Modellalapú szoftverfejlesztés

XI. előadás

Gráfmintaillesztés,
Gráftranszformáció

Dr. Semeráth Oszkár

Modelltranszformáció

- **Alapfogalmak**
- **Transzformációk láncolása**
- **Szabályalapú transzformációk**
- **Technológiák**

Motiváció

- Modell nem csak dokumentációs céllal készül…
  - Generálunk belőle másik modellt…
    - Pl State chart -> Petri háló (analizálás)
    - BPMKN<->BPEL
  - … vagy más állományt …
    - Forráskód
    - Dokumentáció
    - Kimutatás, jelentés
    - Terv, ami feldolgozható egy külső alkalmazásnak, HW-nek
- … ezért a modellfeldolgozás előtérbe kerül

Microsoft T4

- **T**ext **T**emplating **T**ransformation **T**oolkit
- Szöveg blokkok és vezérlési logika egy fájlban
  - Szöveg blokk kimásolódik a kimenetre
  - C# vagy VB
    - Tud írni a kimenetre
  - Hasonló, mint az ASP.NET, PHP, …
- Ahol használják: DSL Tools, Entity Framework, VMTS…

Vezérlési blokkok

- Kód blokk: <# … #>
- Kifejezés blokk: <#= … #>
  - Kiértékelhető
- Számok négyzete:

```
<#@ template language="C#" #>
<#int top = 10;
    for (int i = 0; i<=top; i++)  {  #>
      The square of <#= i #> is <#= i*i #>
<# } #>
```

```
The square of 0 is 0
The square of 1 is 1
The square of 2 is 4
The square of 3 is 9
…
```

Hogy is működik?

- Generálódik (és lefut):

```
<#@ template language="C#" #>
<#int top = 10;
    for (int i = 0; i<=top; i++)  {  #>
      The square of <#= i #> is <#= i*i #>
<# } #>
```

```
public partial class MyTemplate : ... {
  public string TransformText() {
    int top = 10;
    for (int i = 0; i<=top; i++) {
      this.Write("The square of ");
      this.Write(this.ToStringHelper.ToStringWithCulture(i));
      this.Write(" is ");
      this.Write(this.ToStringHelper.ToStringWithCulture(i*i));
      this.Write("\r\n");
    }
    return this.GenerationEnvironment.ToString();  }}
```

Osztály kiegészítése

- <#+ … #>
  - Helper metódusok és propertyk generálása

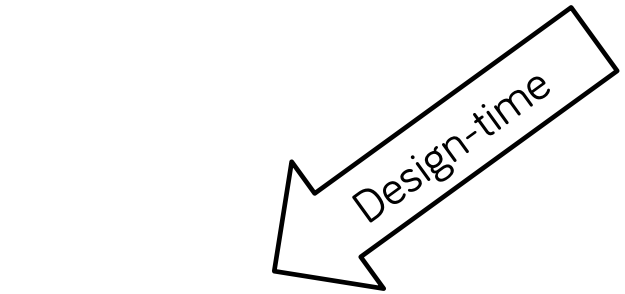
```
<#+ // Class feature block
private void WriteSquareLine(int i) { #>
    The square of <#= i #> is <#= i*i #>.
<# } #>
```

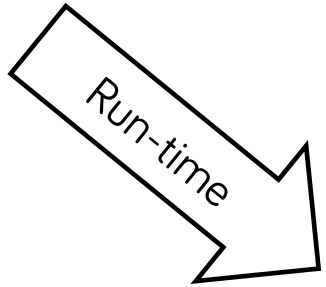
  - Tartalmazhat szöveg blokkot is
  - Hozzáadódik a generált osztályhoz, meghívható:

```
<#int top = 10;
  for (int i = 0; i<=top; i++)
     WriteSquareLine(i);
#>
```

Design-time vs run-time

```
<#int top = 5;
    for (int i = 0; i<=top; i++)  {  #>
      The square of <#= i #> is <#= i*i #>
<# } #>
```






```
The square of 0 is 0
The square of 1 is 1
The square of 2 is 4
The square of 3 is 9
The square of 4 is 16
The square of 5 is 25
```

```
public virtual string TransformText() {
  int top = 5;
  for (int i = 0; i <= top; i++) {
    this.Write(" \r\n      The square of ");
    this.Write(this.ToStringHelper.ToStringWithCulture(i));
    this.Write(" is ");
    this.Write(this.ToStringHelper.ToStringWithCulture(i * i));
    this.Write(" \r\n");  }}
```

T4 összefoglalás

- Pro
  - Jól használható, rugalmas
  - Feladatautomatizáláshoz kiváló (100+ ezer sornyi kód)
  - Gyors, bináris kód generálható belőle
- Kontra
  - T4 scriptek karbantarthatósága
  - Debug lehetőségek
  - Furcsa formázási elvárások
  - Automatikus build? Függőségek?

Modellalapú szoftverfejlesztés

XI. előadás

Gráfmintaillesztés,
Gráftranszformáció

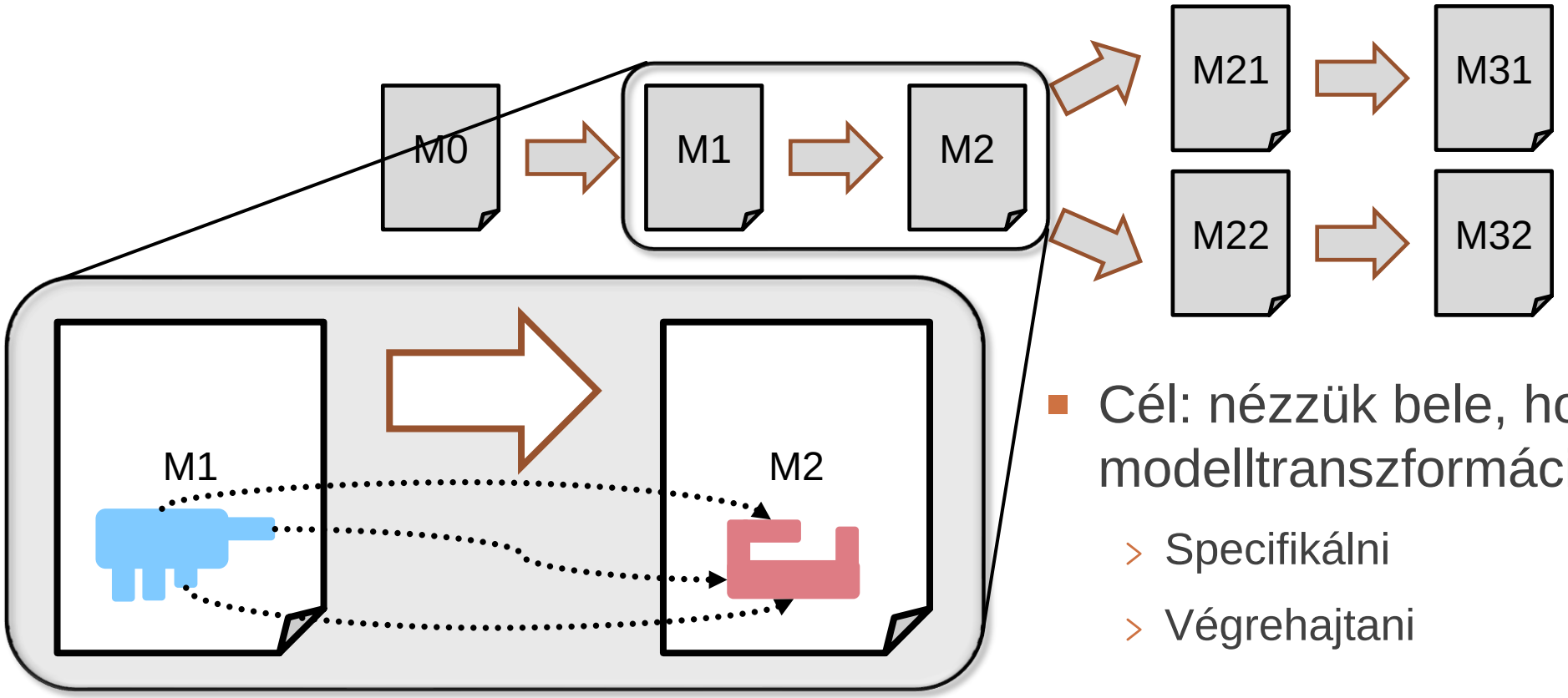
Dr. Semeráth Oszkár

Gráfmintaillesztés, Gráftranszformáció

- **Alapfogalmak**
- **Gráfmintaillesztés**
- **Modelltranszformációk**
- **Inkrementális transzformációk**
- **Tervezésitér bejárás**

Motiváció: Modellek transzformációja

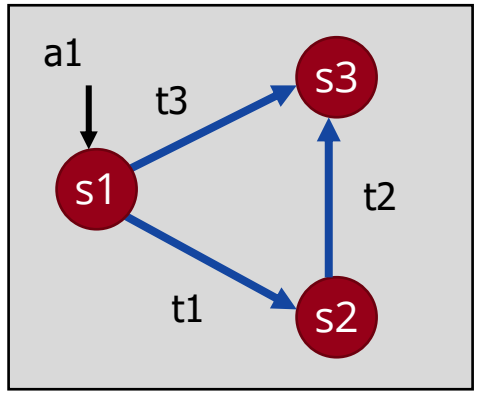
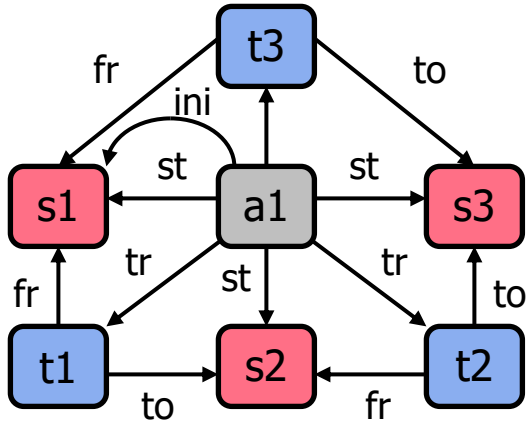
- **Modellalapú fejlesztés:** Modellek az elsődleges dokumentumok
- Modelleket fejlesztünk, automatizáljuk a modellfeldolgozást

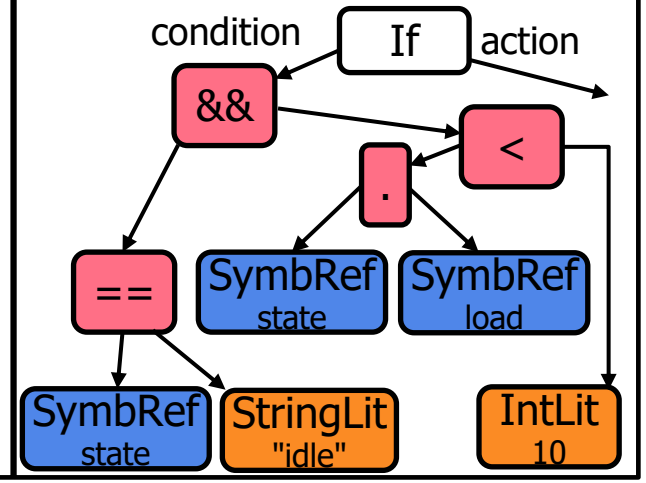


- Cél: nézzük bele, hogyan lehet modelltranszformációkat
  - Specifikálni
  - Végrehajtani

Absztrakt szintaxis

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

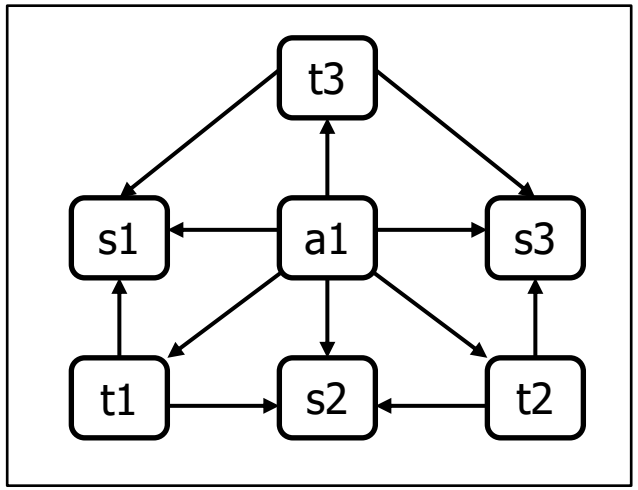
Konkrét	Absztrakt
	

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

- **Feladat:** gráfok módosítására módszer!

Hogyan írjuk le a modelleket gráffal?

- Irányított gráf: $G = \langle V_G, E_G \rangle$, ahol:
  - V_G: **csúcsok** halmaza (a G gráfban)
  - $E_G: V_G \times V_G \to \{true, false\}$: **élek** a csúcsok között (a G gráfban)
  - Hurokélek ☑, Párhuzamos élek ☑, Párhuzamos éle egy irányba ☒
- 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

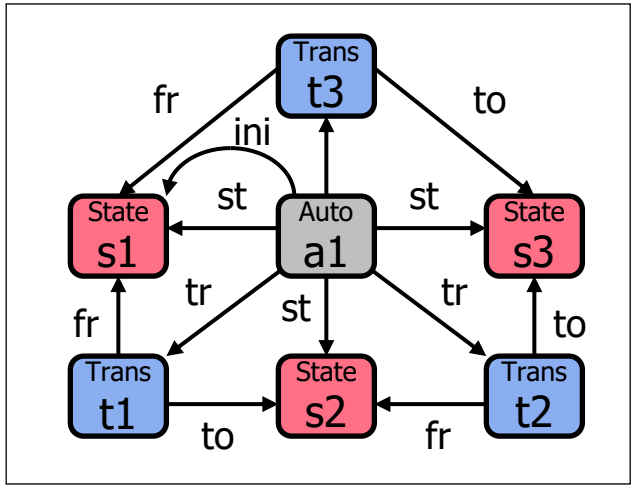
- Szótár: $\langle \Sigma, \alpha \rangle$
- Σ: Címkék halmaza

$$\Sigma = \{Automaton, State, Transition, states, transitions, from, to, init\}$$

- Aritás $\alpha: \Sigma \to \mathbb{N}$ meghatározza a címke szerepét
  - **Csúcs címke** $\alpha: Automaton, State, Transition \mapsto \mathbf{1}$

    Típusok, osztályok leírására
  - **Élcímke**: $\alpha: states, transitions, from, to, init \mapsto \mathbf{2}$

    Attribútumok, referenciák leírására
  - Lehet több is (3+), most nekünk nem fontos



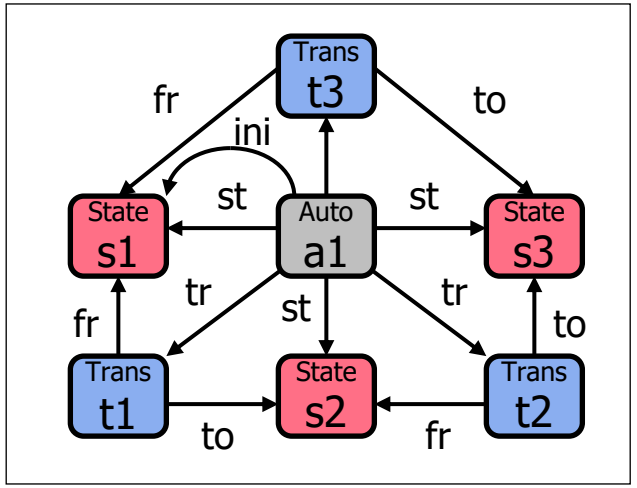
Címkekészlettel címkézett gráf

- 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

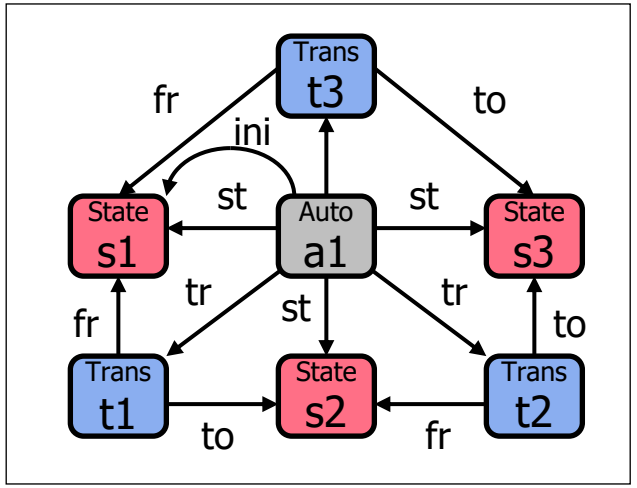
I_M meghatározza, hogy:

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



Példa címkézett gráf (modell)

