# Formális módszerek

- Természetes nyelv – formális leírás
- A programok specifikációját, dokumentációját általában természetes nyelven készítik el.
- Ez tartalmazhat félreérthető elemeket
- Matematikai eszközök használatán alapuló formális leírás egyértelmű
- Ha a megbízhatóság, biztonság kiemelt fontosságú

# Formális jelölőrendszer előnyei

- Matematikailag elemezhetők, vagyis matematikai bizonyítási módszereket használhatunk a leírás helyességének vizsgálata során
- A leírások könnyebben karbantarthatók
- A formális leíráshoz használt nyelv bővíthető
- A programok helyességét, a leírással való összhangját matematikai szigorúsággal bizonyíthatjuk
- Lehetővé teszi a *leírás* → *program* átalakítás automatizálását

# Formális leírás szintjei

- 0. szint: formális specifikációt készítenek, majd a fejlesztés hagyományos eszközökkel folyik
  - *formal methods lite*
  - A legtöbb esetben a legköltséghatékonyabb választás
- 1. szint: formális fejlesztés és ellenőrzés
  - Nagy integritású rendszerek fejlesztésekor
- 2. szint: A tervezés egyes lépéseinek helyességét (mint matematikai tételeket) bizonyítják (Automatic Theorem Prover)
  - Nagyon drága, csak akkor éri meg, ha a hibák költsége különlegesen nagy

# Szoftverfejlesztés lépései formális eljárással

- Leírás (specifikáció)
  - modell alapú megközelítés
  - matematikai fogalmakkal jellemzi a program futása során felvehető egyes állapotokat, állapotátmeneteket (műveleteket)
- Tervezés
  - Az elvont matematikai modell közelítése egy konkrét programnyelv eszközeihez (*adatfinomítás*)
- Megvalósítás
  - Kódolás (*műveleti finomítás*)

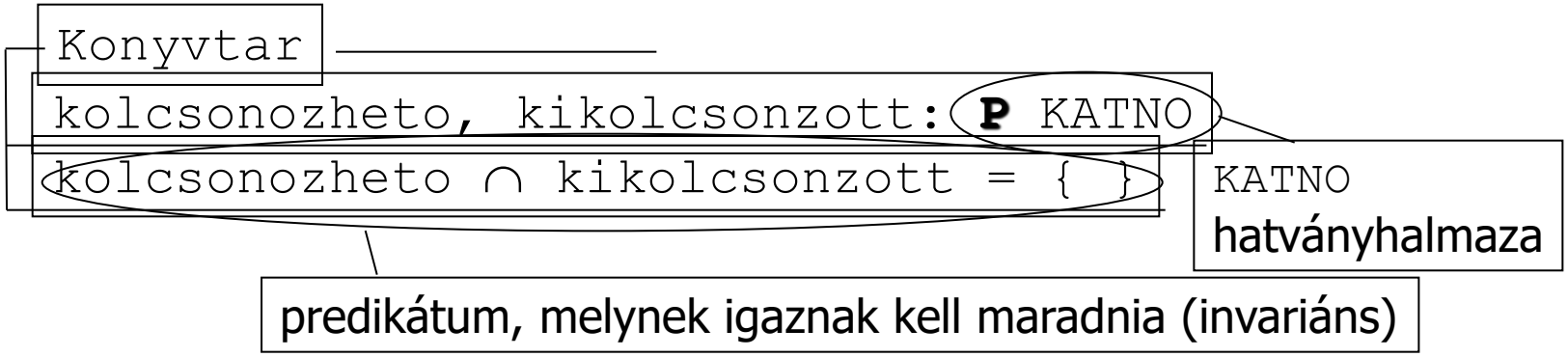
# Formális eszközök

- Z formális nyelv (Z notation, ejtsd zed)
- Vienna Development Method (VDM)

# Példa – könyvtári program

1. Leírás

- A matematikai modell megalkotása (sémák)
  - Felső rész: az állapottal kapcsolatos összetevők
  - Alsó rész: invariáns, melynek igaznak kell maradnia állapot-változáskor
  - A könyvtár két halmazból áll: a bent lévő és a kikölcsönzött könyvek halmazából
  - Egy típus bevezetése: `KATNO` – katalógusszám, illetve azok halmaza

