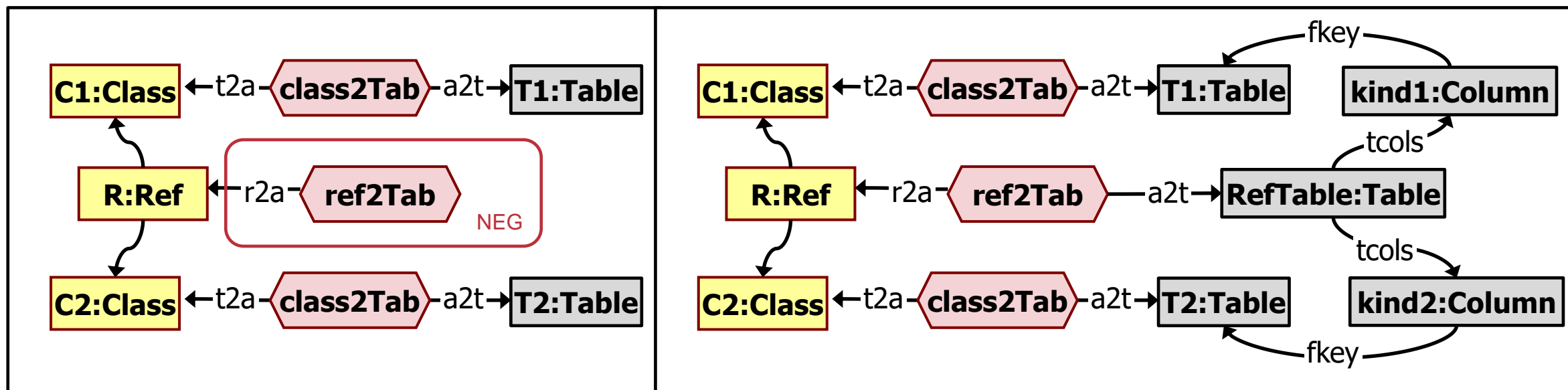
Példák

- Ősosztályok leképezése nyomonkövethetőséggel:
 - > Keressünk olyan ősosztályt,
 - > amely még nem lett leképezve,
 - > majd képezzük le.



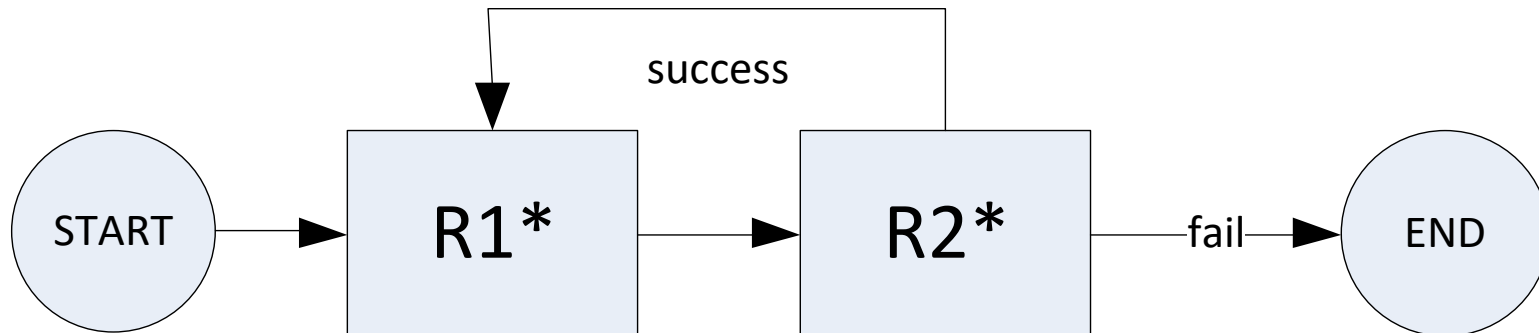
Példák

■ Referenciák leképezése

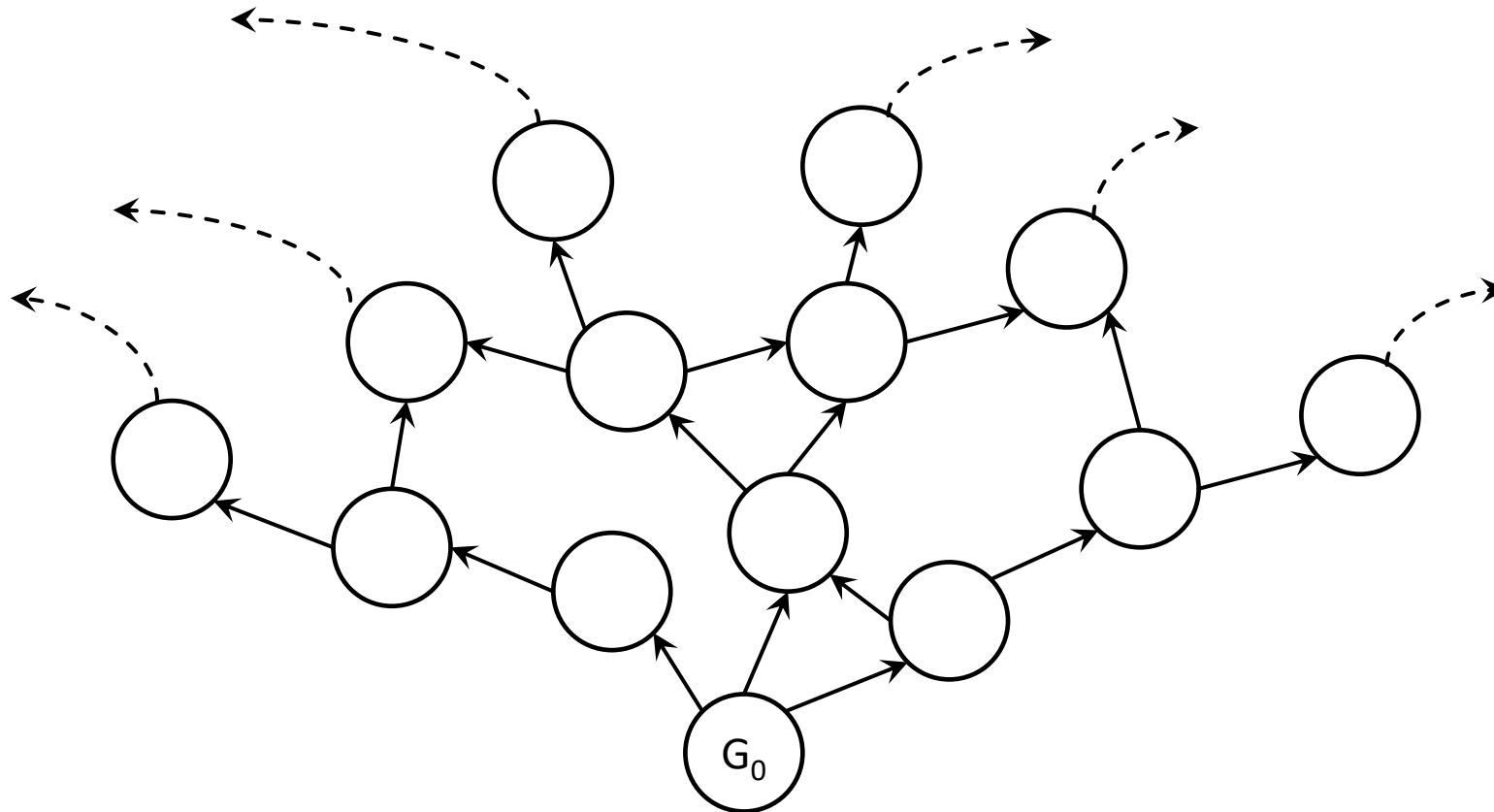


Vezérlési szerkezetek

- Milyen sorrendben hajtsuk végre a szabályokat?
- Több lehetőség, lásd előző előadás.
- De például:
 - > Tűzelj szabadon választott transzformációkat, amíg ez lehetséges (~ alapértelmezett)
 - > Tűzeld el az összes szabad transzformációt egyszer
 - > vezérlési gráf (explicit vezérlés)