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

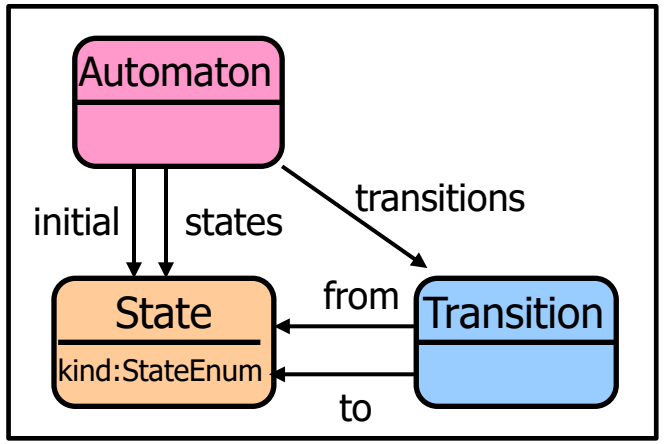


Gráfmintaillesztés, Gráftranszformáció

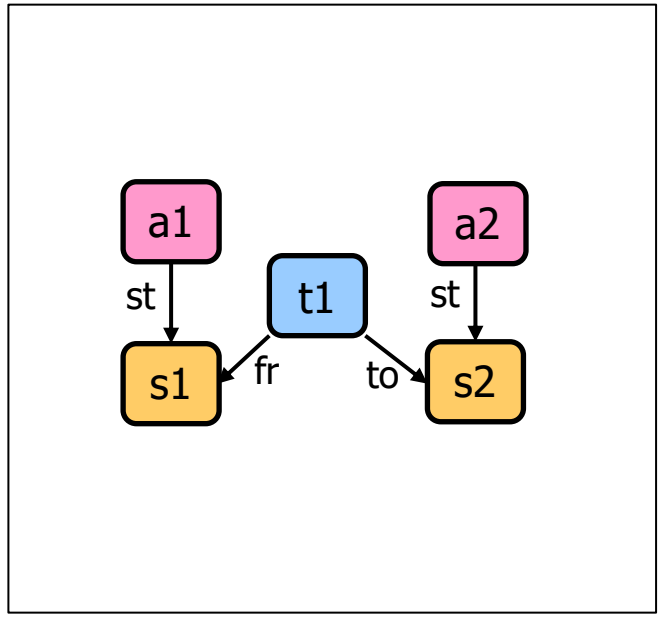
- Alapfogalmak
- **Gráfmintaillesztés**
- Modelltranszformációk
- Inkrementális transzformációk
- Tervezésitér bejárás

Egy egyszerű példa

Metamodell



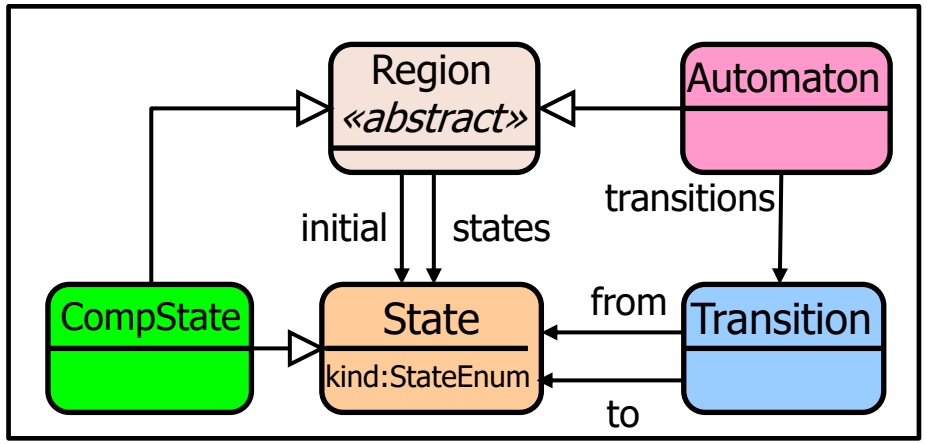
Szabálysértés példa



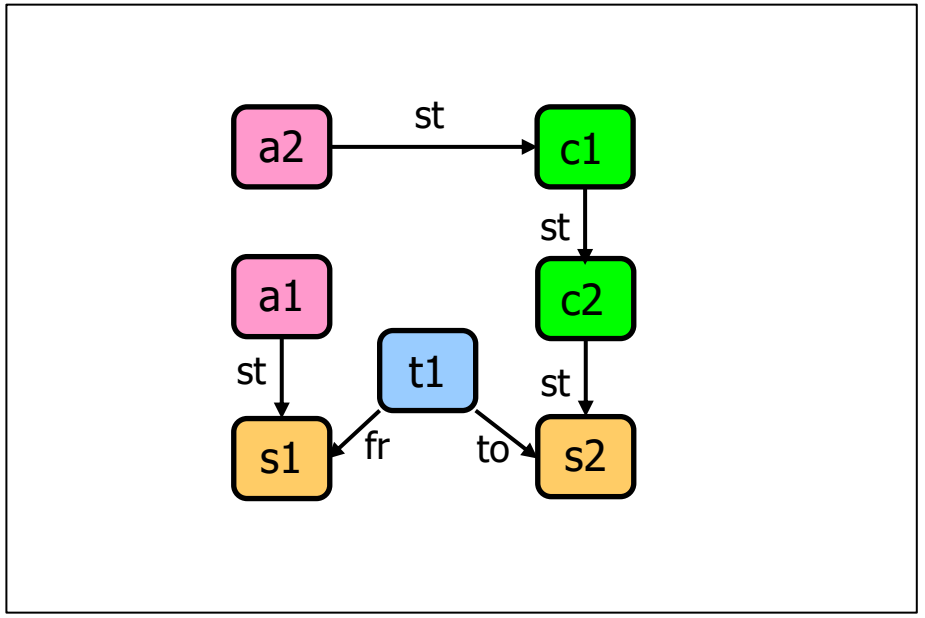
- Jólformáltsági kényszer:
  - Tranzíció forrás- és célállapotának ugyanabban az állapotgépben kell lennie
- Cél: szabálysértések megtalálása...
  - A szabályszegés egy *Tranzíció*, aminek „*from*” éle egy *s1 State*-re mutat, és „*to*” éle egy *s2 State*-re mutat, ahol *s1 állapotgépe* nem *s2* állapotgépe

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 : automatons) {
  for (Transition transition : automaton.getTransitions()) {
    State sourceState = transition.from;
    // melyik automaton definiálja ezt az állapotot?
    Automaton sourceAutomaton = null;
    for (Automaton candidate : automatons) {
      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
  - …

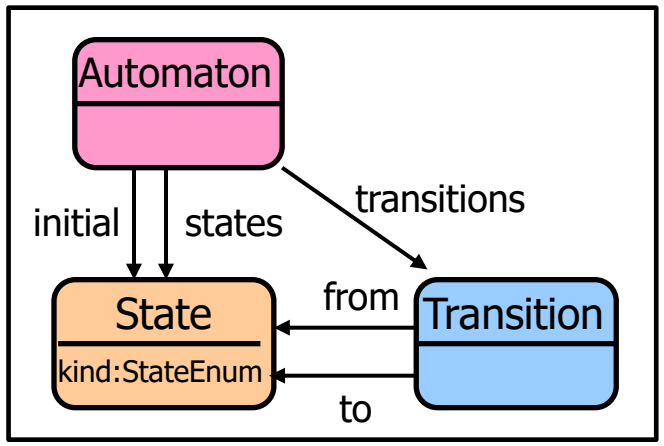
Lekérdezési nyelvi stílusok

- 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

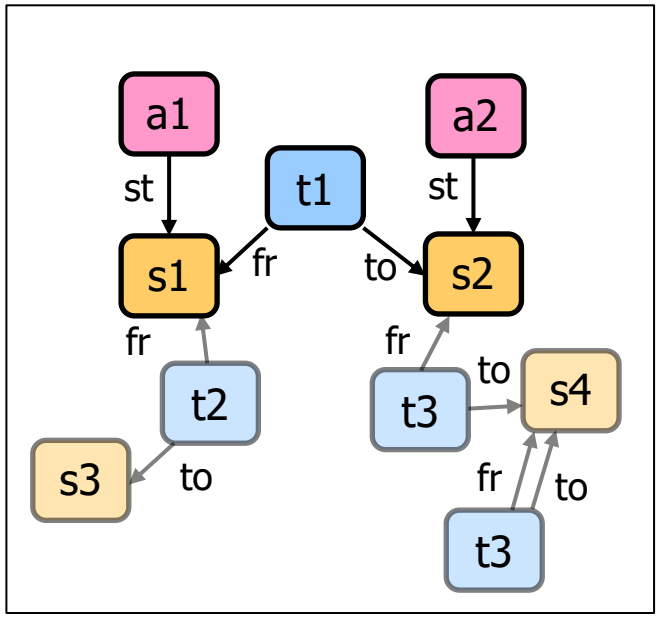


A szabályszegés egy Tranzíció, aminek „from” éle egy s1 State-re mutat, és „to” éle egy s2 State-re mutat, ahol s1 állapotgépe nem s2 állapotgépe

Formális logikával (Domain Relációs Kalkulus)

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

Szabálysértés példa

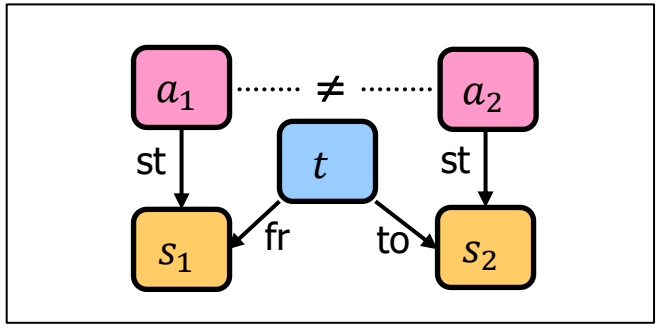



```
violates(t):-
Transition(t),from(t,s1),to(t,s2),
states(a1,s1),states(a2,s2),a1=\=a2
```

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 (*$Q(t)$*)*

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

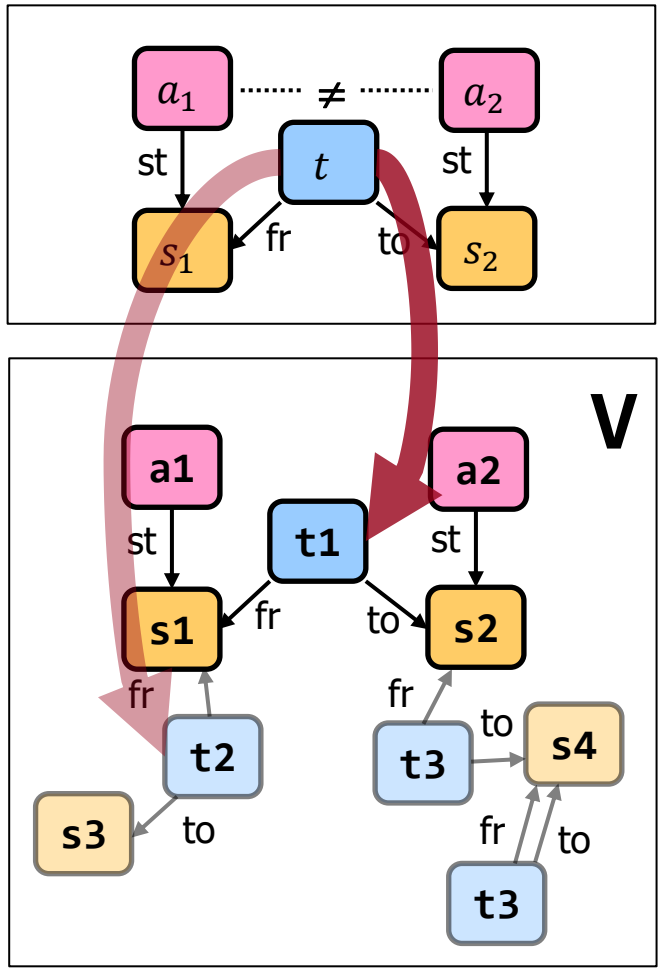
  - **Modell** = *a lekérdezés kiértékelésének célpontja (***V***)*
  - **Lekötés** = *kényszerváltozókat a modell elemeihez köti*

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

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

$$[\![Q(t)]\!]^{V}_{Z_1} = \boldsymbol{true}$$

lekérdezés — *modell* — *lekötés* — *Eredmény*



Lekérdezés

Szabálysértés példa

Lekérdezés kiértékelése

- **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 $[\![Q]\!]_Z^M$ definíciójára
- A lekérdezőmotor támogatja
  - a modell-lekérdezések definiálását,
  - lekérdezés-illesztést:

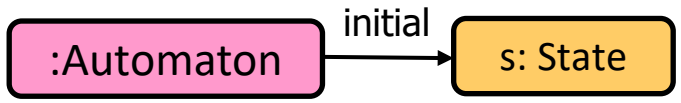
M modellre és Q lekérdezésre, visszaad minden Z lekötést, ahol $[\![Q]\!]_Z^M = true$.

≈ DB query

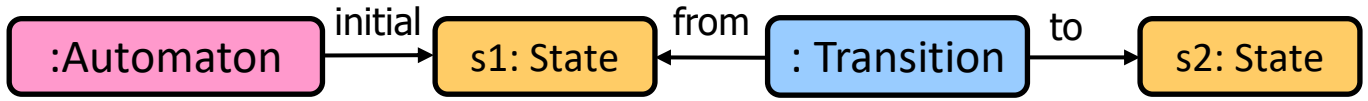
$$
\begin{aligned}
[\![\mathtt{C}(v)]\!]_Z^M &:= \mathcal{I}_M(\mathtt{C})(Z(v)) \\
[\![\mathtt{R}(v_1, v_2)]\!]_Z^M &:= \mathcal{I}_M(\mathtt{R})(Z(v_1), Z(v_2)) \\
[\![v_1 = v_2]\!]_Z^M &:= Z(v_1) = Z(v_2) \\
[\![\varphi_1 \wedge \varphi_2]\!]_Z^M &:= [\![\varphi_1]\!]_Z^M \wedge [\![\varphi_2]\!]_Z^M \\
[\![\varphi_1 \vee \varphi_2]\!]_Z^M &:= [\![\varphi_1]\!]_Z^M \vee [\![\varphi_2]\!]_Z^M \\
[\![\neg\varphi]\!]_Z^M &:= \neg [\![\varphi]\!]_Z^M \\
[\![\forall v : \varphi]\!]_Z^M &:= \bigwedge_{x \in Obj_M} [\![\varphi]\!]_{Z, v \mapsto x}^M \\
[\![\exists v : \varphi]\!]_Z^M &:= \bigvee_{x \in Obj_M} [\![\varphi]\!]_{Z, v \mapsto x}^M
\end{aligned}
$$

Példák

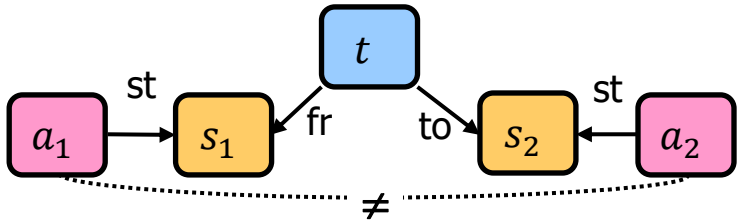
- **Egyszerű példa:** kezdőállapotok a modellben



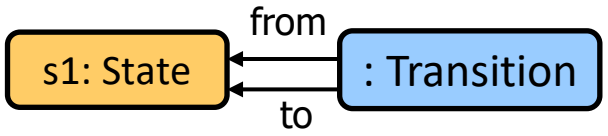
- **Láncolat (∧):** Második állapotok a modellben