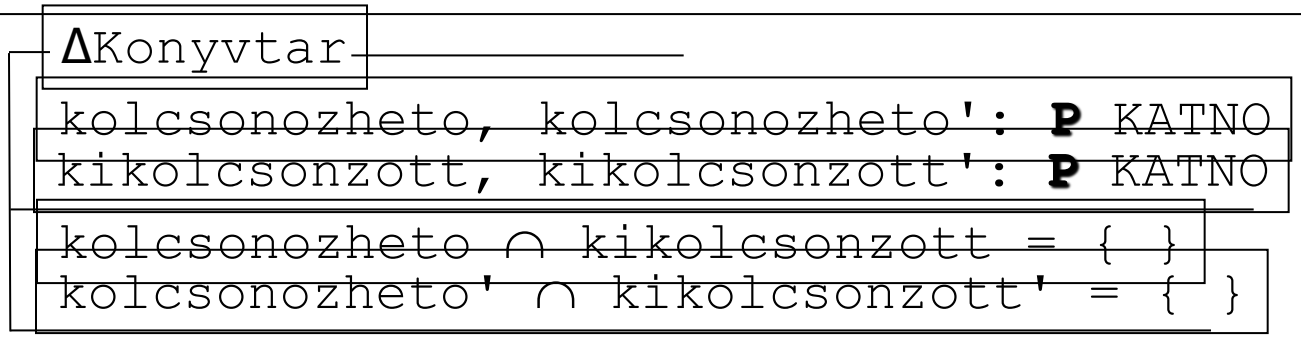


# Példa – könyvtári program

- Műveletek megadása
  - ellenőrzés: az invariánsoknak igaznak kell maradniuk
  - pl. `Kolcsonzes` művelet

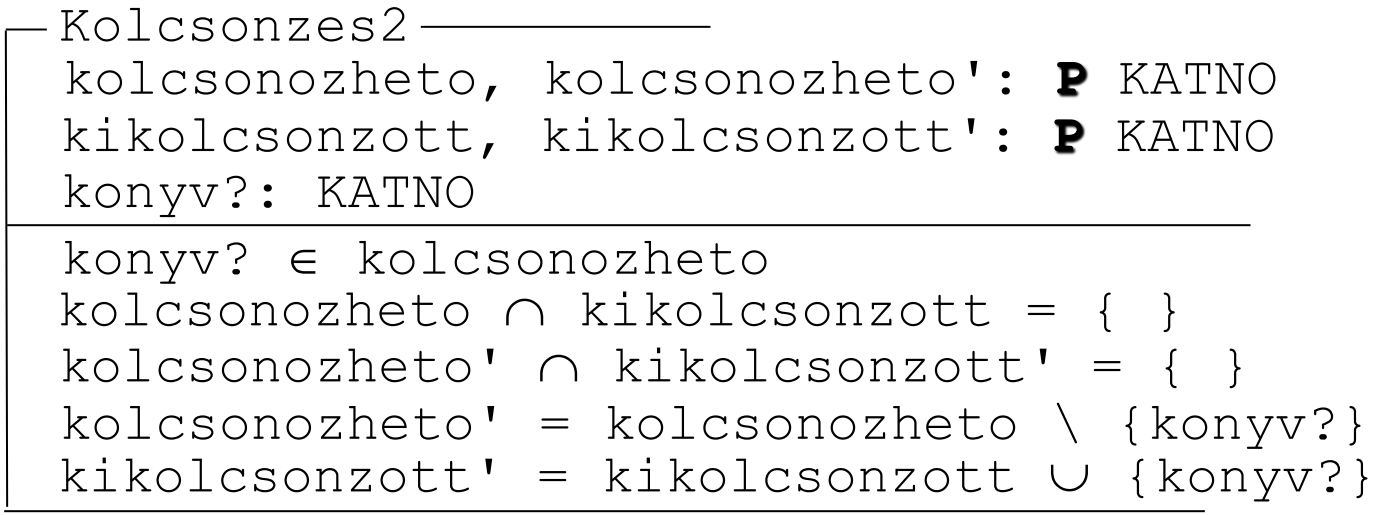
# Példa – könyvtári program


A predikátumok logikai ÉS kapcsolatban állnak.
A ΔKonyvtar séma automatikusan a rendelkezésünkre áll ( modularitás)