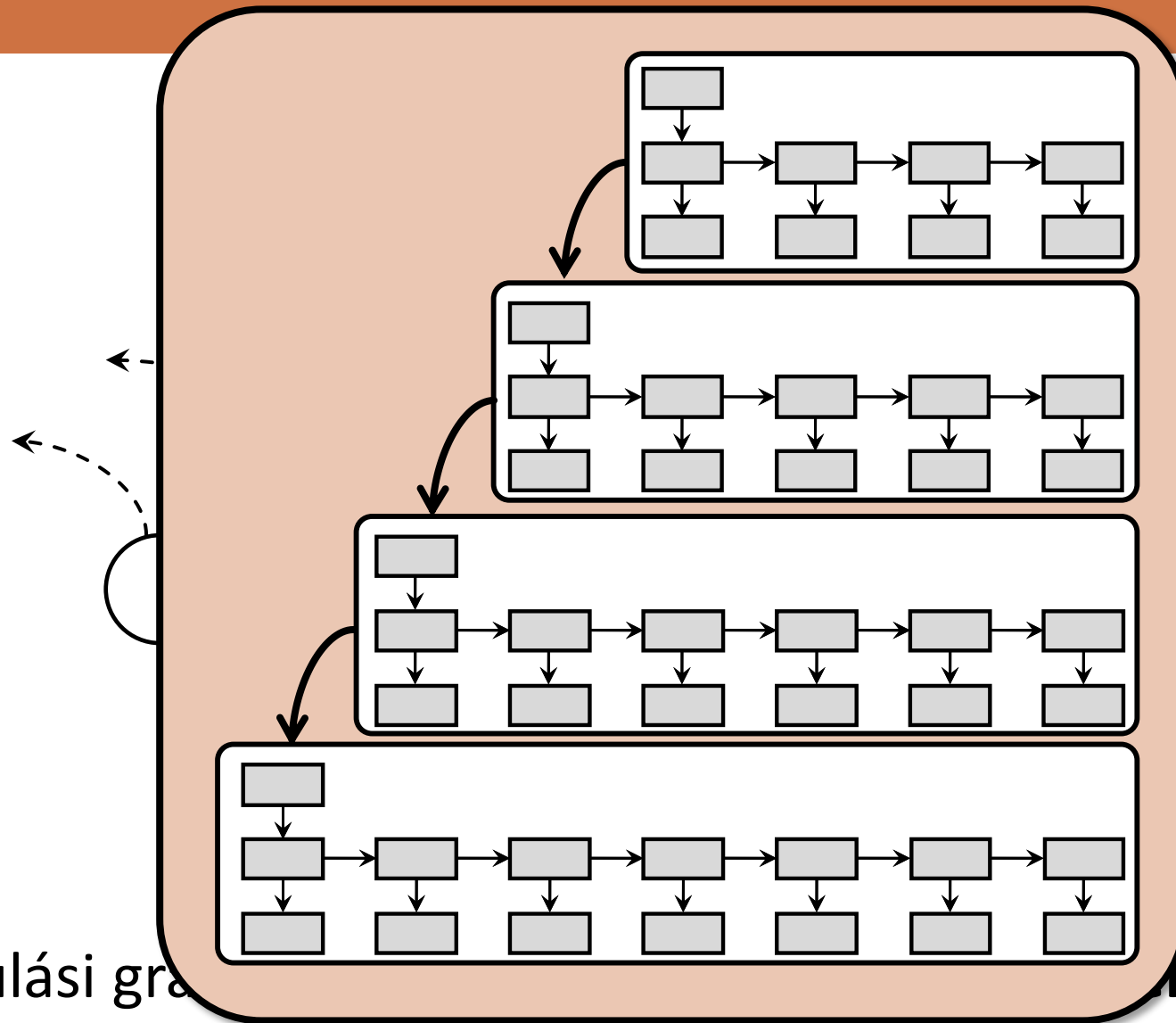


Állapottér



Kiindulási gráf + GT szabályok \rightarrow (tipikusan végtelen) Állapottér

Állapottér



Még egyszerű példa esetén is végtelen

Olyan rendszereket jellemezhetünk, amelyek:

- Nemdeterminisztikusak
- Végtelen állapottérrel rendelkeznek

Kiindulási gra

(en) Állapottér

Modeltranszformációk típusai

- A bemenetek és kimentek száma
 - > In-place: ugyanaz a bemenet és a kimenet, azaz a modellt felülírjuk (pl szimuláció léptetése, quick fix)
 - > Out-place: A kimenet egy másik modell (pl: ORM esetén osztálydiagram → Tábla)
- A nyelv szerint
 - > Endogén: Ugyanaz a metamodel (pl control-flow egyszerűsítése, eredeti megtartása)
 - > Exogén: Különböző a metamodel
- Az irány szerint
 - > Egyirányú: Akkor van értelme, ha egy forrás modellből egy célmodellt készítünk (ORM példa)
 - > Kétirányú: Mindkét irányból végrehajtható (PI ha adatbázis sémából is tudnánk osztályokat készíteni)