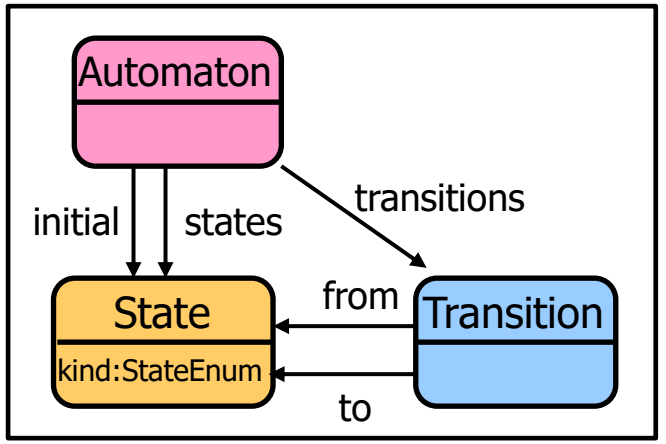
- **≠ :** Automatákon átívelő tranzíció



- =: Hurolél

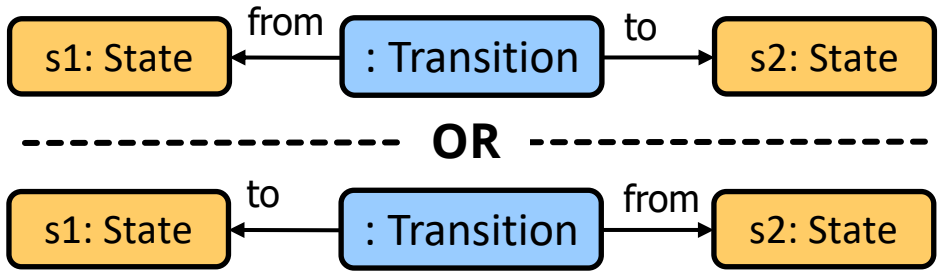


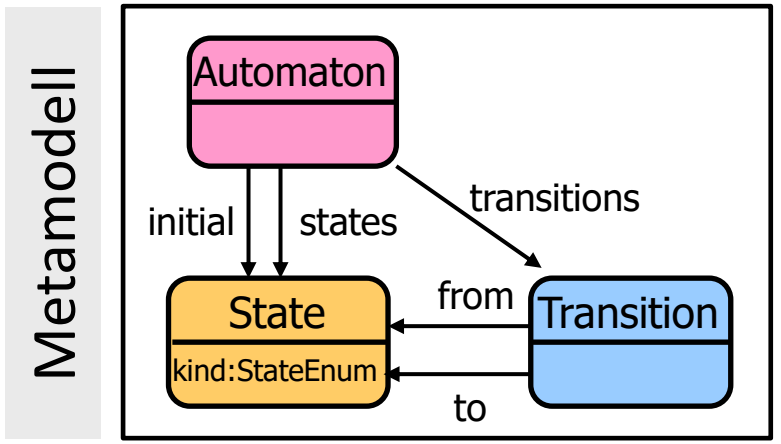
Metamodell



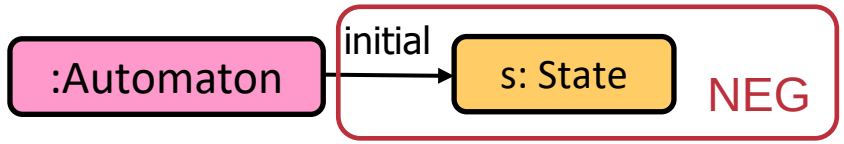
Példák 2

- (∨): Két állapot össze van kötve

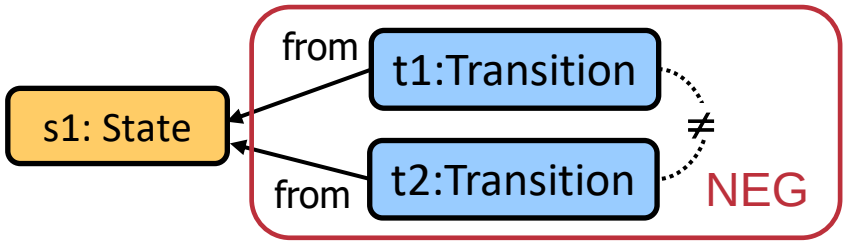




- (¬, Negative Application Condition):
  - kezdőállapot nélküli automata



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

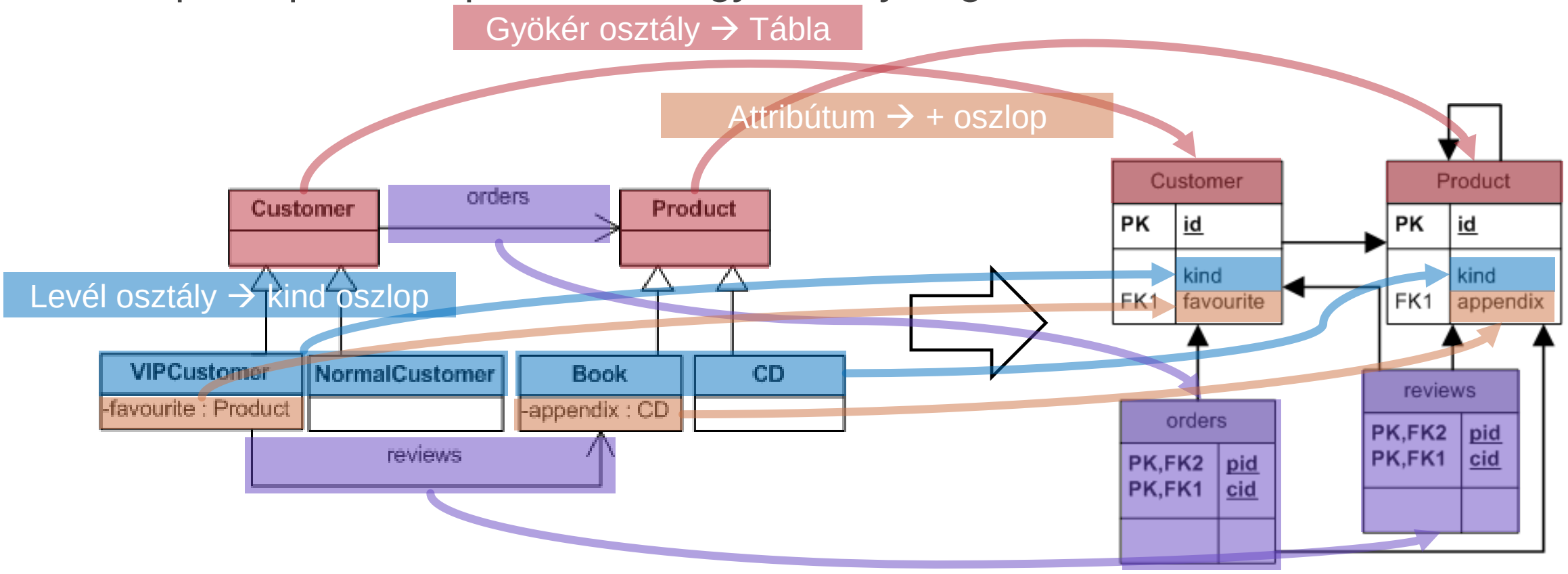


Gráfmintaillesztés, Gráftranszformáció

- Alapfogalmak
- Gráfmintaillesztés
- **Modelltranszformációk**
- Inkrementális transzformációk
- Tervezésitér bejárás

Példa Transzformáció

- Tipikus példa: képezzünk le egy osztálydiagramot adatbázis táblákra!

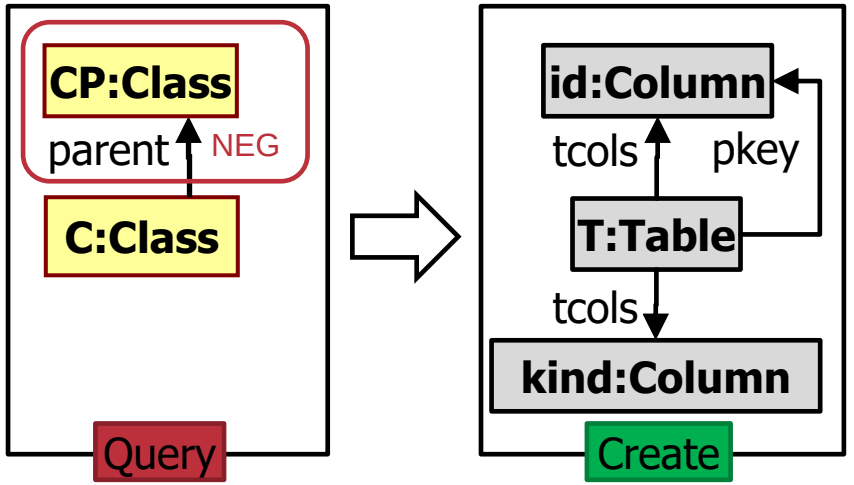


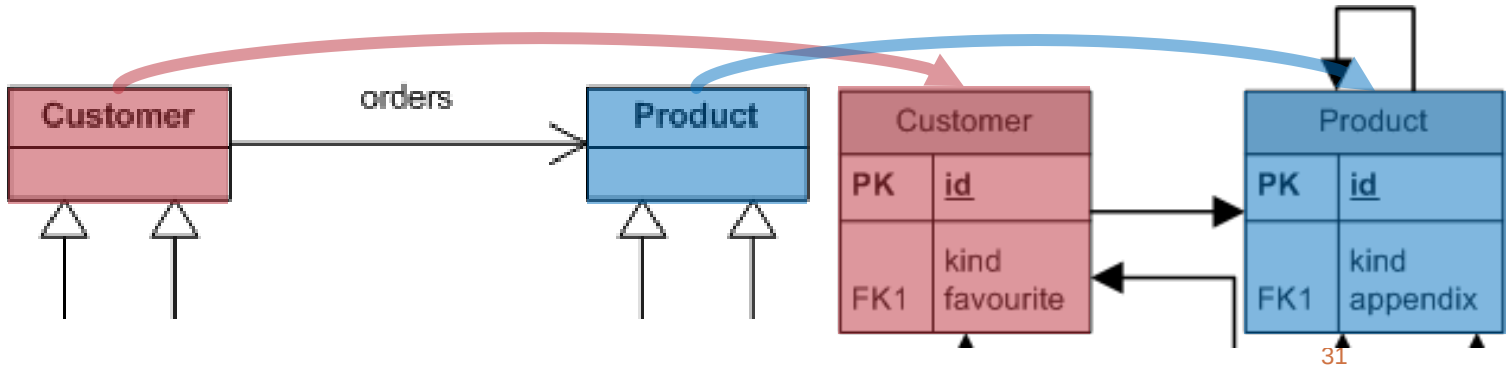
Példa Transzformáció

- Hogyan oldanánk a gyökér osztályokat reprezentáló táblák létrehozását?

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

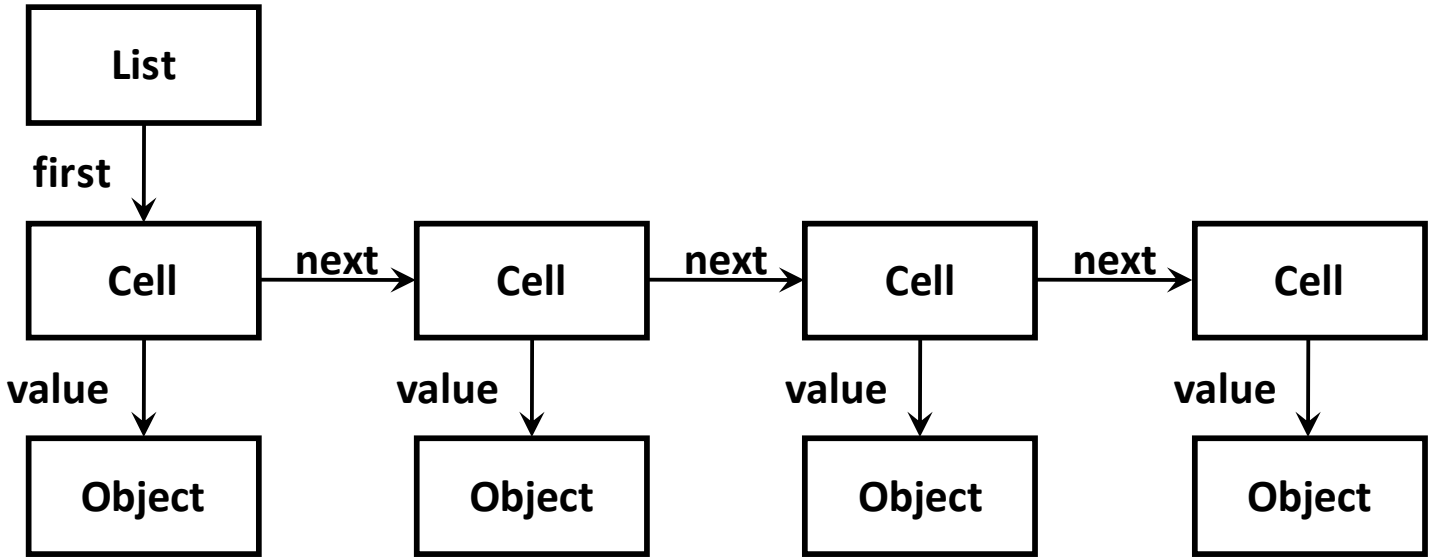
- Cél: Hasonló szabályokkal megfogalmazni az egész transzformációt





Gráftranszformáció

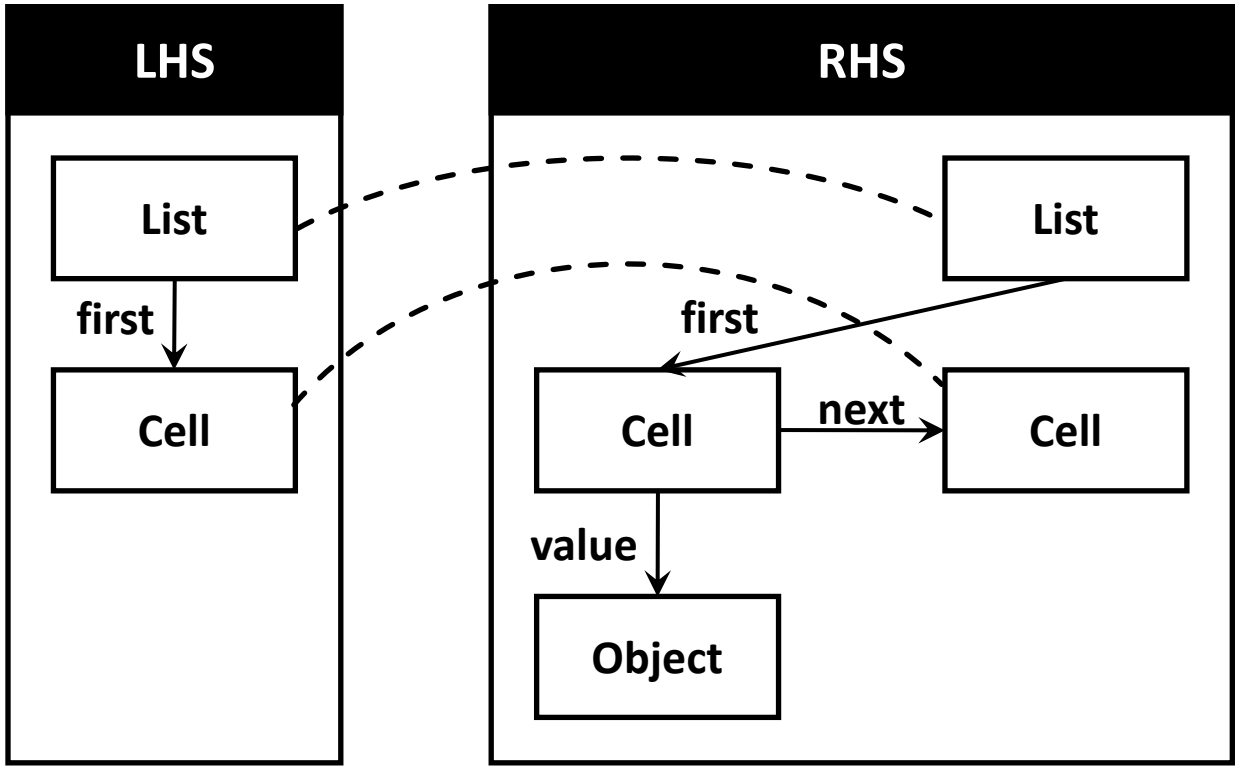
- Modell = Címkézett gráf



Gráftranszformációs szabály

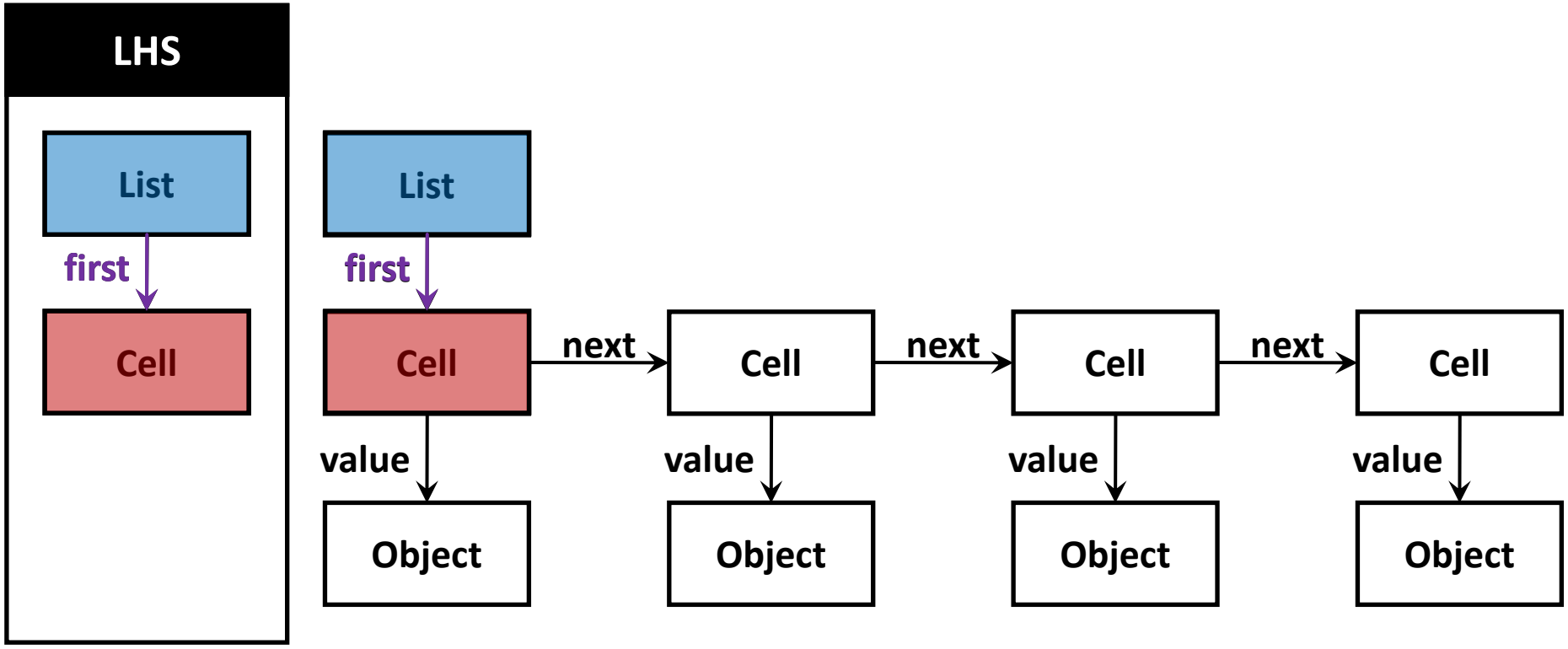
- Gráf átírási szabály, két gráffal van megfogalmazva