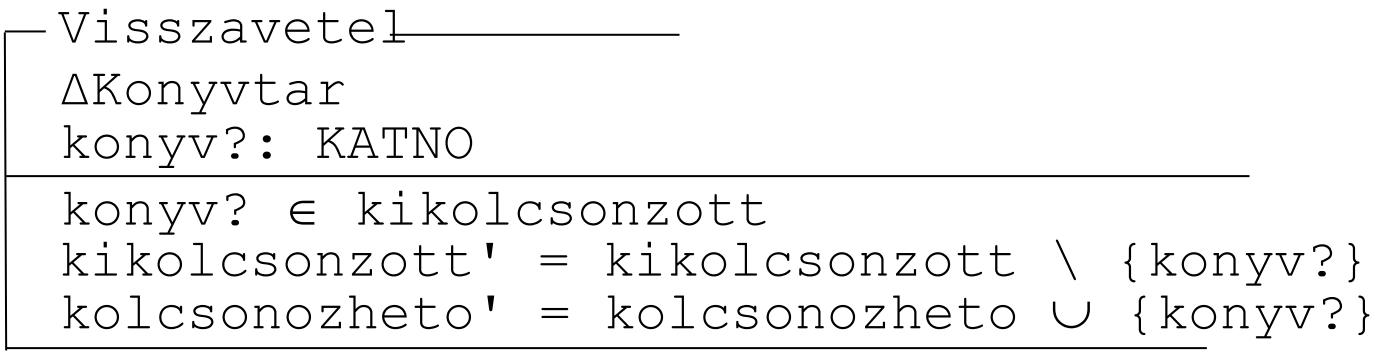
# Példa – könyvtári program

- A `Kolcsonzes` séma teljes alakja sémabeszúrás nélkül a következőképpen nézne ki



# Példa – könyvtári program

- A `Visszavetel` művelet leírás az előbbihez ha-sonlóan a következő



# Példa – könyvtári program

- A leírás helyességének ellenőrzése
- Az, hogy a teljes logikai rendszert Z nyelven fogalmaztuk meg, még nem biztosítja annak helyességét.
- Megfelel-e a leírás a megrendelő által megfogalmazott elvárásoknak?
- Követtünk-e el valamilyen hibát a Z nyelv használata közben?
- Bizonyítanunk kell a modell helyességét, egyfajta matematikai tételként kell azt kezelnünk.
- Például vizsgáljuk meg, milyen hatással van egy könyv kikölcsönzése a könyvtár teljes raktárkészletére.
  - Nyilvánvaló, hogy a könyvállománynak nem szabad megváltoznia.
  - Bizonyítsuk be, hogy ennek az elvárásnak megfelel a modellünk

# Példa – könyvtári program

Tehát a bizonyítandó állítás

```
kolcsonozheto' ∪ kikolcsonzott' =
      kolcsonozheto ∪ kikolcsonzott
```

Bizonyítás:

```
kolcsonozheto' ∪ kikolcsonzott'
1. = (kolcsonozheto \ {konyv?}) ∪
(kikolcsonzott ∪
     {konyv?})
```

a `Kolcsonzes` művelet meghatározása alapján

```
2. = (kolcsonozheto \ {konyv?}) ∪ ({konyv?} ∪
     kikolcsonzott)
```

az unióképzés kommutatív

```
3. = ((kolcsonozheto \ {konyv?}) ∪ {konyv?}) ∪
     kikolcsonzott
```

az unióképzés asszociatív

```
4. = (kolcsonozheto ∪ {konyv?}) ∪ kikolcsonzott
```

az unióképzés és a különbség közötti általános összefüggés

```
5. = kolcsonozheto ∪ kikolcsonzott
```

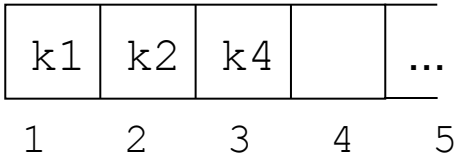
a `Kolcsonzes` művelet `konyv? ∈ kolcsonozheto` kitétele miatt

# Példa – könyvtári program

- 2. Tervezés
  - Elkészítettünk egy logikailag ellenőrzött, halmazokkal megfogalmazott elvont leírásunk
  - Az elvont adattípusokat át kell alakítanunk olyan szerkezetekké, melyek könnyen megfeleltethetők a programozás során használható szerkezeteknek. (Halmazok helyett pl. tömbök)
  - Ezeket a Z nyelv új eszközeivel, a *függvényekkel* valósítjuk meg