Gráfmintaillesztés, Gráftranszformáció

Alapfogalmak

Gráfmintaillesztés

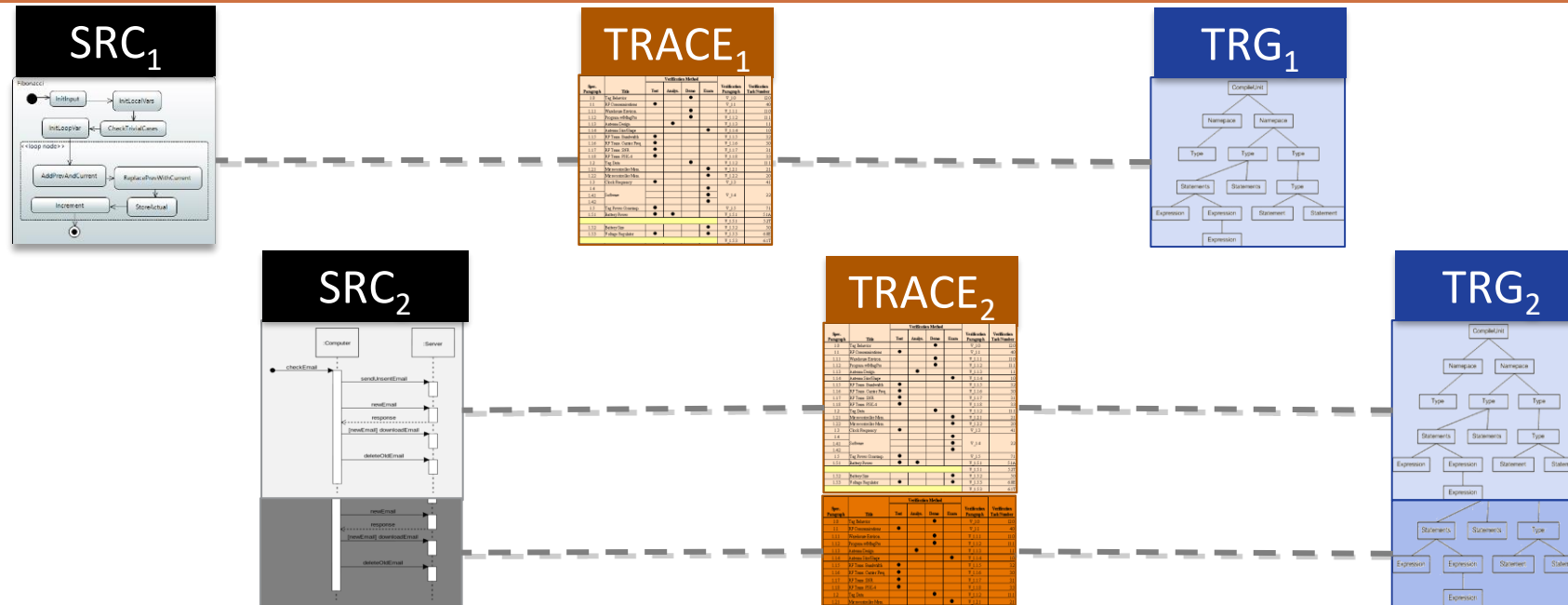
Modelltranszformációk

Inkrementális transzformációk

Tervezésítér bejárás



Inkrementális vérgelhajtás: Batch transzformáció

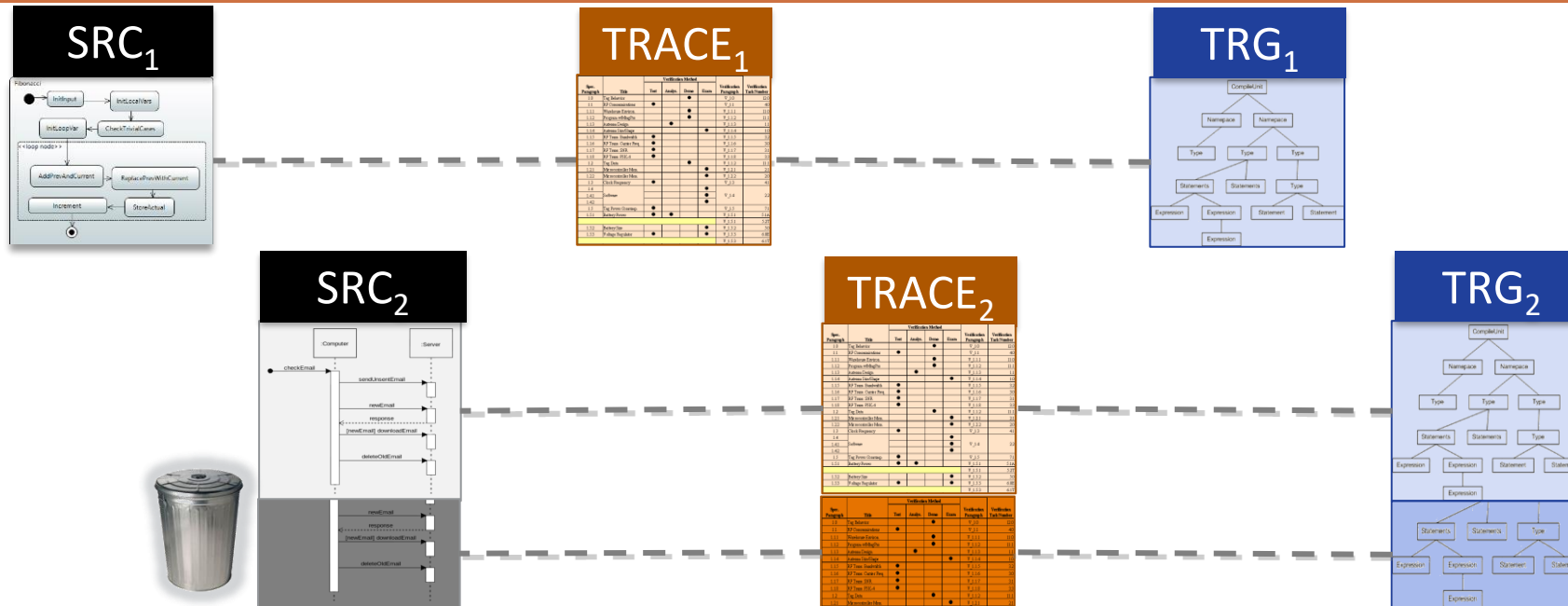


1. Első transzformáció

2. Forrásmodell megváltozik

3. Újrafuttajuk az elejétől az összes forrásmodellre

Piszkos Inkrementalitás



Előnyök:

- Nagy lépésenkénti inkrementalitás
- Kerüli a folyamatos végrehajtást

Hátrányok:

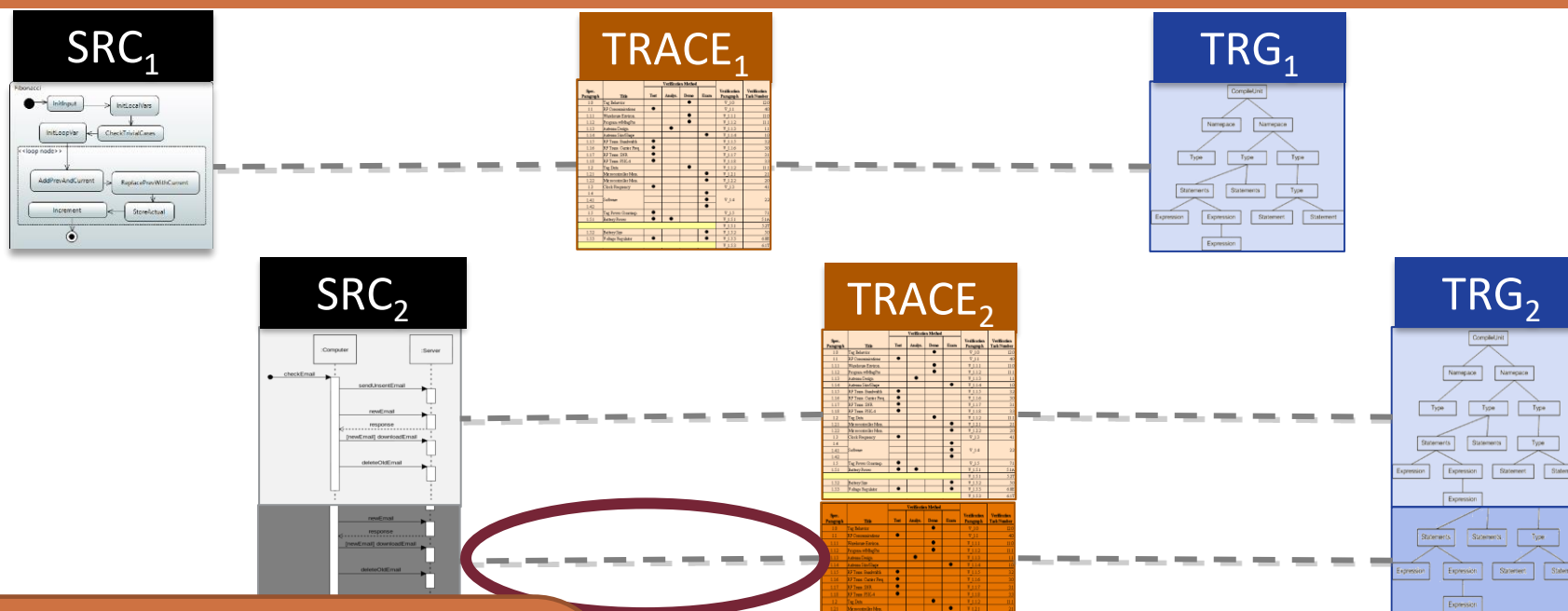
- Összetett MT lassú lehet
- Tisztítás (hiba után)?
- Láncolás?