LHS = bal oldal, RHS = jobb oldal



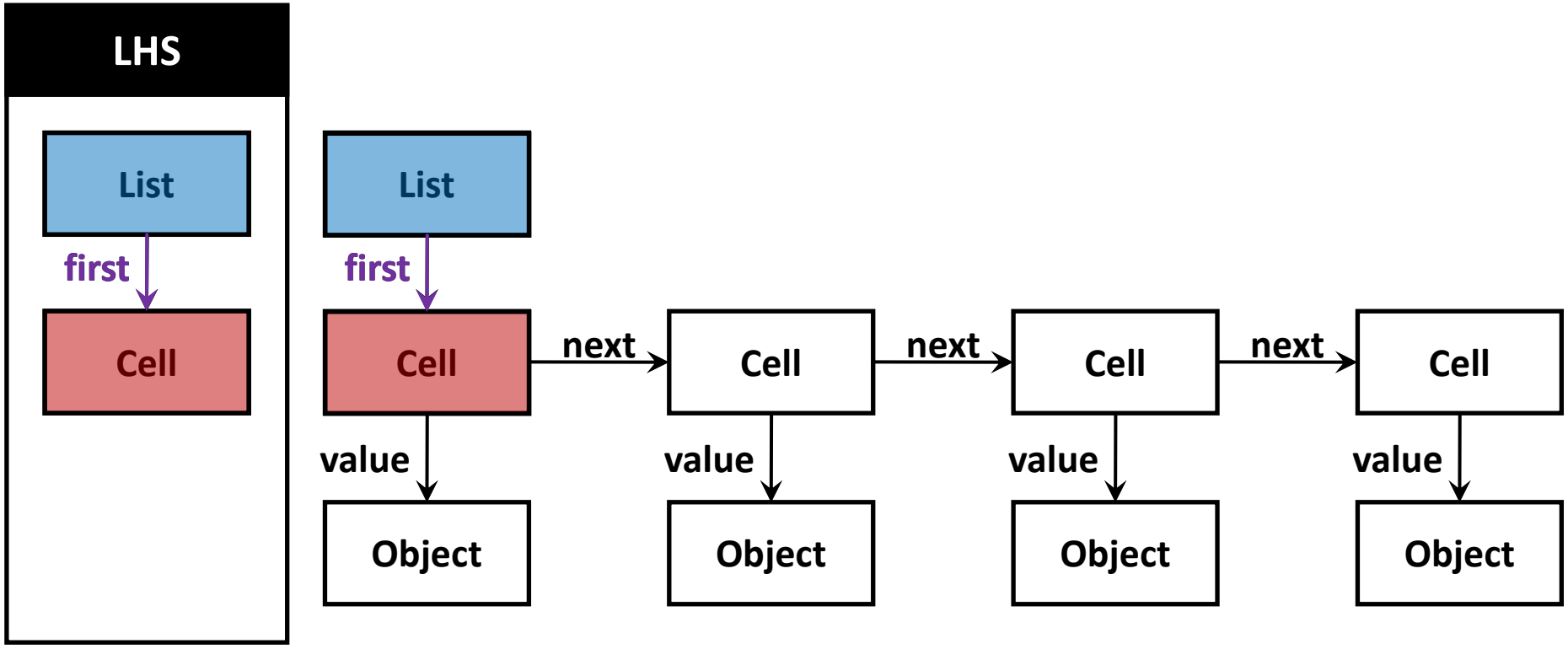
Gráftranszformáció: Mintaillesztés

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



Gráftranszformáció: Mintaillesztés

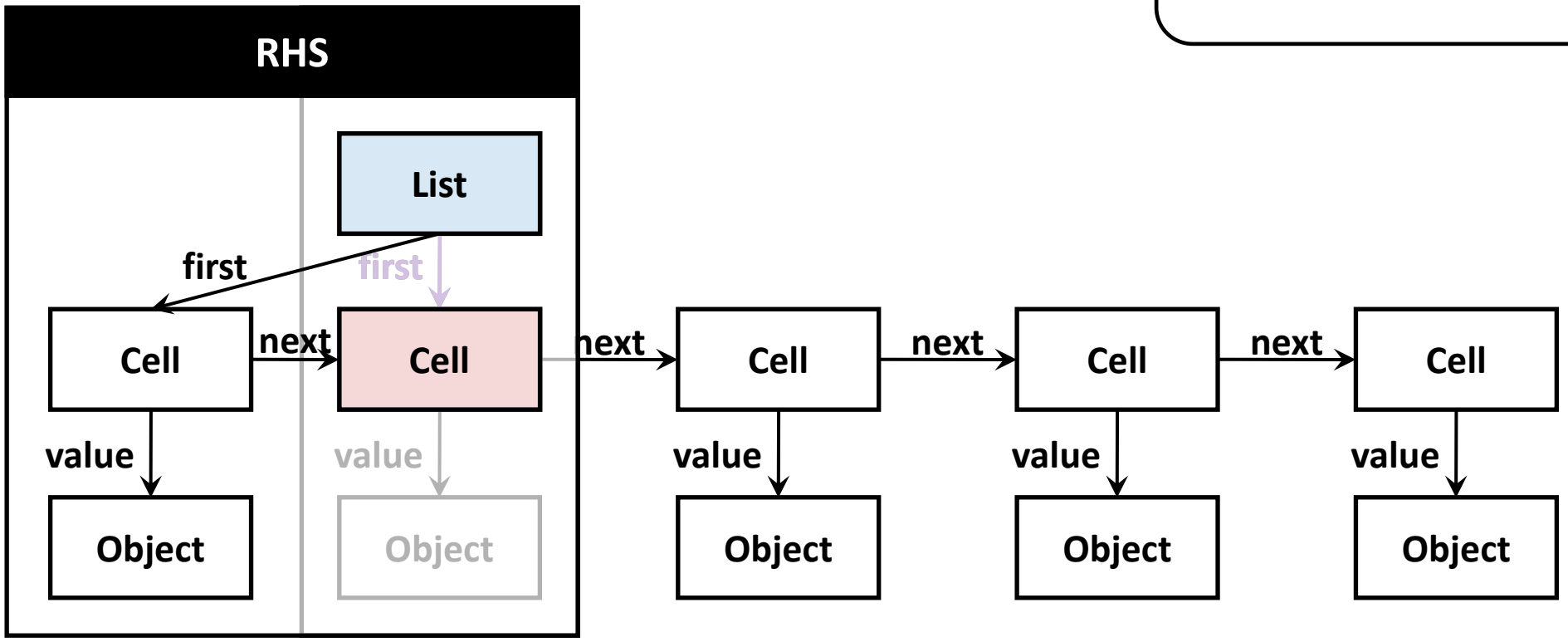
- **Illesztés:** megkeressük a LHS-t tartalmazó részgráfokat a forrás gráfban



Gráftranszformáció: Átírás végrehajtása

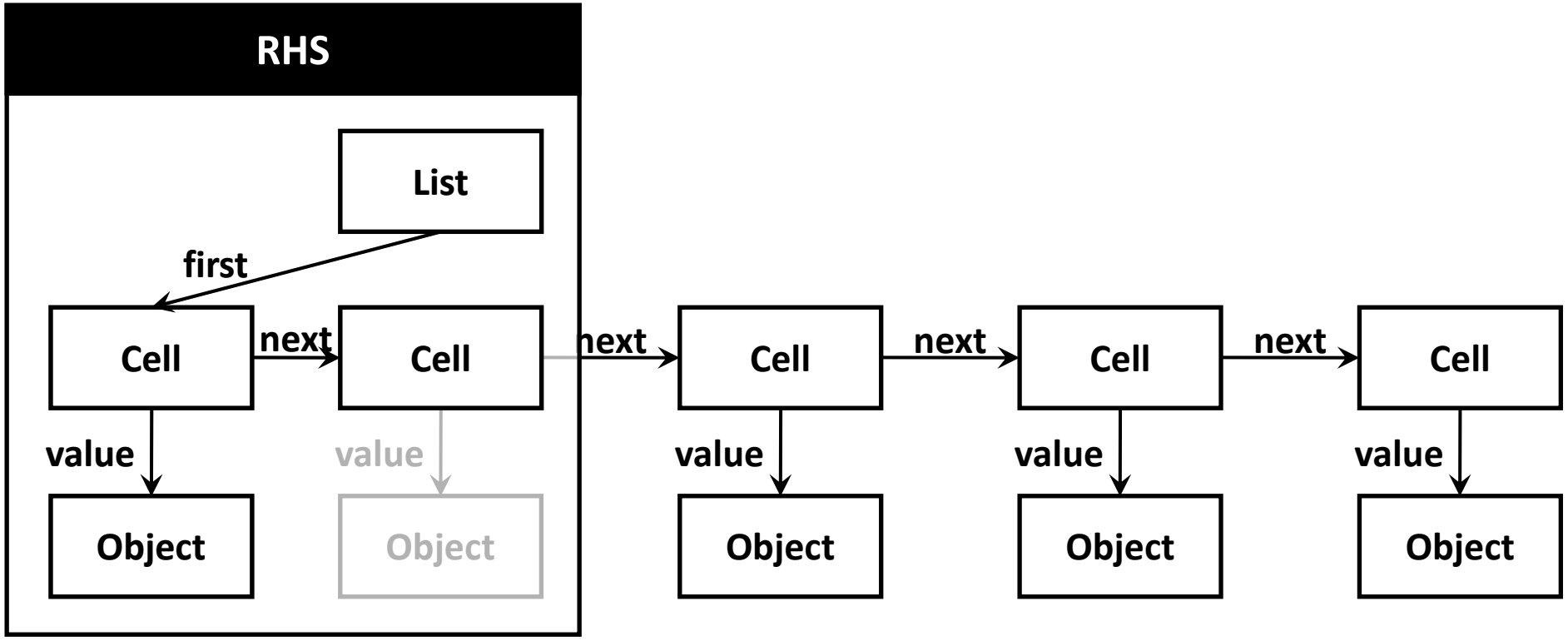
- Illesztés mentén lecseréljük az LHS-t RHS-re.