# Példa – könyvtári program

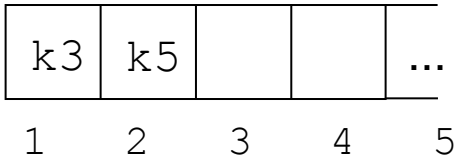
Pl. Kkolcsonozheto:




na = 3

K: konkrét

Kkikolcsonzott:




nb = 2

A Z nyelvben a tömb egy függvénnyel modellezhető, példánkban

Kkolcsonozheto, Kkikolcsonzott: $\mathbf{N}_1 \rightarrow$ KATNO

ahol $\mathbf{N}_1$ = {1, 2, 3, …} a Z nyelv egy alapvető halmaza és a példánk szerint:

Kkolcsonozheto = {1 ↦ k1, 2 ↦ k2, 3 ↦ k4, …} na =3

Kkikolcsonzott = {1 ↦ k3, 2 ↦ k5, …} nb =2

A programnyelvekben használatos Kkolcsonozheto[2] hivatkozás megfelel a Z nyelv Kkolcsonozheto(2) függvényhívásának.

# Példa – könyvtári program

- Mint a leírásnál, a konkrét modellt is egy séma formájában fogalmazzuk meg:

$$
\begin{array}{|l}
\hline
\texttt{Kkolcsonozheto, Kkikolcsonzott: } \mathbf{N}_1 \rightarrow \texttt{KATNO} \\
\texttt{na, nb: } \mathbf{N} \\
\hline
\forall i,j: 1..na \bullet i \neq j \Rightarrow Kkolcsonozheto(i) \neq Kkolcsonozheto(j) \\
\forall i,j: 1..nb \bullet i \neq j \Rightarrow Kkikolcsonozott(i) \neq Kkikolcsonzott(j) \\
\forall i: 1..na \bullet (\forall j: 1..nb \bullet Kkolcsonozheto(i) \neq Kkikolcsonzott(j)) \\
\hline
\end{array}
$$

# Példa – könyvtári program

- A `KKonyvtar` séma invariánsa három „univerzális meghatározásból” áll, melynek általános alakja:

`∀ deklarációk • predikátum`

- azaz „valamennyi így bevezetett változóra teljesülnie kell a következő állításnak”
- A példában az állítások azt jelentik, hogy nem lehet egyik tömbnek sem ismétlődő eleme, illet nem szerepelhet egyszerre egy katalógusszám mindkét tömbben

# Példa – könyvtári program

- A konkrét `Kolcsonzes` művelet:

**KKolcsonzes**

$\Delta$KKonyvtar
konyv?: KATNO

---

$\exists i: 1..na \bullet konyv? = Kkolcsonozheto(i)$
$na' = na - 1$
$Kkolcsonozheto' = squash(\{i\} \lhd Kkolcsonozheto)$
$nb' = nb + 1$
$Kkikolcsonzott' = Kkikolcsonzott \cup (nb' \rightarrow konyv?)$

# Példa – könyvtári program

- $\exists$ deklaráció $\bullet$ predikátum
  a változónak létezik olyan értéke, amelyre az állítás igaz

- s $\lhd$ f
  tartománykivonás: azt a függvényt jelenti, amely úgy keletkezik, hogy az f függvény értelmezési tartományából elvesszük az s tartományt.

- squash: az indexek átrendezésével összehúzza a „lyukas” tömböt

# Példa – könyvtári program

- Még formálisan be kell bizonyítani, hogy a `KKolcsonzes` művelet valóban a `Kolcsonzes` művelet finomítása
  - a két séma előfeltétele egyenértékű, azaz valahányszor a `Kolcsonzes` végrehajtható, mindannyiszor a neki megfelelő `KKolcsonzes` is elvégezhető;
  - a `Kolcsonzes` és a konkrét `KKolcsonzes` művelet által előállított végállapotok egyenértékűek
- Ezt az egyenértékűséget természetesen a tervben szereplő valamennyi műveletpárra be kell bizonyítanunk
- Mindez nem egyszerű feladat

# Példa – könyvtári program

- 3. Megvalósítás
  - A következő lépés az elvont leírás meg-valósítása, vagyis a kódolás
    - intuitív módon megalkotjuk a szükséges algoritmusokat, majd a keletkezett kódot formális módszerek használatával összevetjük az eredeti tervvel;
    - formális átalakítási szabályok alkalmazásával közvetlenül állítjuk elő az elvont tervből a kódot (*műveleti finomítás*).