1. Első transzformáció

2. Forrásmodell megváltozik

3. Újrafuttatjuk az elejétől csak a módosított modellekre

Inkrementális nyomonkövethetőséggel



Előnyök:

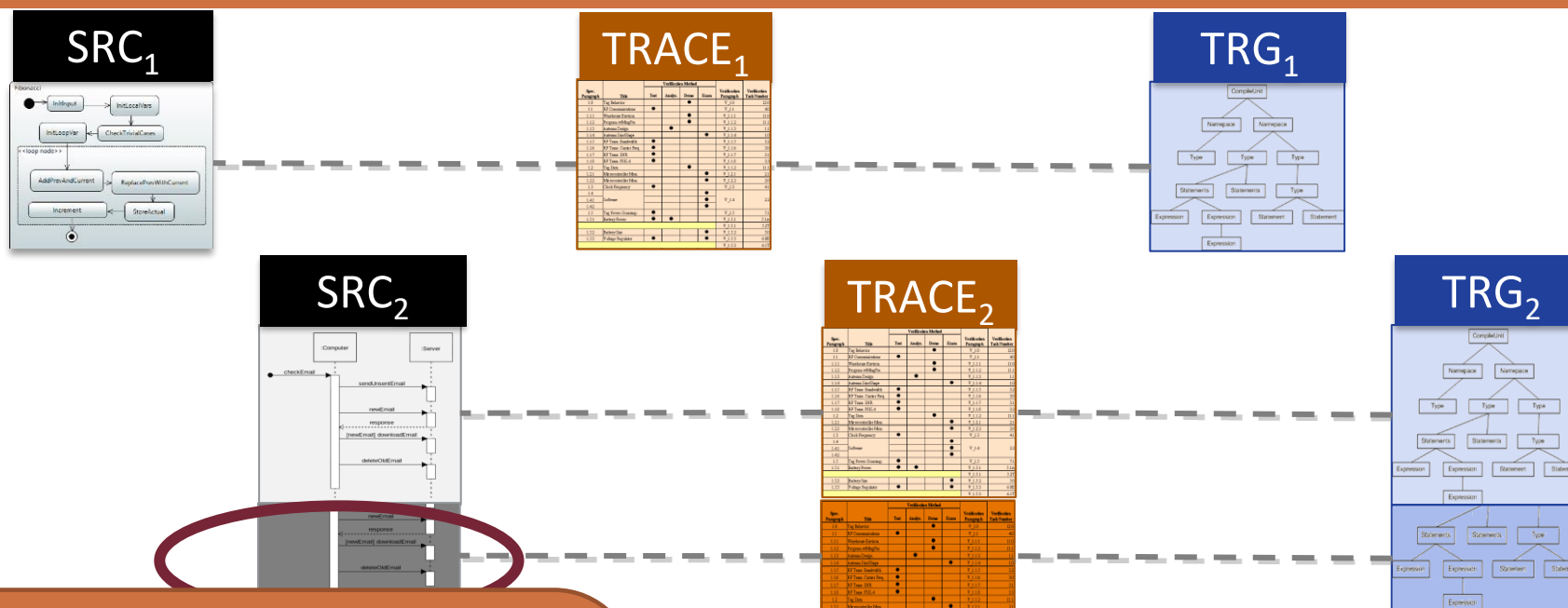
- Kis lépésenkénti inkrementalitás
- Jobb teljesítmény

Hátrányok:

- Nagymértékben függ a nyomkövethetőségi kapcsolatokról
- Intelligens illesztőre van szükség

1. Első transzformáció
2. Forrásmodell megváltozik
3. Hiányzó nyomkövetési kapcsolatok felderítése
4. MT újbóli végrehajtása csak a nem nyomon követhető elemek esetében

Eseményvezérelt Transzformációk



Előnyök:

- Finomított kontextus: a lekérdezés eredményének változásai vezérlik
- Láncolás
- Elkerüli a folyamatos számolást

Hátrányok:

- Nyelvi szintű korlátozások
- Élőben kell "figyelni"

1. Első transzformáció

2. Forrásmodell megváltozik

3. Change notification feldolgozása

4. Változás terjesztése



Inkrementális Előre Történő Transzformáció

■ Célok: munka megtakarítása...

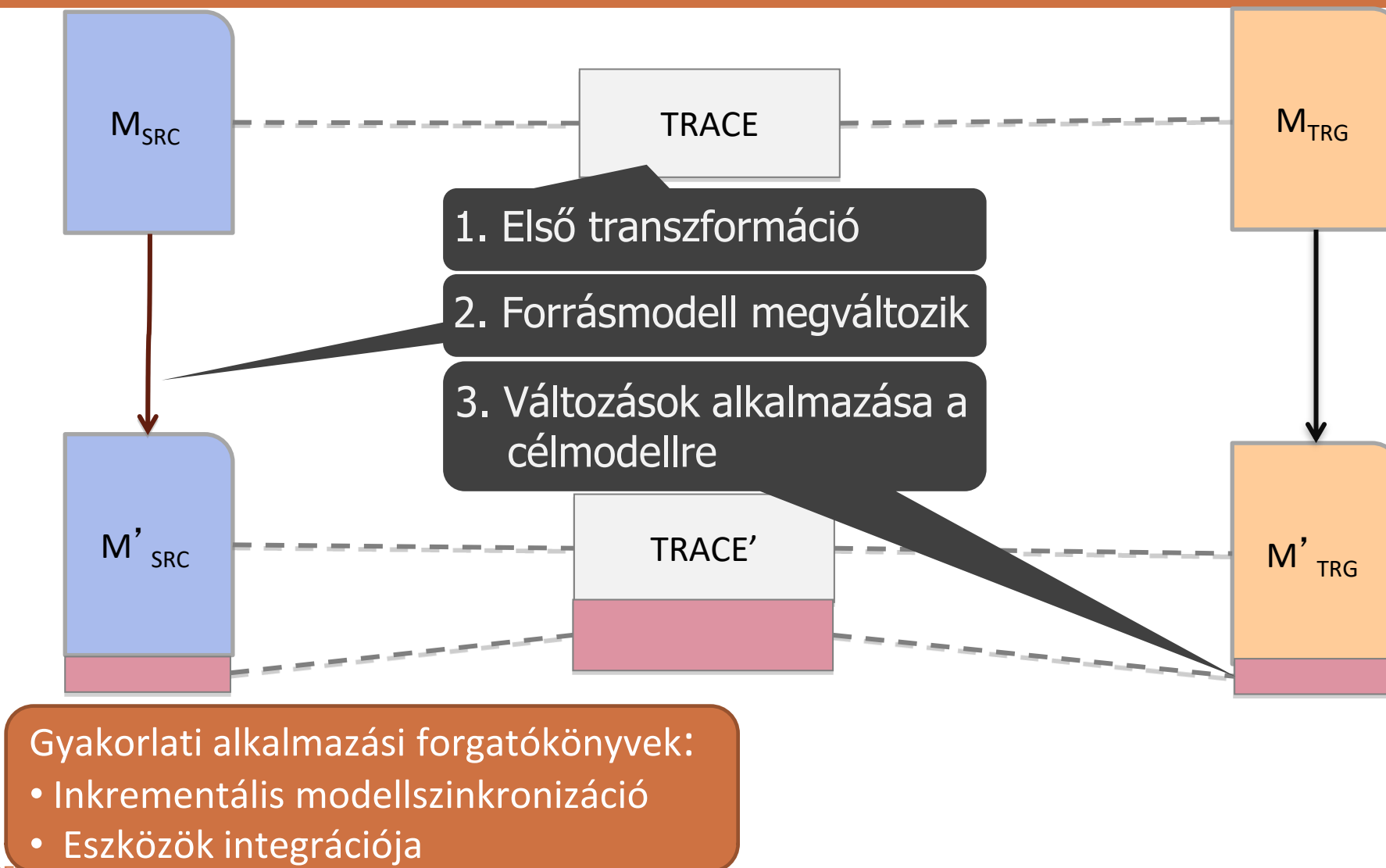
> Cél Inkrementalitás

- ...a cél változatlan részeinek újrafelhasználása által
- További előnyök
 - A változatlan részekhez meglévő kapcsolatok megmaradnak
 - A változatlan részekre meglévő elemzések megmaradnak
 - Nem terjed tovább a transzformációs láncok mentén

> Forrás Inkrementalitás

- ... a forrás változatlan részeinek figyelmen kívül hagyása által
- Használjunk inkrementális modellekérdezést!