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



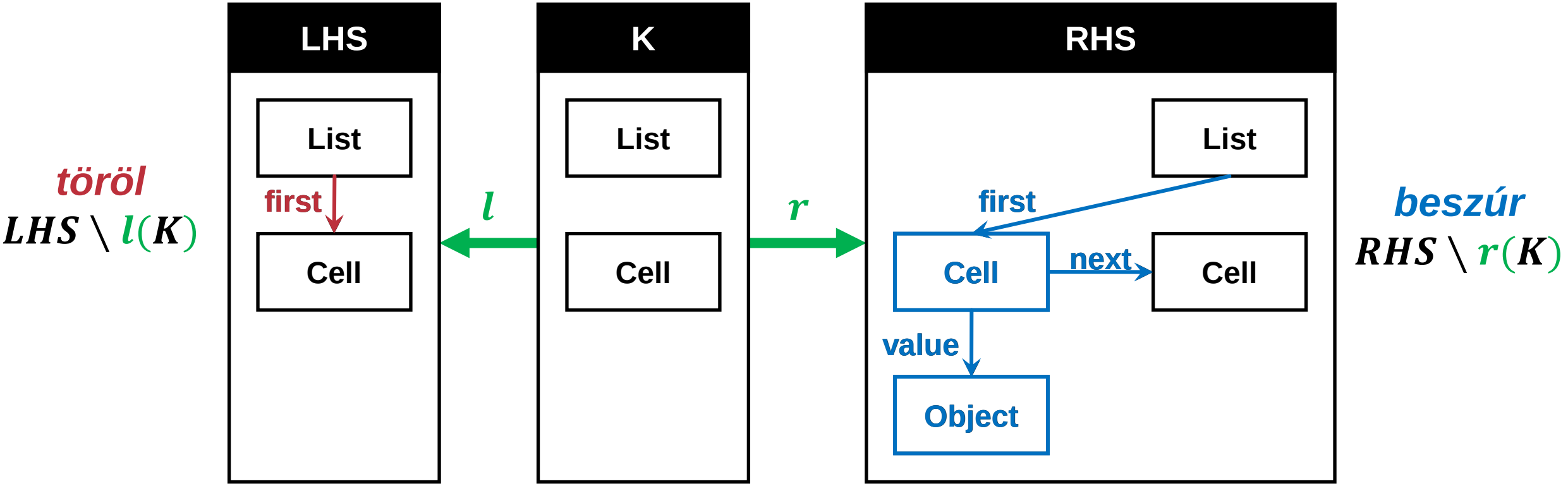
Gráftranszformáció: Átírás végrehajtása

- Új gráfot kapunk

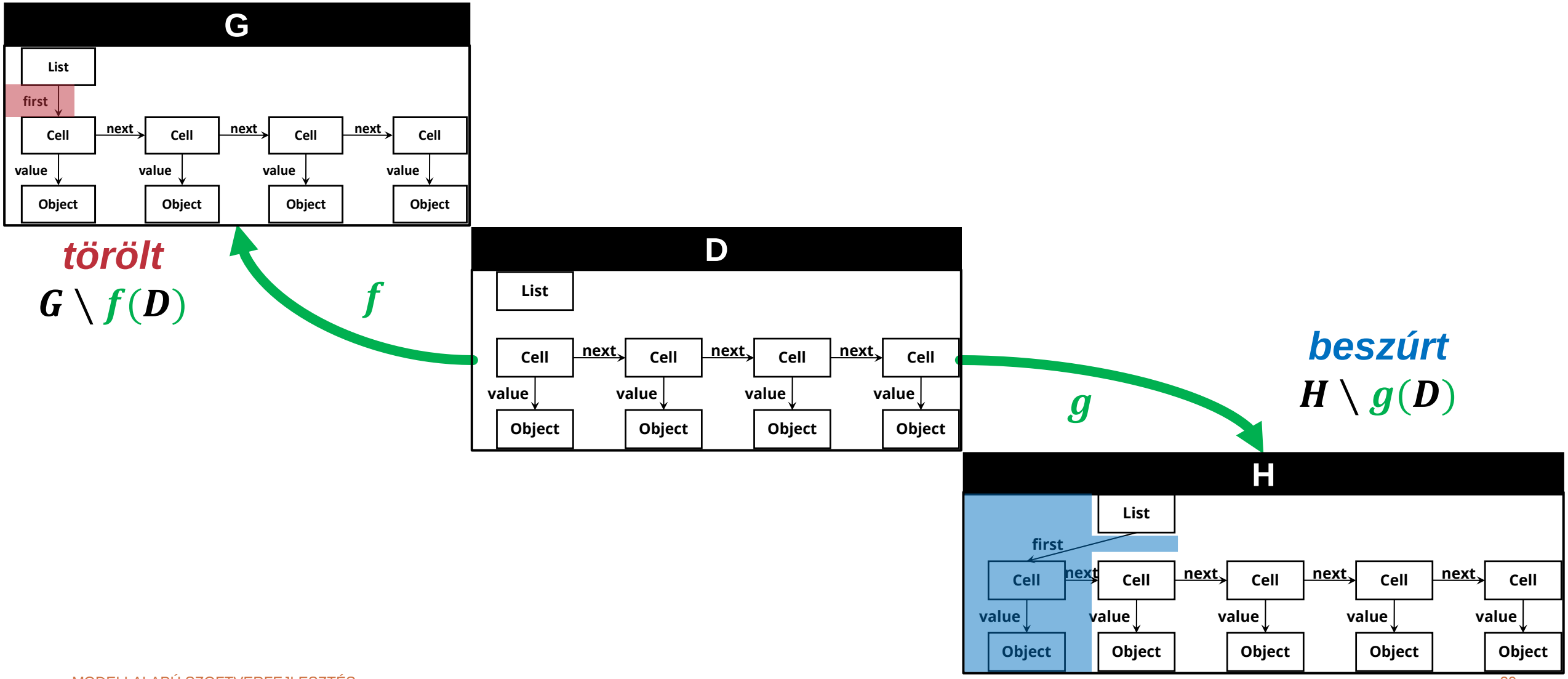


Gráftranszformációk anatómiája

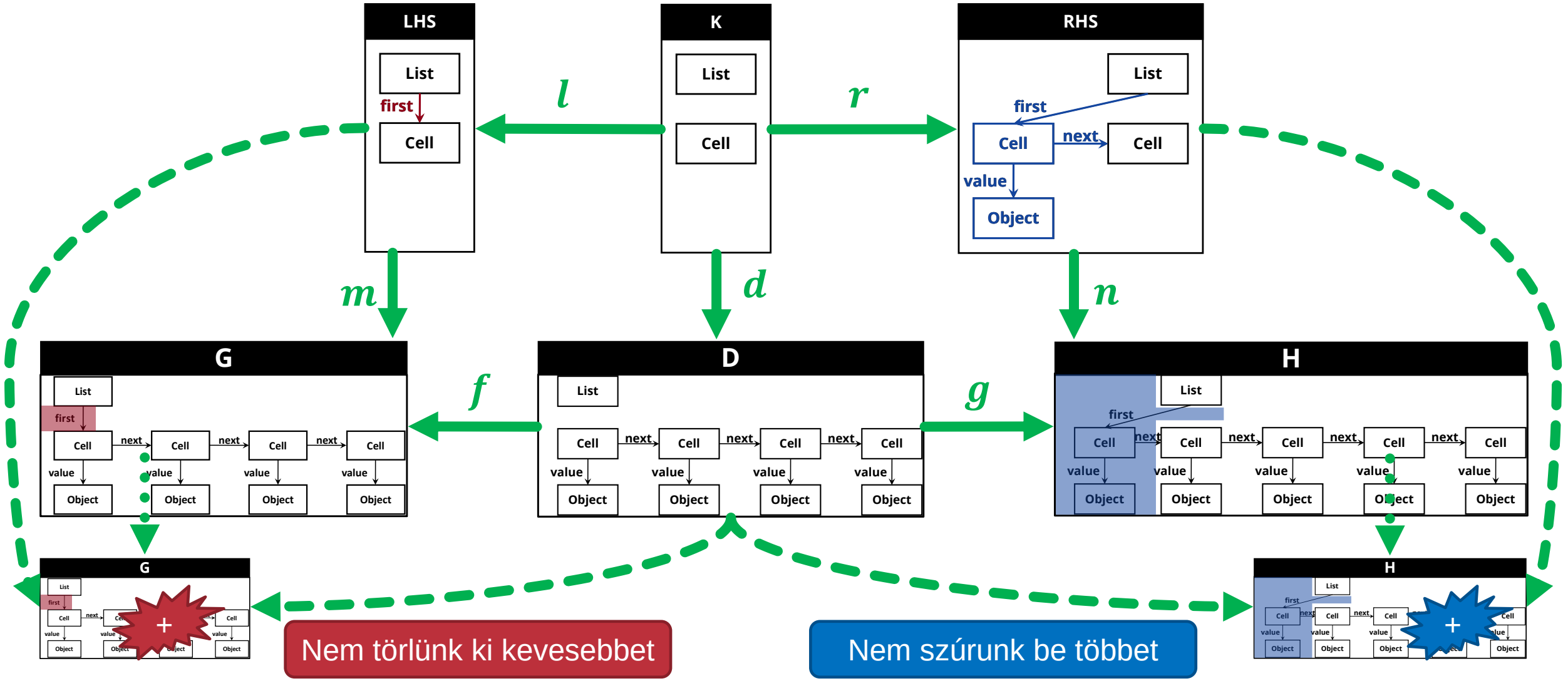
- Vizsgáljuk meg, hogy a transzformáció során mely gráf illeszthető melyikbe!



Gráftranszformációk anatómiája



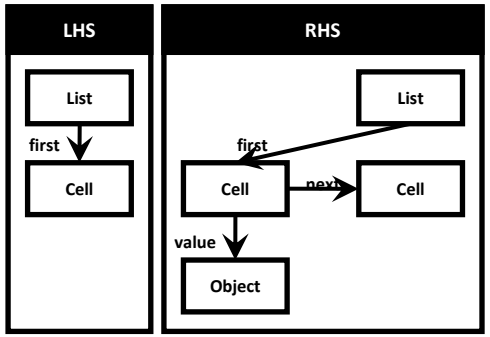
Teljes anatómia



Gráftranszformáció

- Szabályok megfogalmazása modellek átírására
- Nyelvtani szabályok kiterjesztése

List → List,Cell **vs**



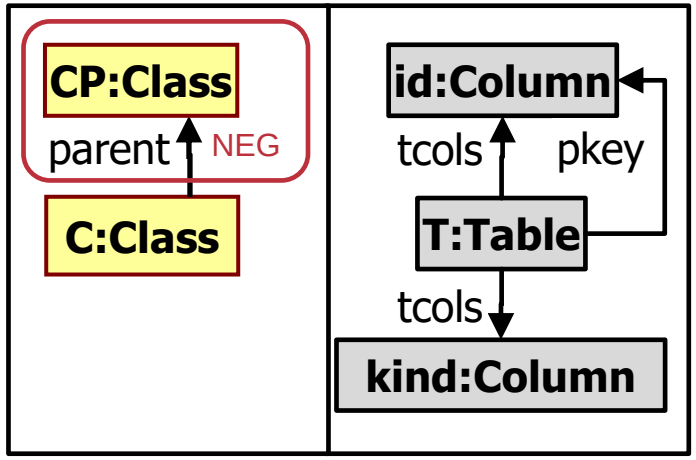
- Szemléletes, de matematikailag precíz formalizmus

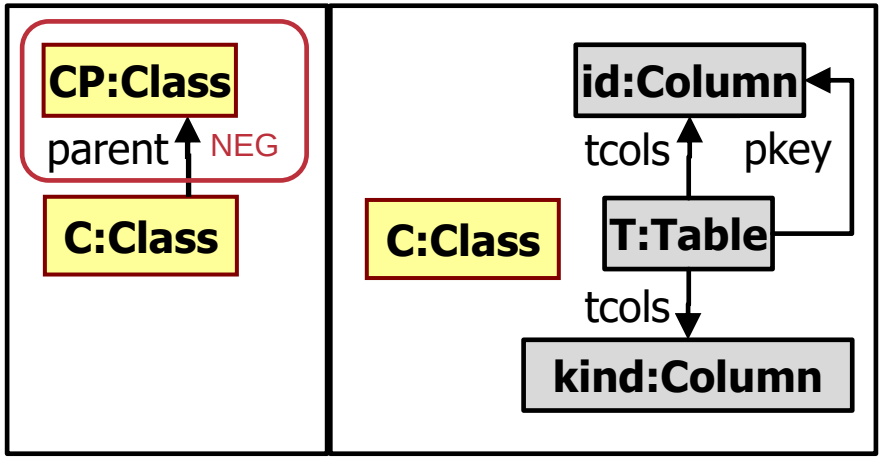
  (Terminálódás, Sorrendezés, Konfluencia, …)

- Eszköztámogatás (lásd előző gyakorlat)

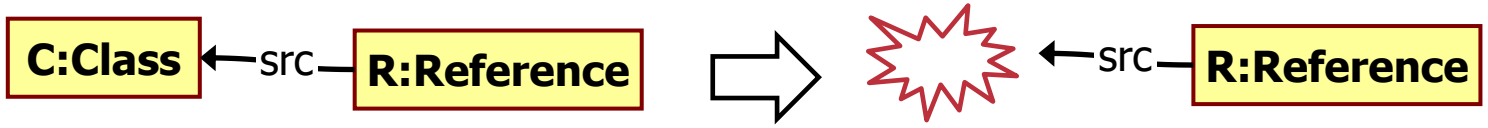
Példák

- Ősosztályok leképezése (törléssel és törlés nélkül)





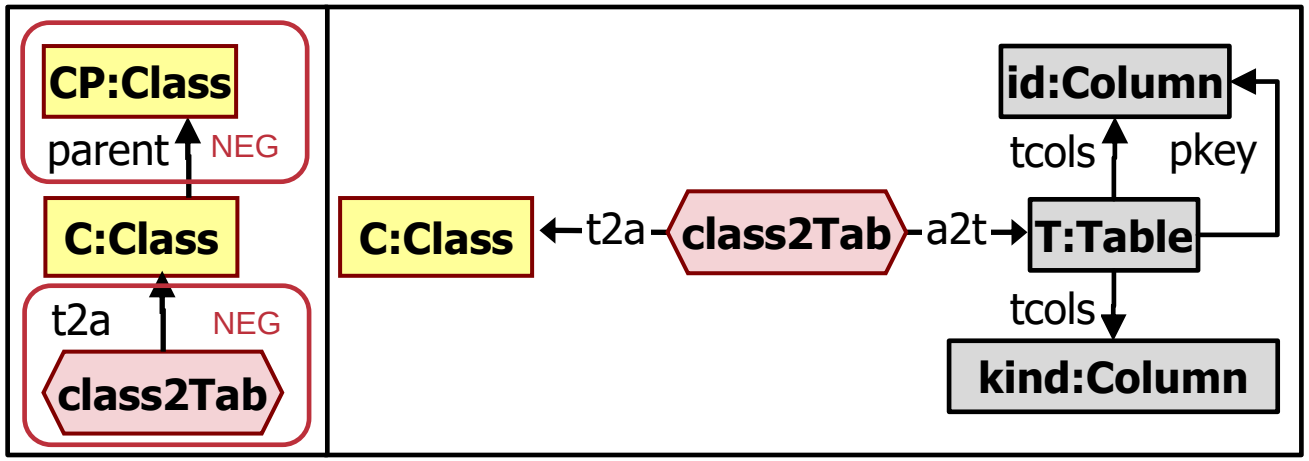
- Mi történik, ha kitörlünk egy elemet, amire mutat még él?



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

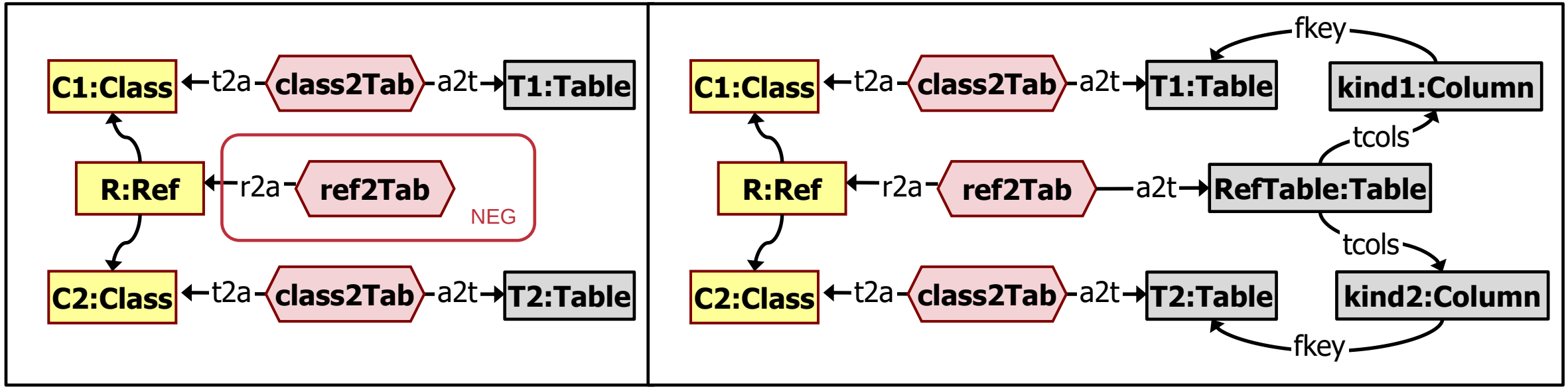
Példák

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



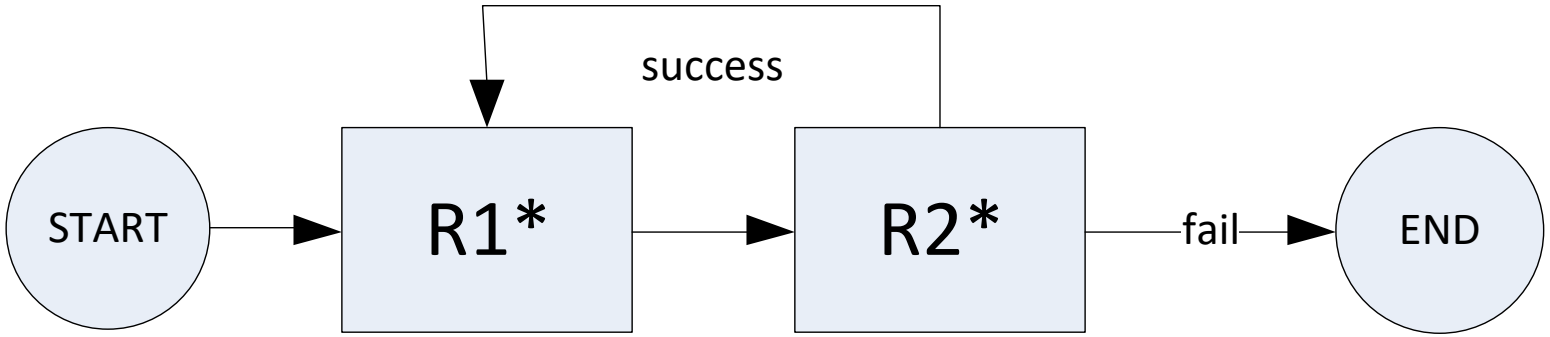
Példák

- Referenciák leképezése

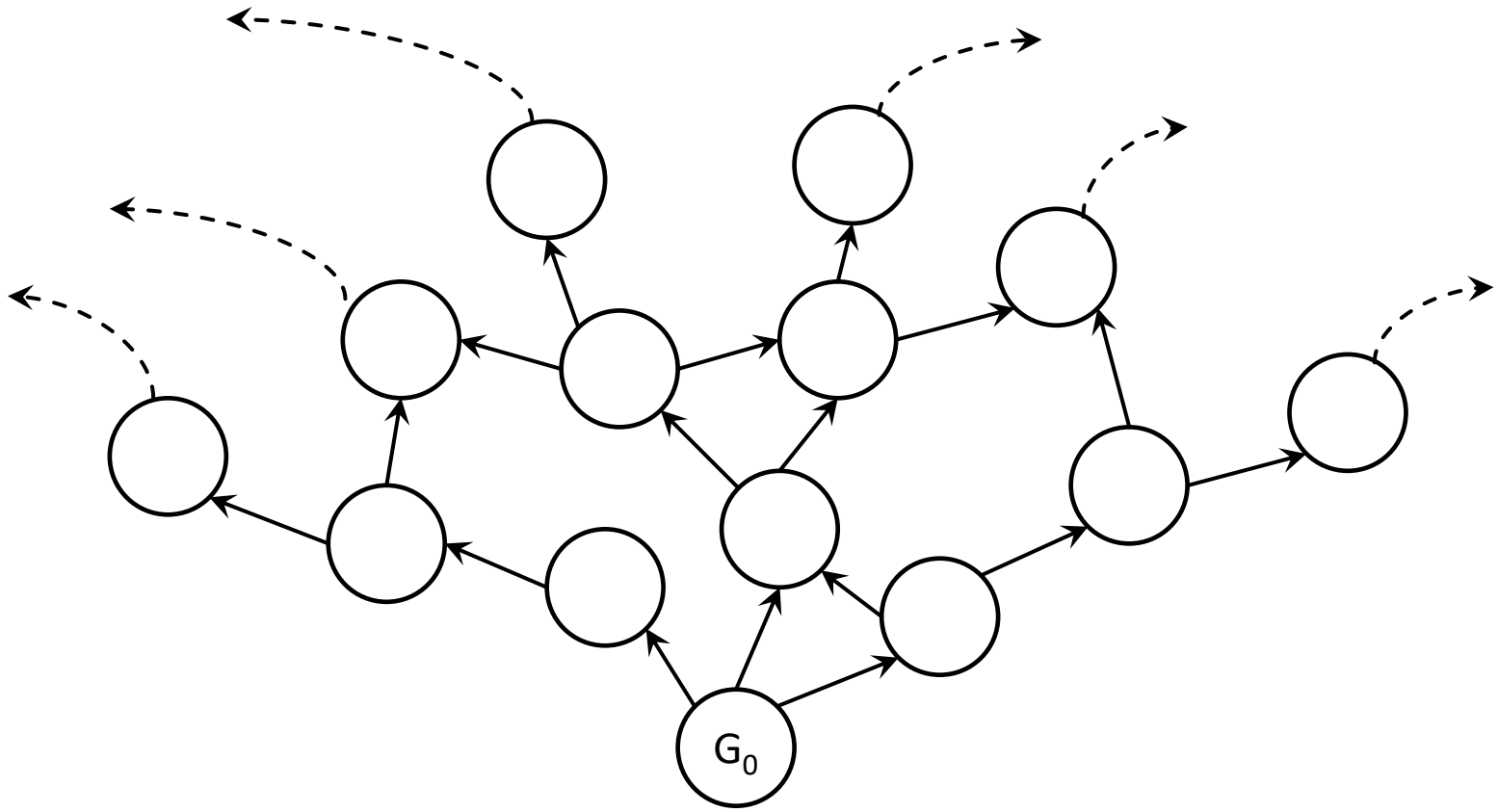


Vezérlési szerkezetek

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

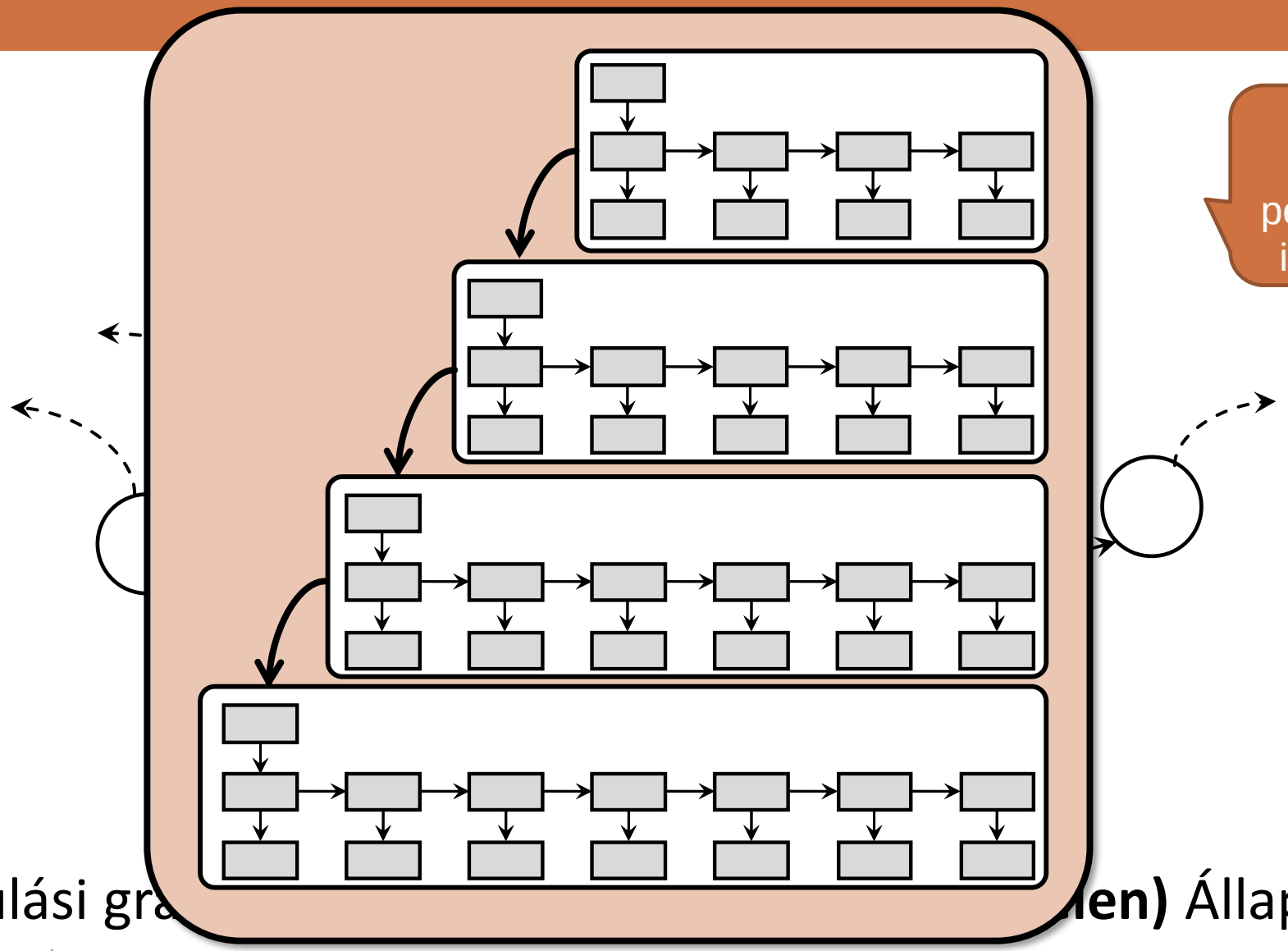


Állapottér



Kiindulási gráf **+** GT szabályok → **(tipikusan végtelen)** Állapottér

Állapottér



Még egyszerű példa esetén is végtelen

Kiindulási gra**len)** Állapottér

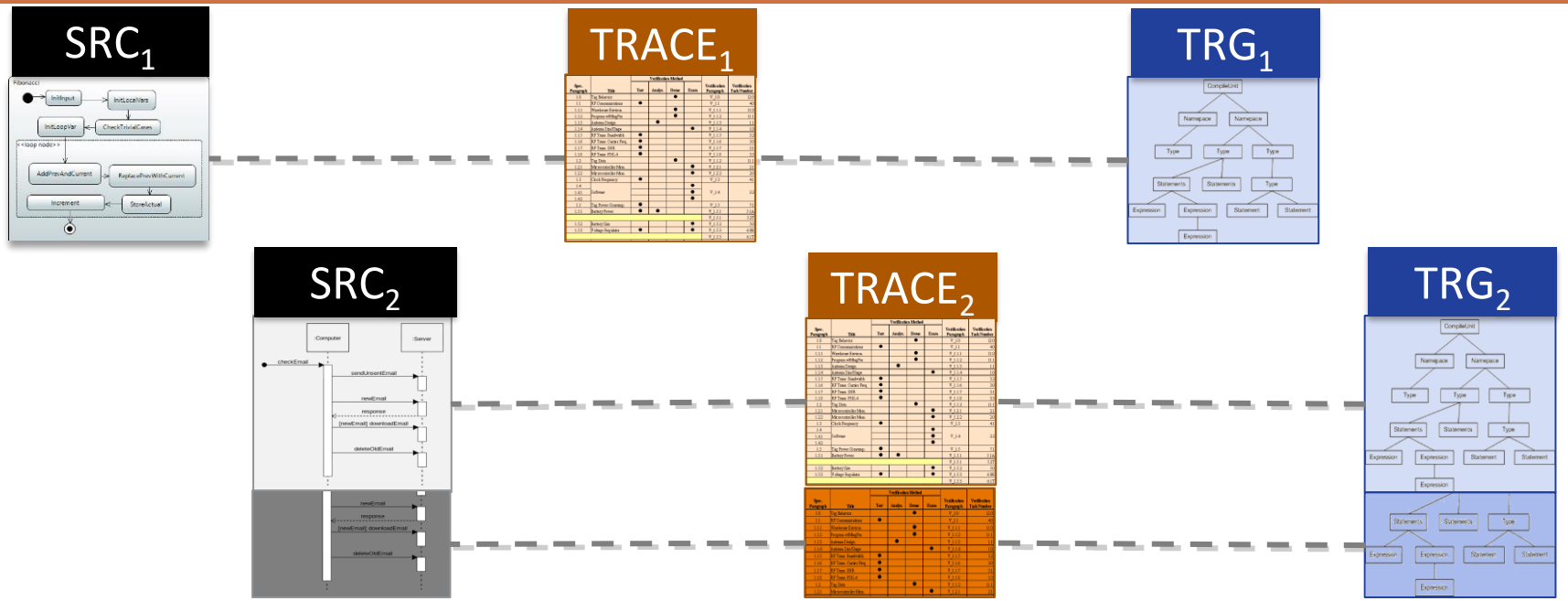
Modeltranszformációk típusai

- A bemenetek és kimentek száma
  (In-place vs out-place)
- A nyelv szerint
  (Endogenous vs exogenous)
- Az irány szerint
  (unidirectional vs bidirectional)

Gráfmintaillesztés, Gráftranszformáció

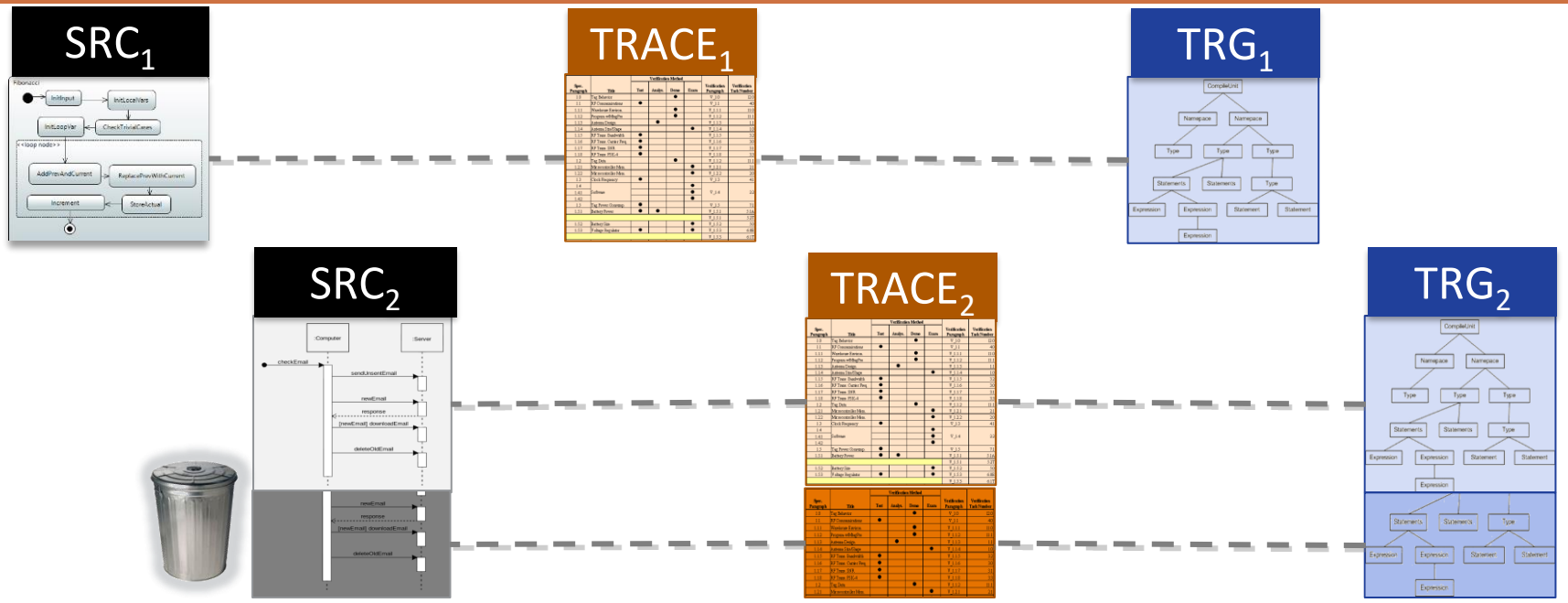
- Alapfogalmak
- Gráfmintaillesztés
- Modelltranszformációk
- **Inkrementális transzformációk**
- Tervezésitér bejárás

Inkrementális végrehajtás: 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

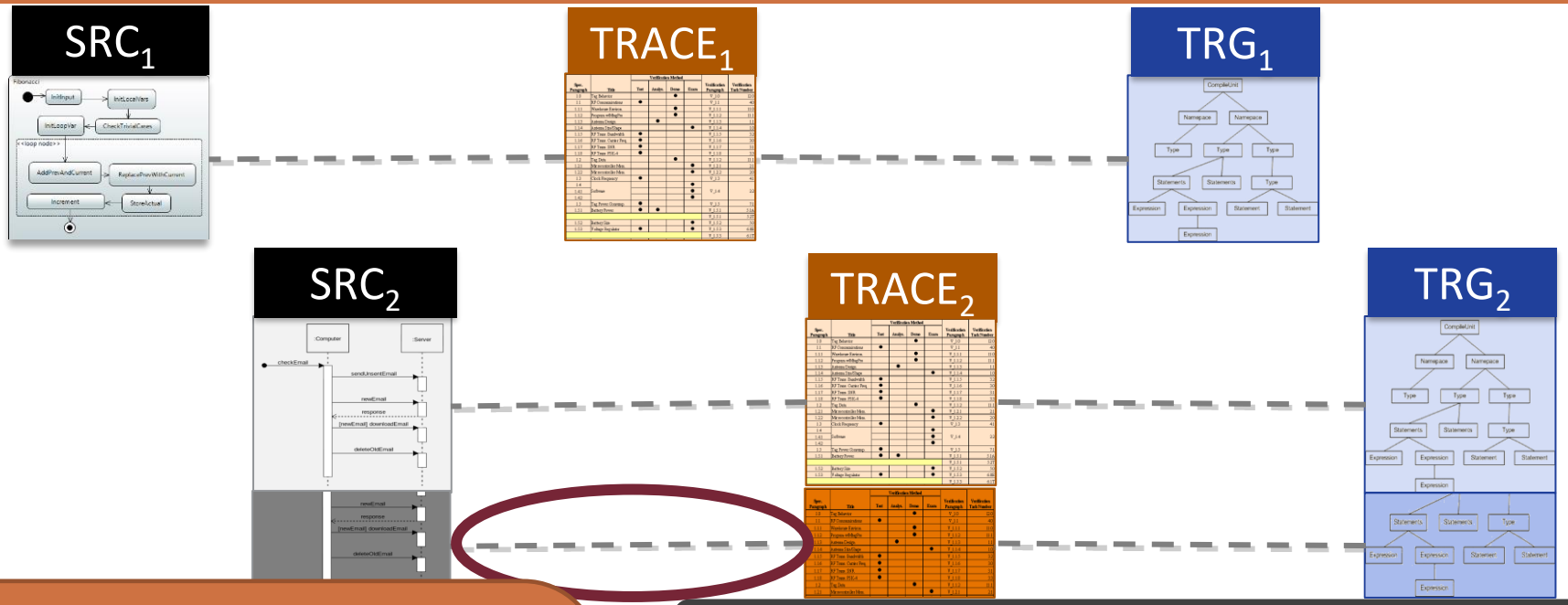
Hátrányok:
- Összetett MT lassú lehet
- Tisztítás (hiba után)?
- Láncolás?

1. Első transzformáció

2. Forrásmodell megváltozik

3. Újrafuttajuk az elejétől csak a módosított modellekre

Inkrementalitás nyomonkövethetőséggel

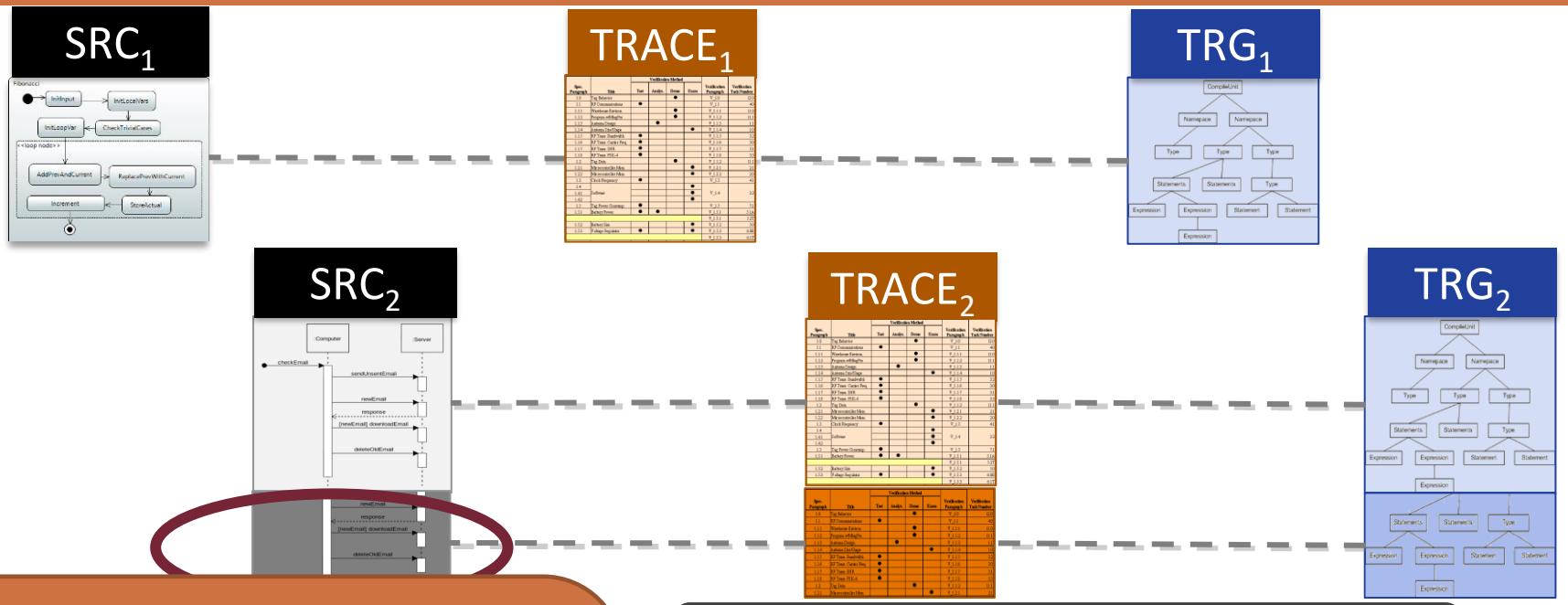


Előnyök:
- Kis lépésenkénti inkrementalitás
- Jobb teljesítmény

Hátrányok:
- Nagymértékben függ a nyomonkövethetőségi kapcsolatoktól
- 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"