Inkrementális Visszafelé Történő Transzformáció

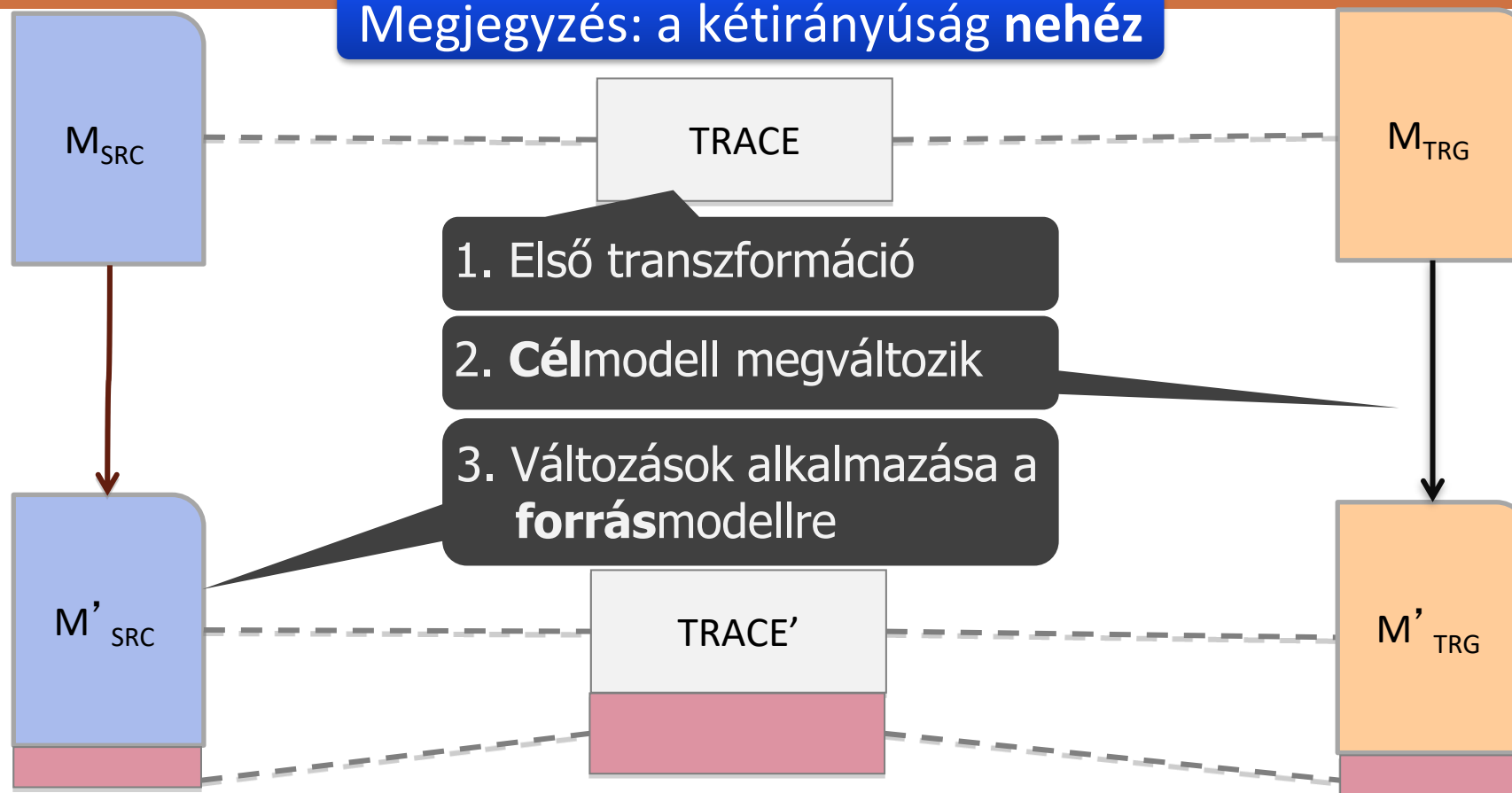


Inkrementális szabályok közös mintája

- Kihívás az inkrementalításban:
 - > Mi van, ha a kontextust korábban eltávolítjuk?
 - > Pl. Táblázat törlése az oszlopai előtt.
- Megoldások
 - > Szabályok sorba rendezése
 - Prioritások
 - Extra alkalmazási feltételek
 - > Törlési szabály kíméletes / toleráns a hiányzó kontextussal szemben
 - > A kontextustörlési szabály eltávolítja a függő elemeket

Inkrementális Visszafelé Történő Transzformáció

Megjegyzés: a kétirányúság nehéz



Extra kihívás, ha nem elég nehéz:
SRC \rightarrow TRG megadott
TRG \rightarrow SRC következtetett

Újabb megközelítések::

A. Schürr, P. Stevens, N. Foster, T. Hettel,
Cicchetti&Pierantonio, Czarnecki&Diskin

Gráfmintaillesztés, Gráftranszformáció

Alapfogalmak

Gráfmintaillesztés

Modelltranszformációk

Inkrementális transzformációk

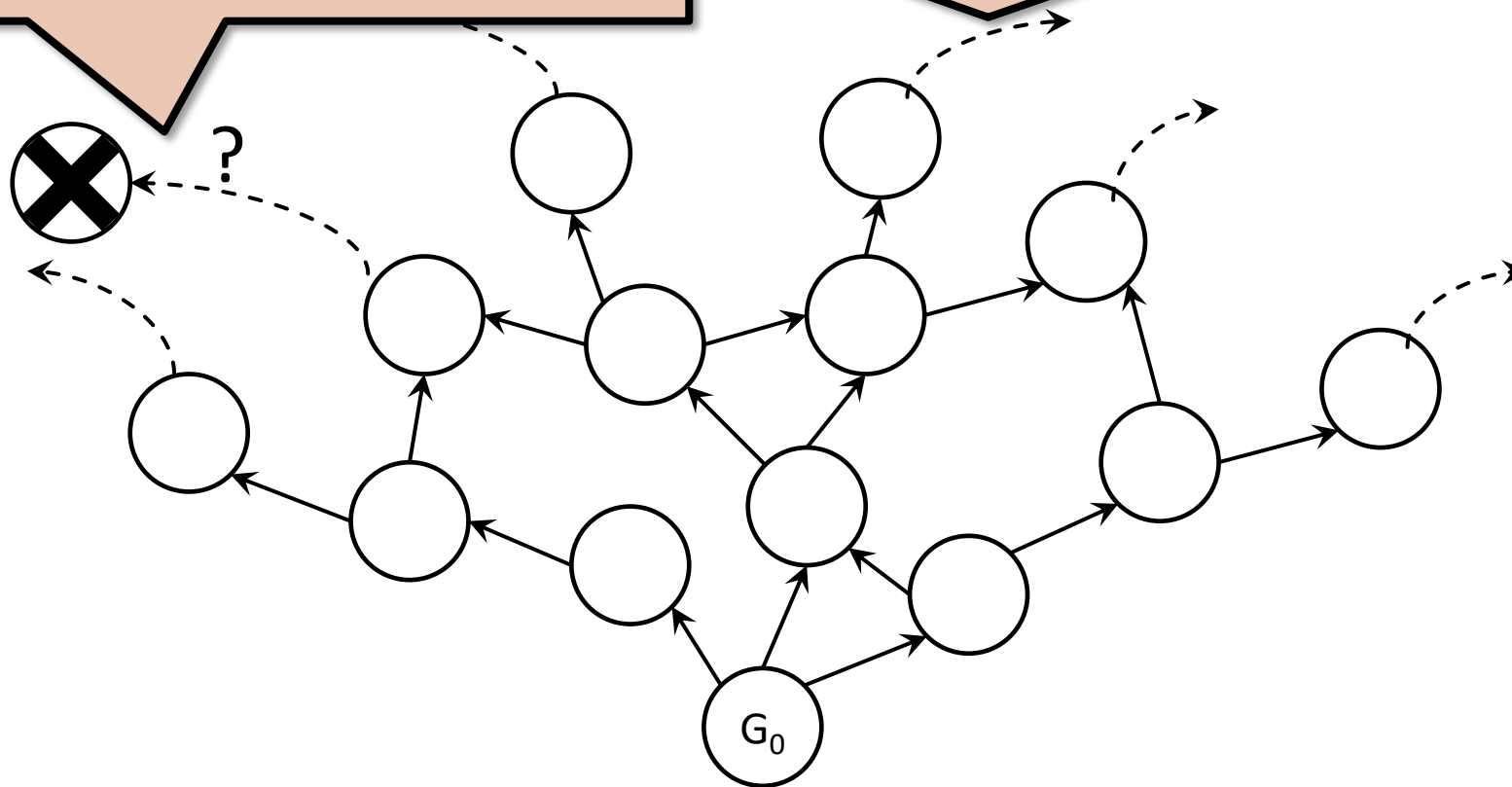
Tervezésítér bejárás



Visszatekintés: GT rendszer állapottere

A megoldások az
állapottérben vannak

Potenciálisan végtelen
állapottér



Kiindulási gráf + GT szabályok \rightarrow Állapottér

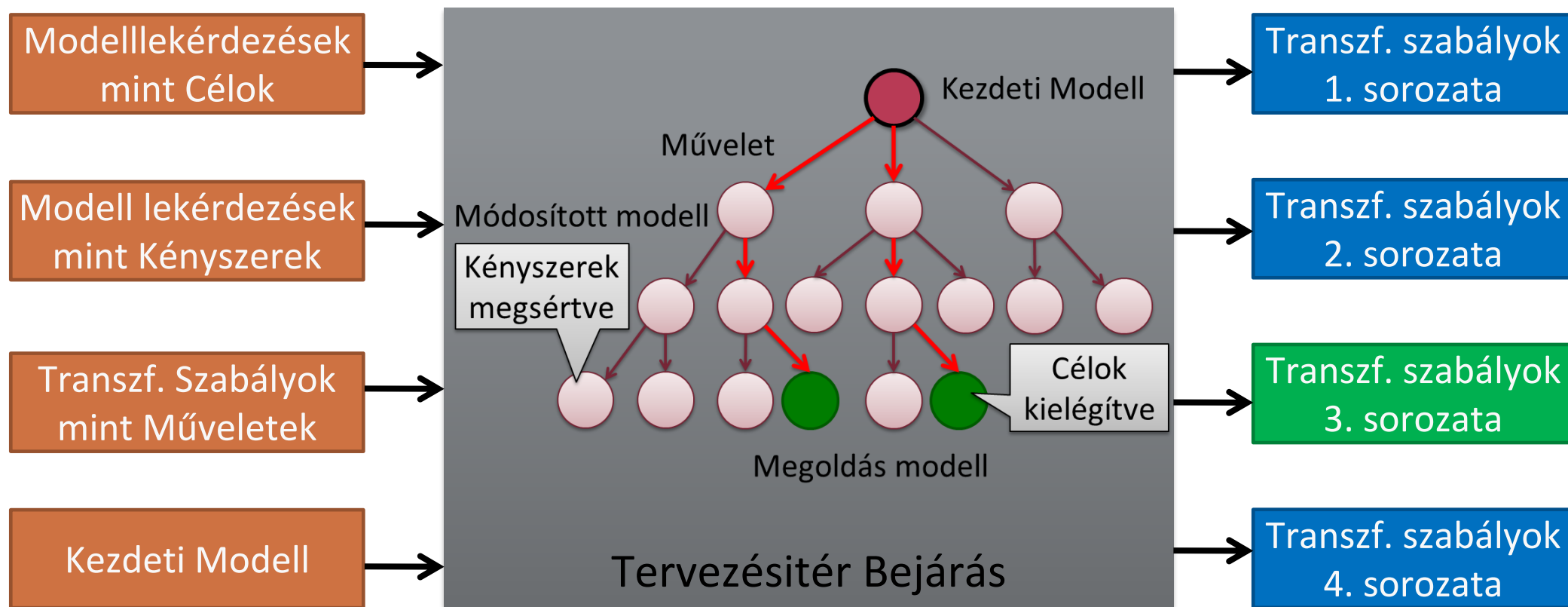