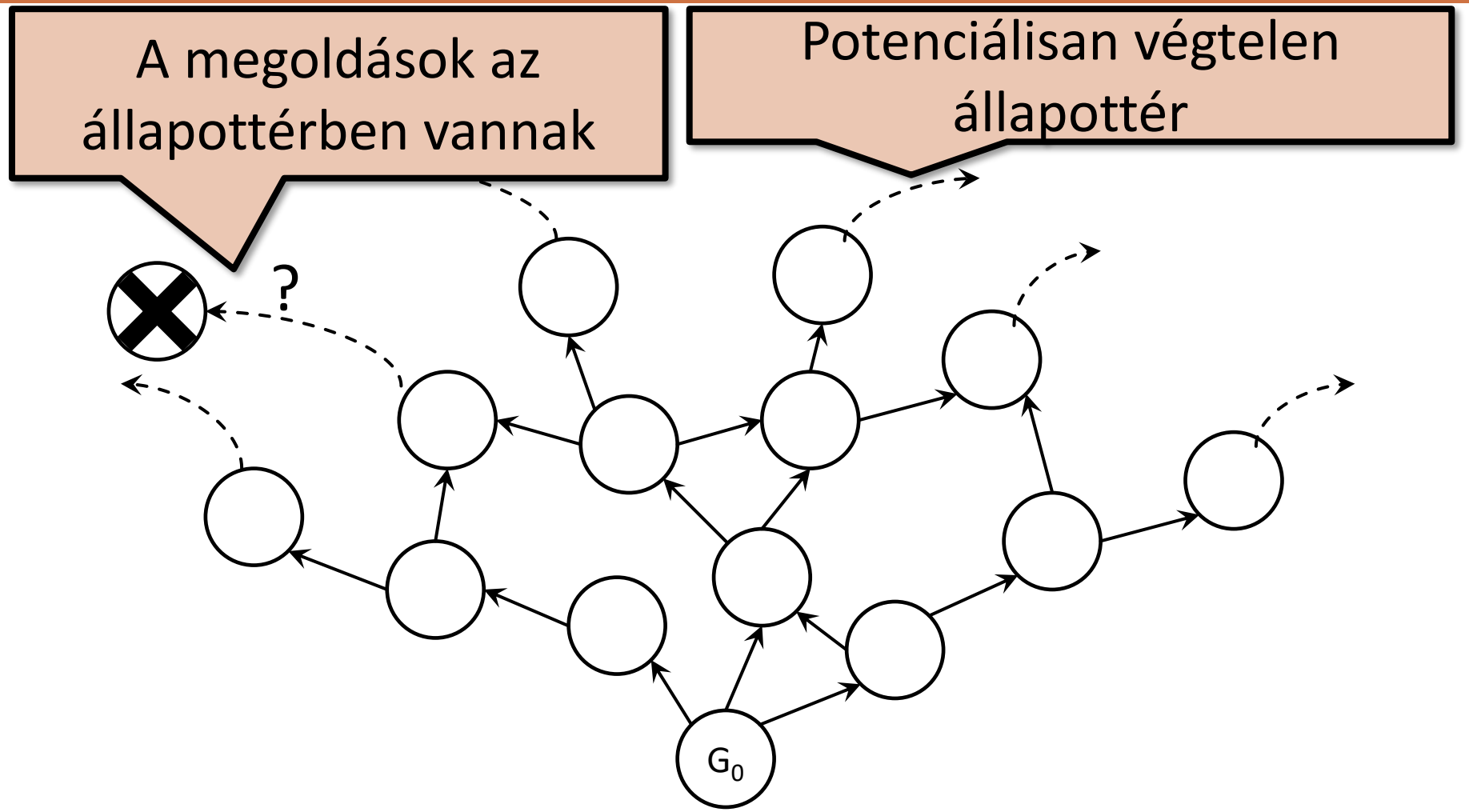
1. Első transzformáció
2. Forrásmodell megváltozik
3. Change notification feldolgozása
4. Változás terjesztése

VIATRA

Gráfmintaillesztés, Gráftranszformáció

- Alapfogalmak
- Gráfmintaillesztés
- Modelltranszformációk
- Inkrementális transzformációk
- Tervezésitér bejárás

Visszatekintés: GT rendszer állapottere



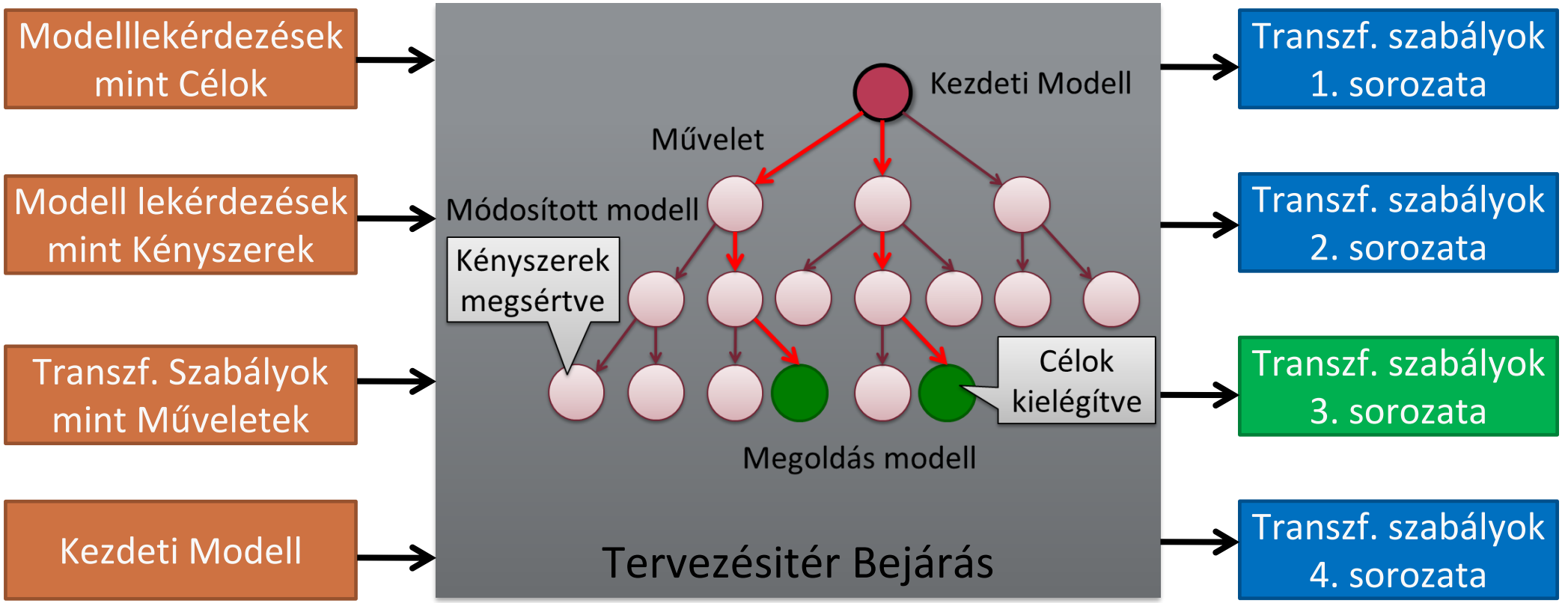
Kiindulási gráf + GT szabályok → Állapottér

Tervezésitér bejárás

Speciális állapottér bejárás

- potenciálisan végtelen állapottér
- „sűrű” megoldási tér

Modellvezérelt Irányított Tervezésitér Bejárás

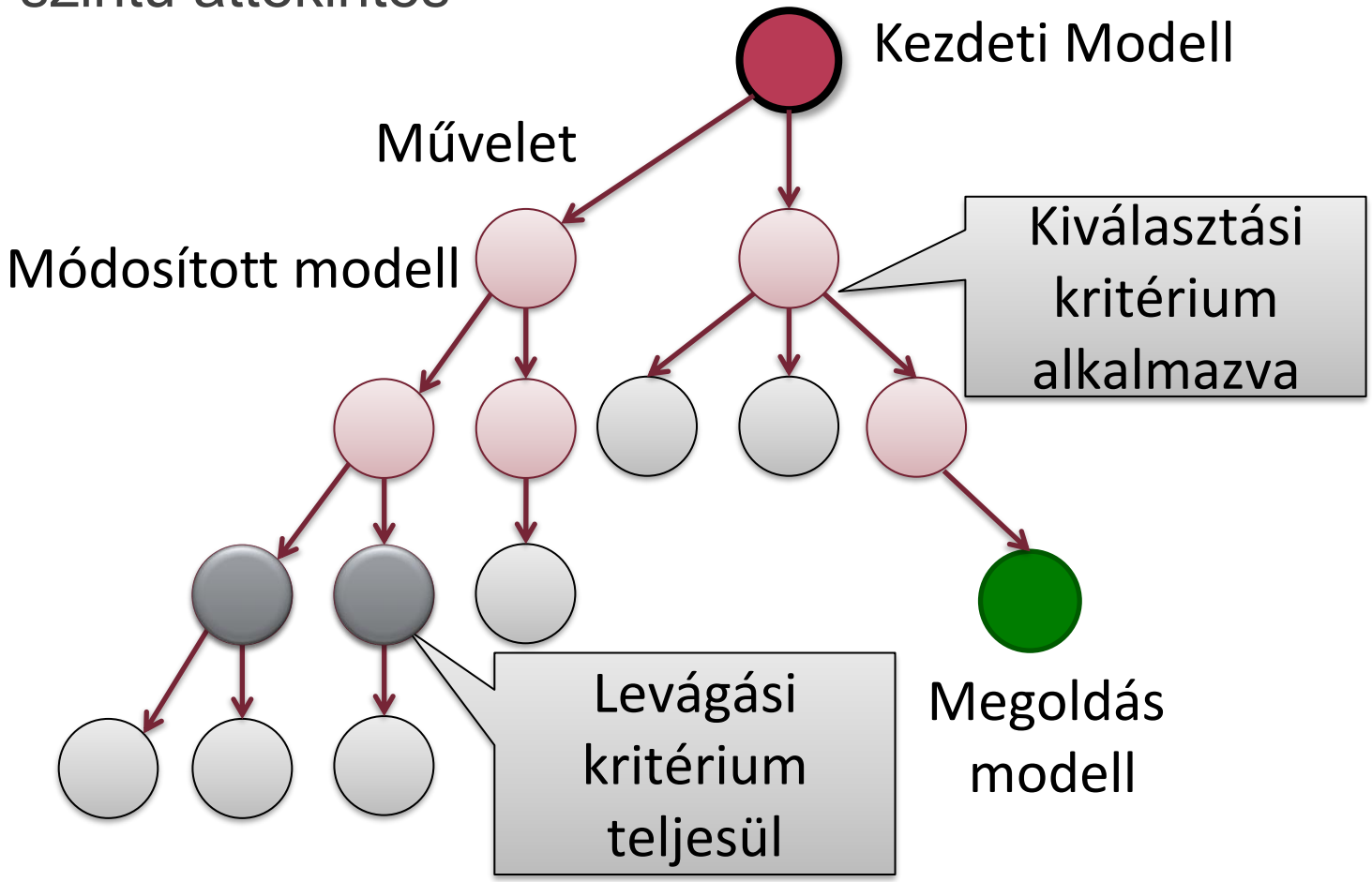


Útmutatás a bejáráshoz: Tippek

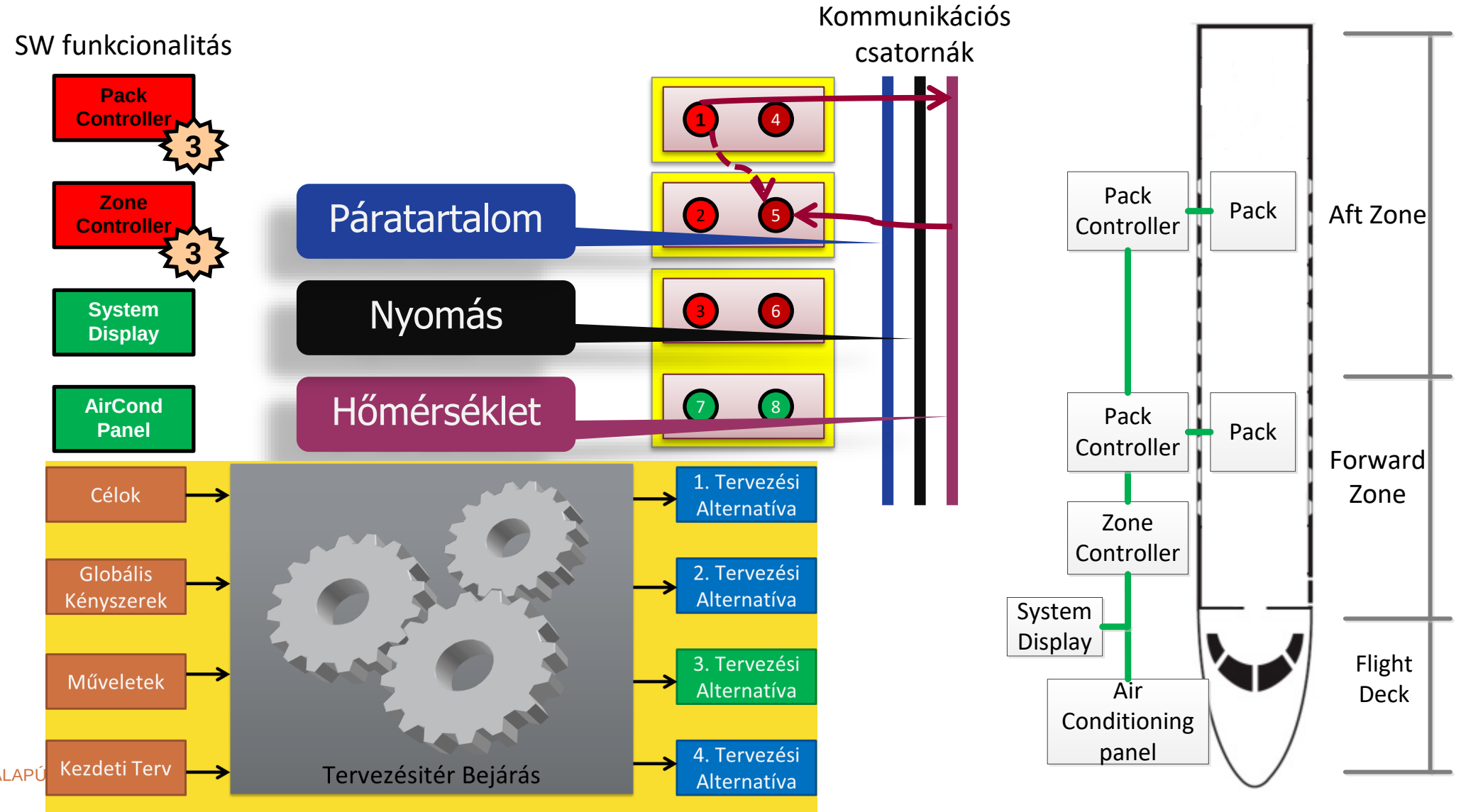
- tervező / végfelhasználó
- formális elemzés