Tervezésítér bejárás



Speciális állapottér bejárás

- potenciálisan végtelen állapottér
- „sűrű” megoldási tér

Modellvezérelt Irányított Tervezésítér Bejárás

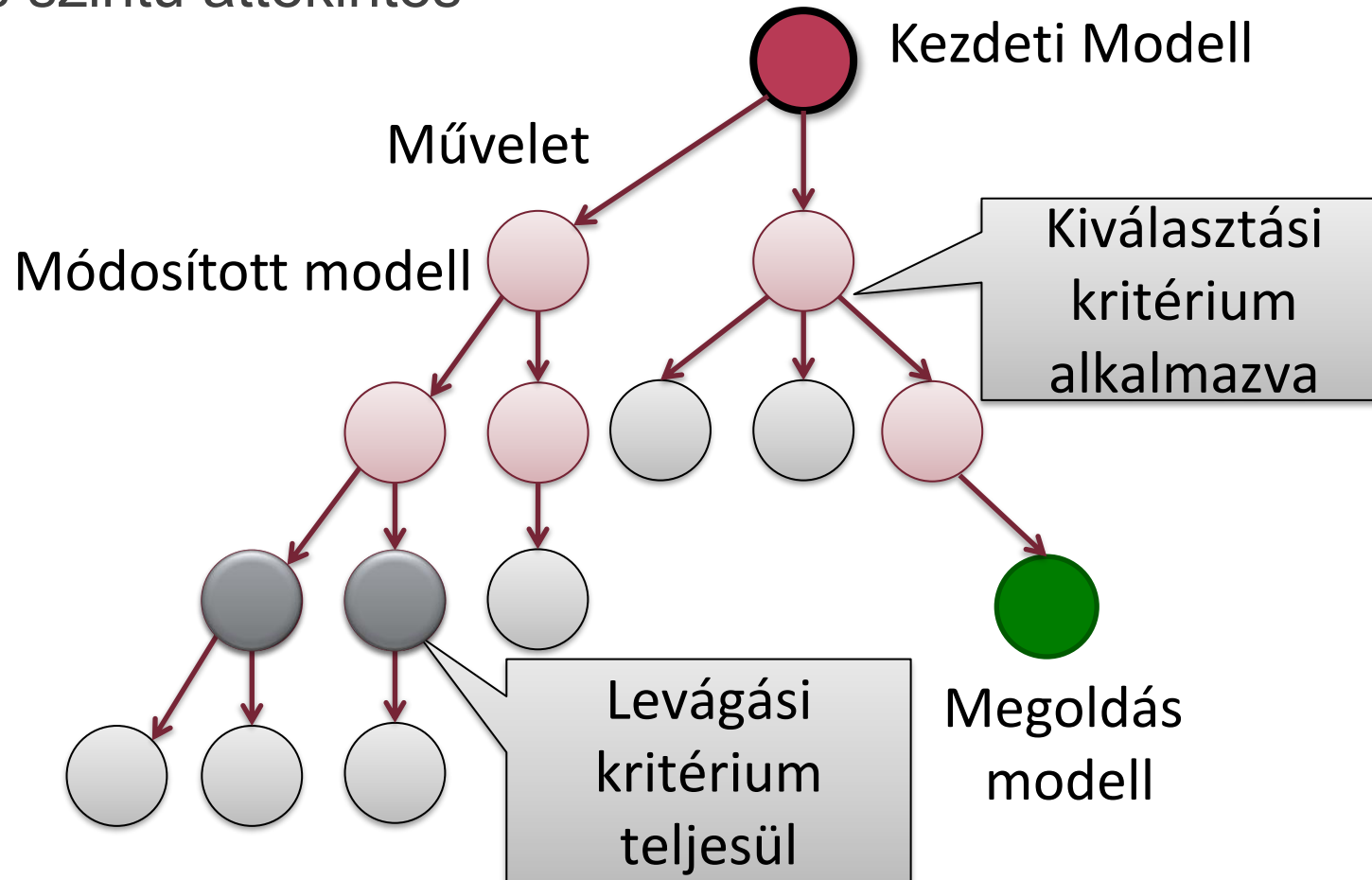


Útmutatás a bejáráshoz: Tippek

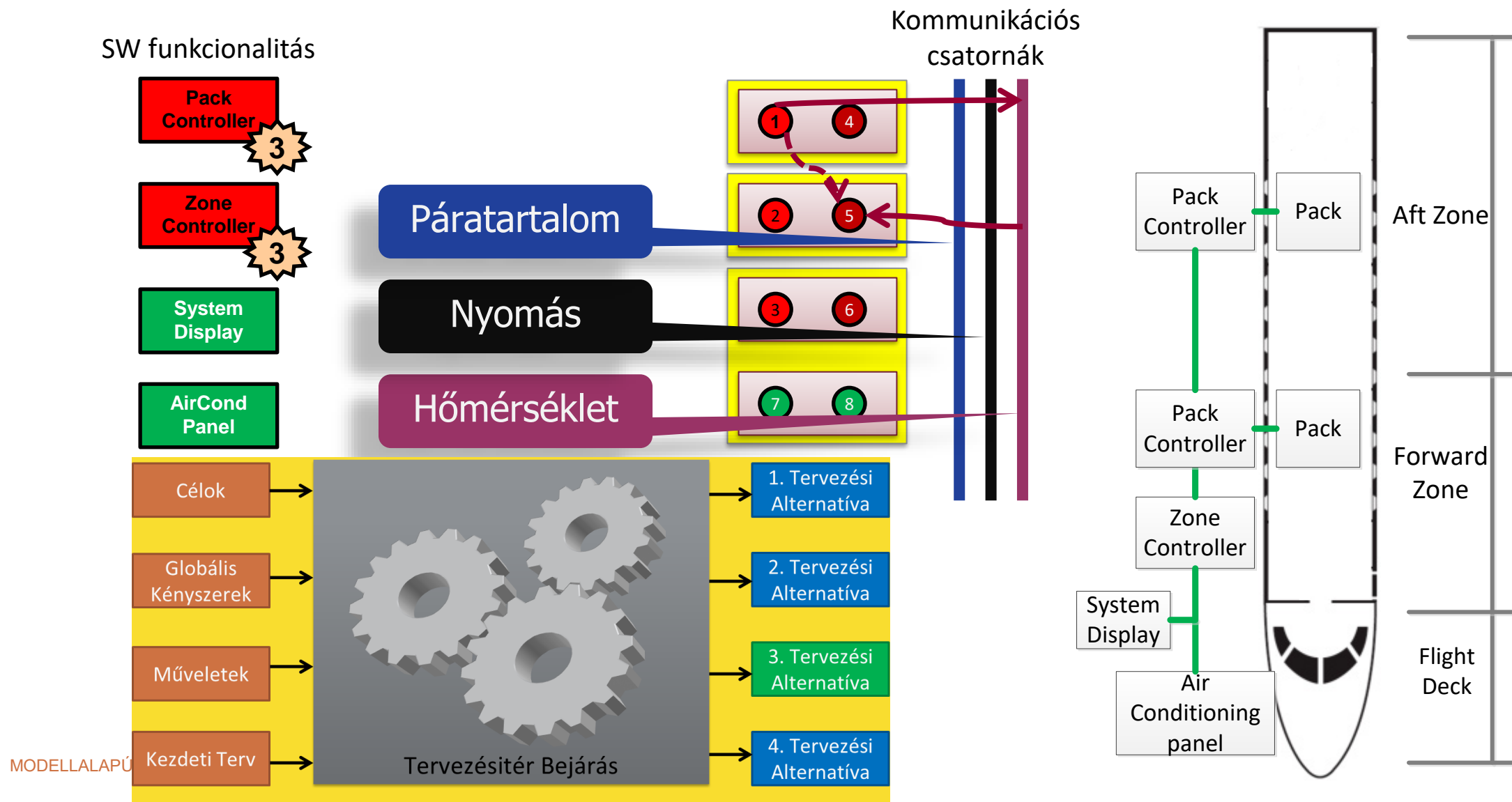
- tervező / végfelhasználó
- formális elemzés

Irányított Tervezésítér Bejárás

- Magas szintű áttekintés

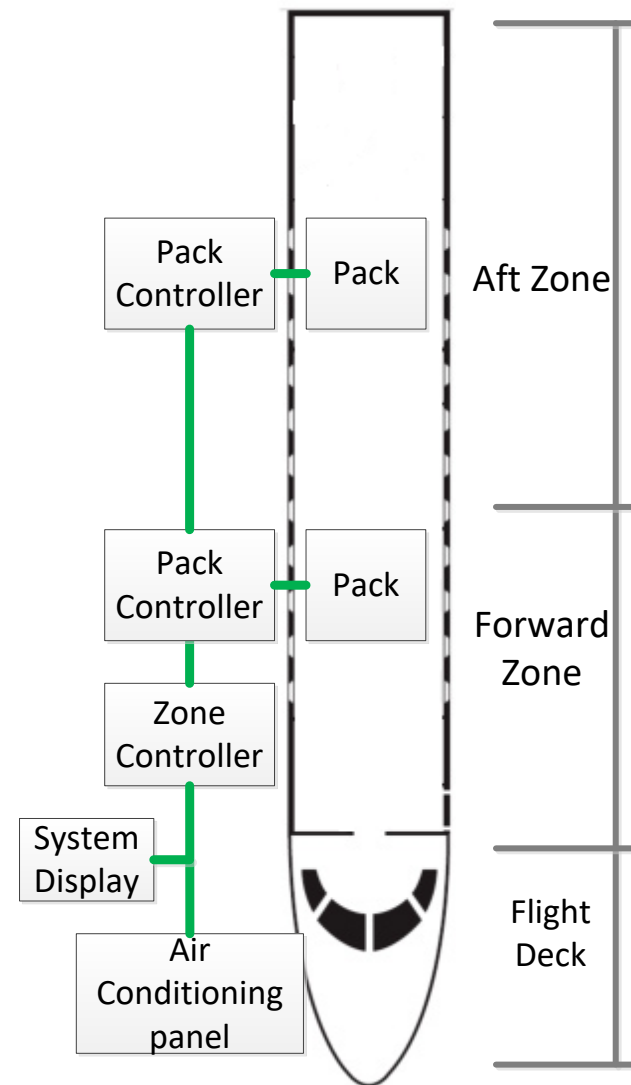
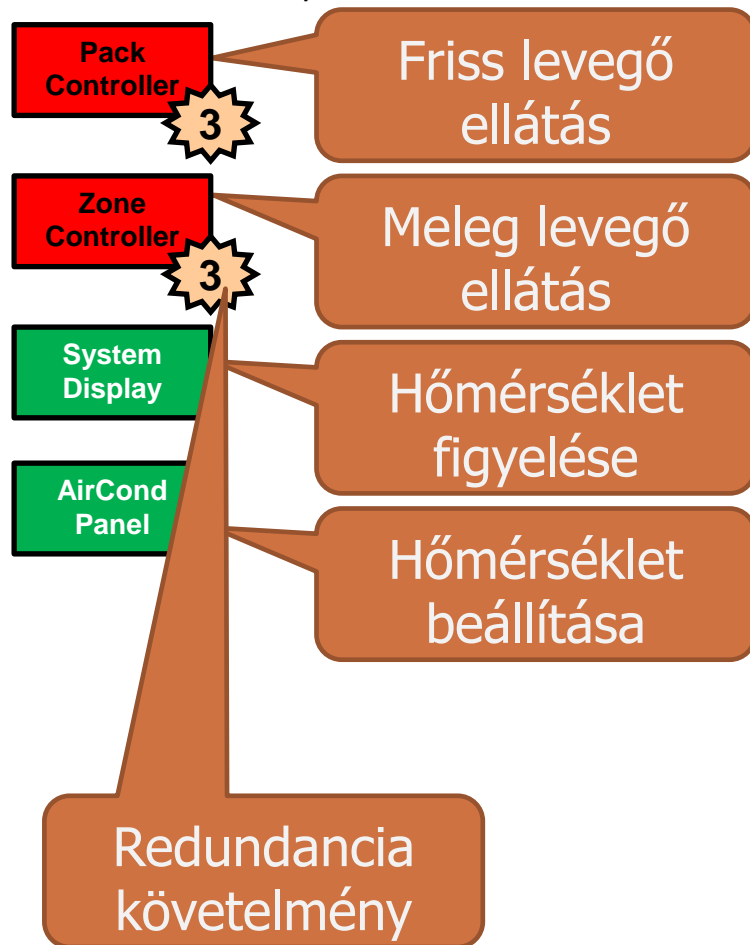


Tervezésítér Bejárás IMA Konfiguráció Tervezéséhez



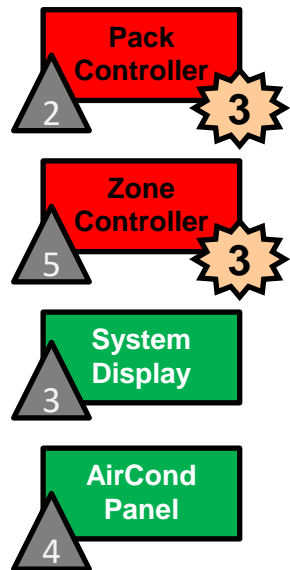
ARINC653 konfigurációk tervezése

SW funkcionalitás
(kritikus + nem kritikus)

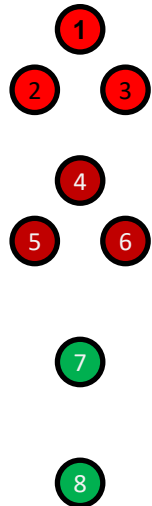


Munkapéldányok, Partíciók, Modulok

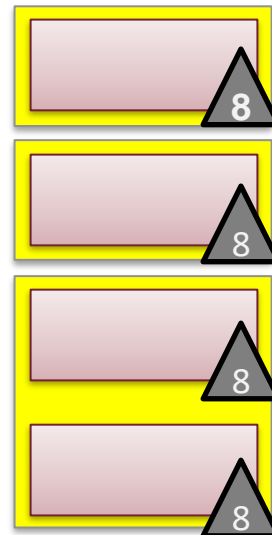
SW funkcionalitás



Munkapéldányok



Partíciók

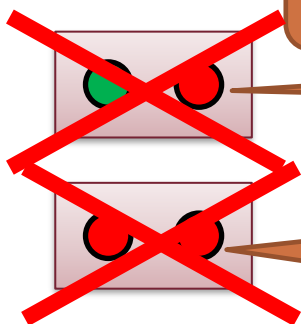


Modulok

További kényszerek

- WCET,
- ütemezés stb.
- interfészek
- adattípusok

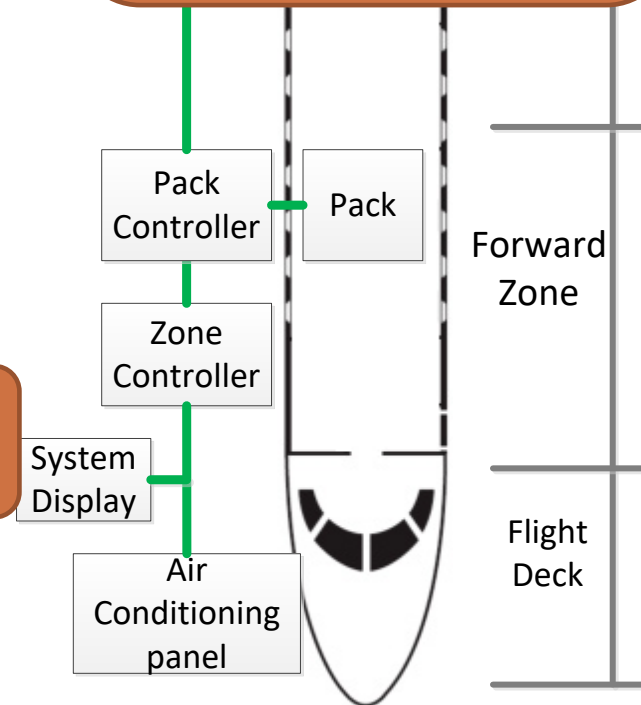
Kényszerek



Memóriaigény
+ kényszerek

Ne keverjen kritikus és
nem kritikus feladatokat

Ne keverje ugyanazon
kritikus feladat
példányait



Gráfmintaillesztés, Gráftranszformáció

Alapfogalmak

Gráfmintaillesztés

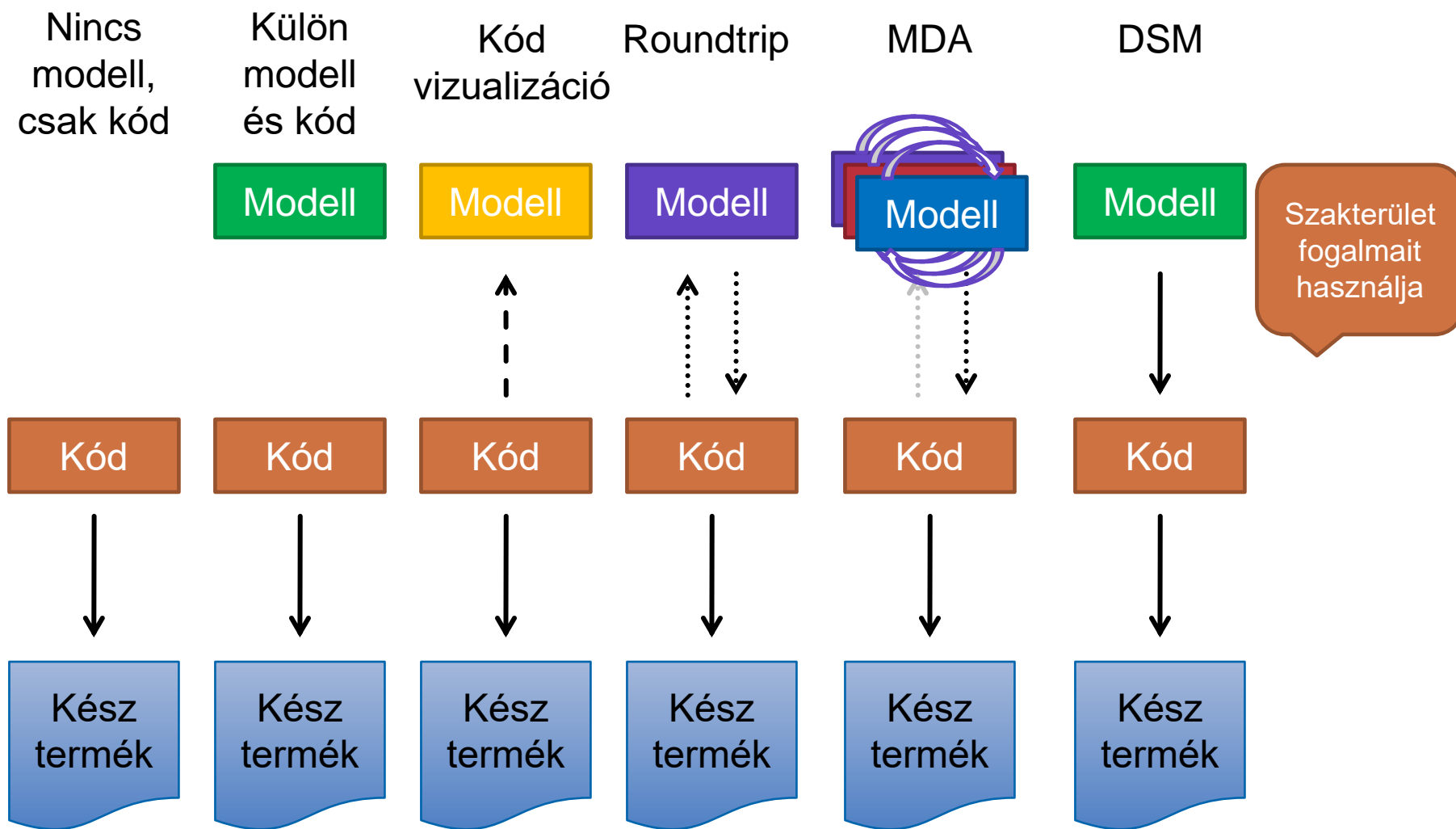
Modelltranszformációk

Inkrementális transzformációk

Tervezésítér bejárás



Hogyan használunk modelleket?





Köszönöm a figyelmet!