Irányított Tervezésitér Bejárás

- Magas szintű áttekintés

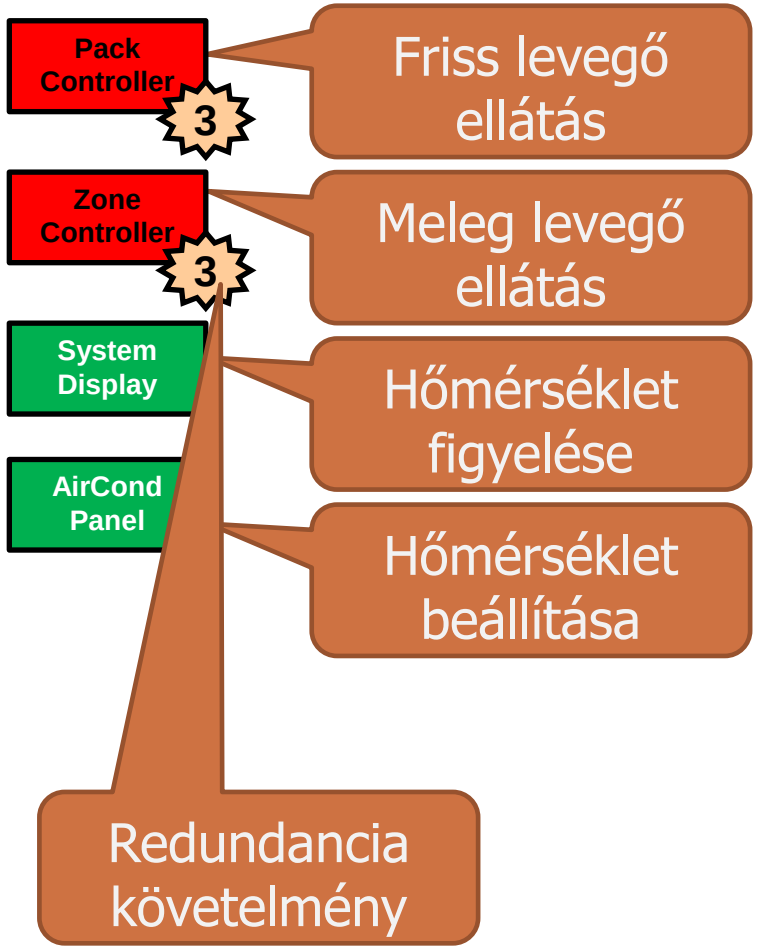


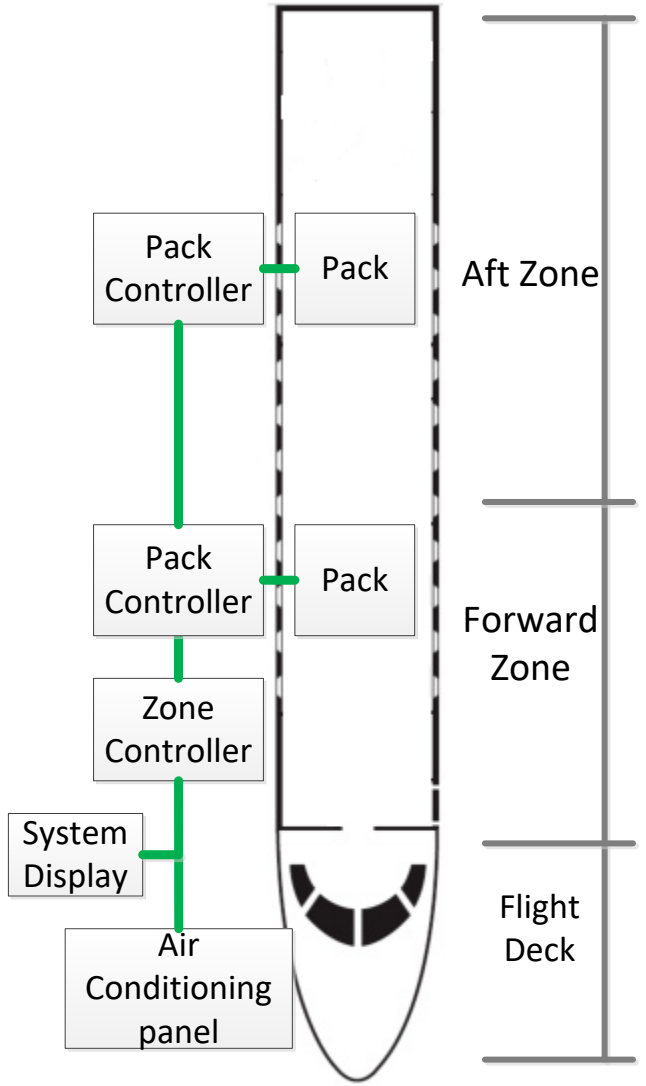
Tervezésitér Bejárás IMA Konfiguráció Tervezéséhez



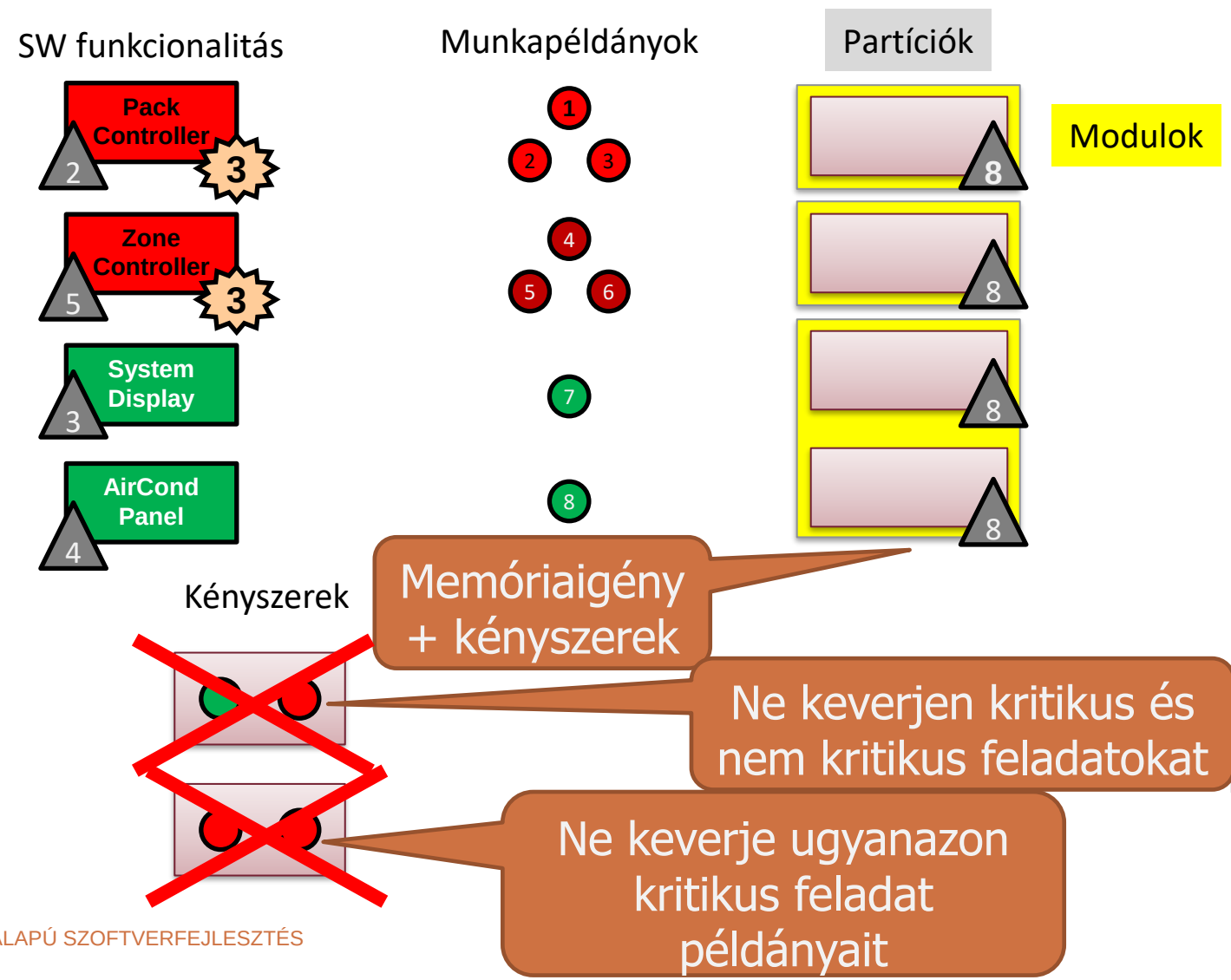
ARINC653 konfigurációk tervezése

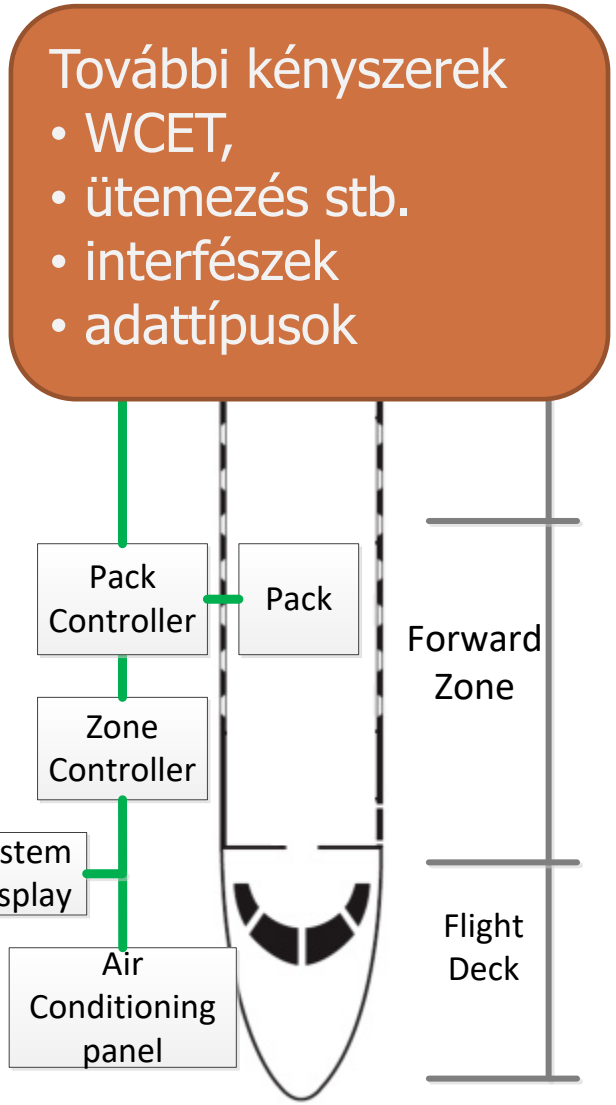
SW funkcionalitás
(kritikus + nem kritikus)



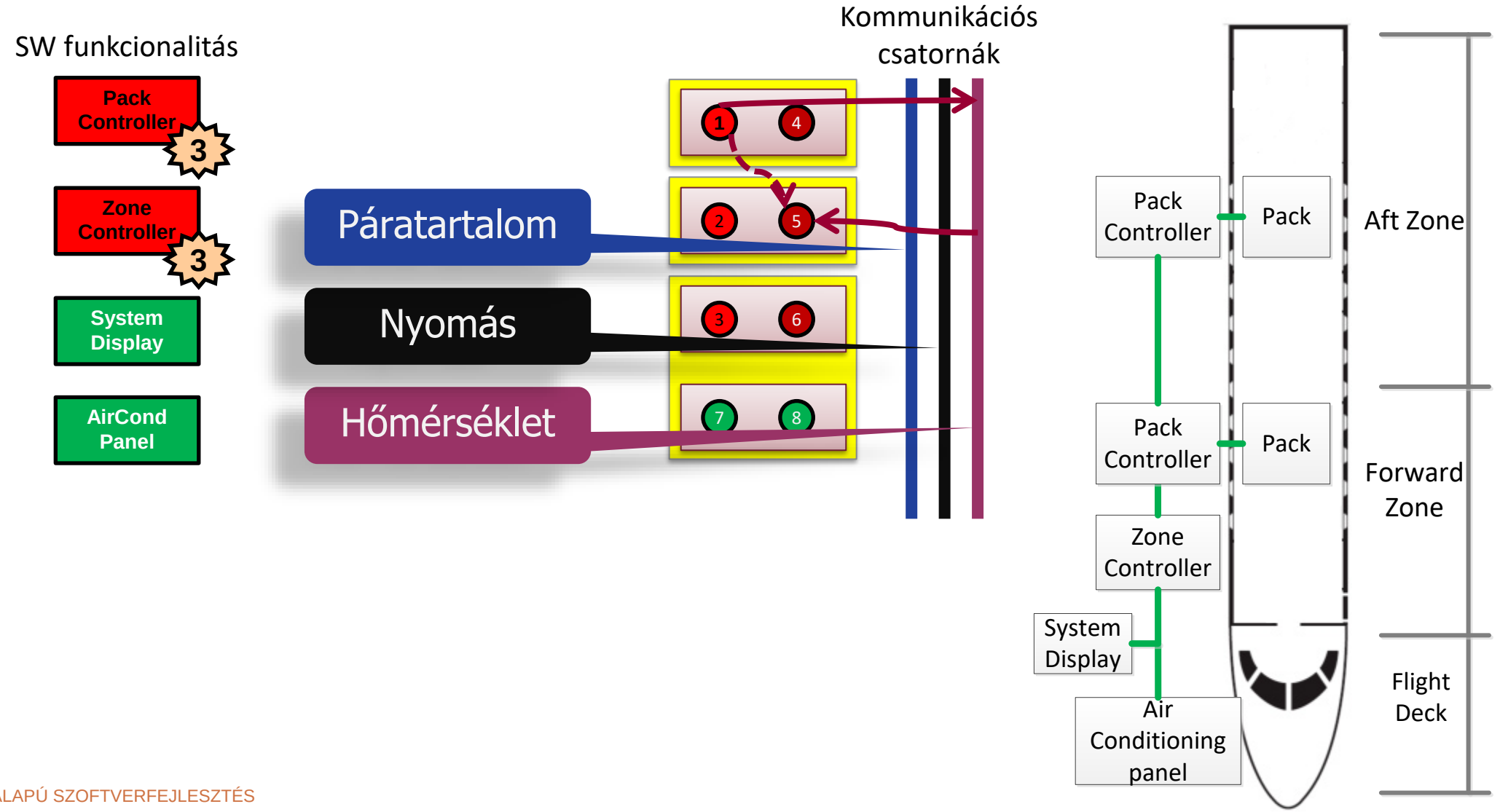


Munkapéldányok, Partíciók, Modulok





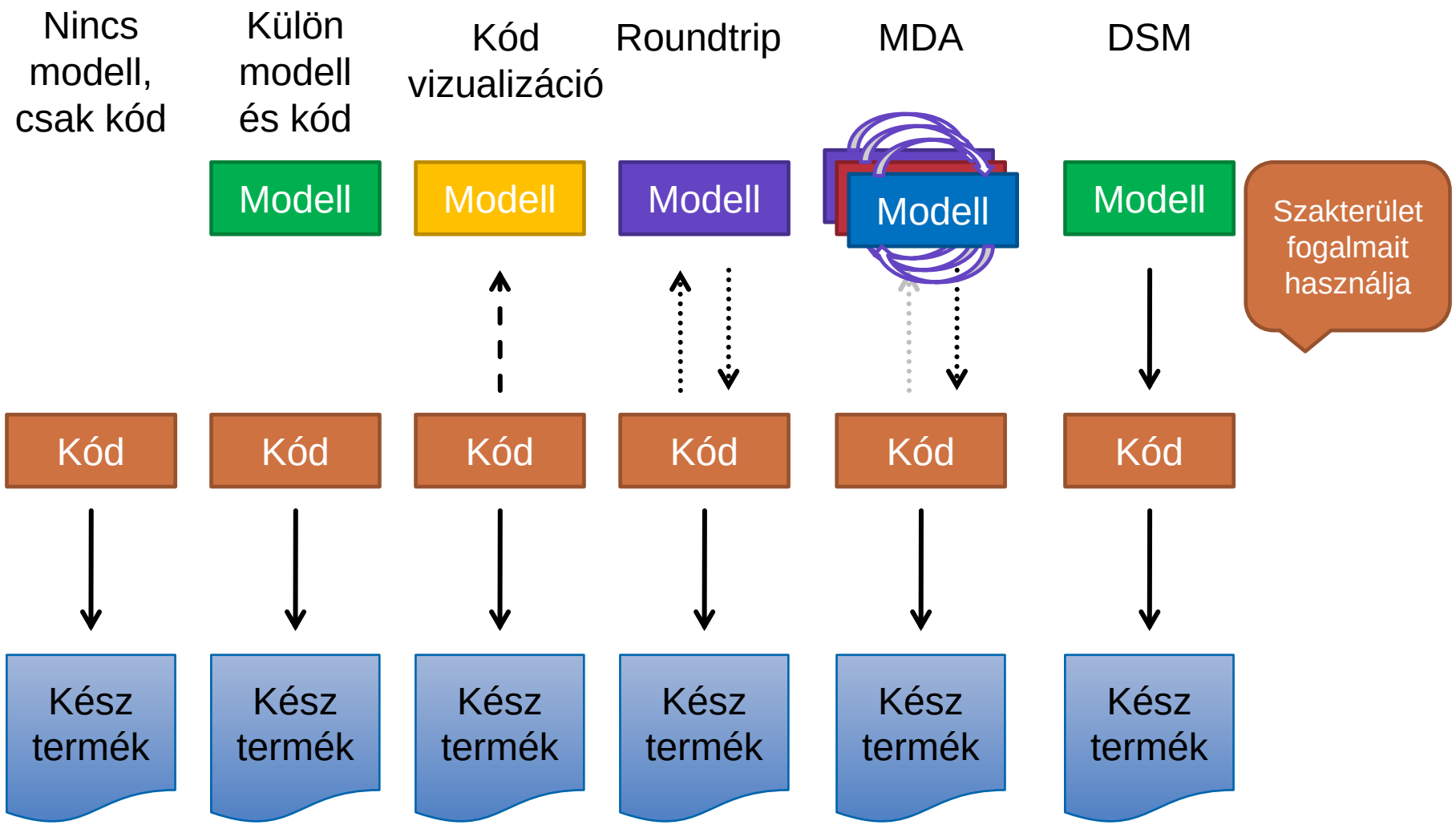
Kommunikációs csatornák kiosztása



Gráfmintaillesztés, Gráftranszformáció

- **Alapfogalmak**
- **Gráfmintaillesztés**
- **Modelltranszformációk**
- **Inkrementális transzformációk**
- **Tervezésitér bejárás**

Hogyan használunk modelleket?



Köszönöm a figyelmet!