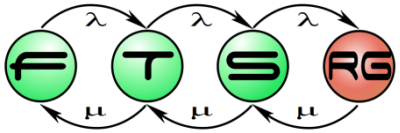
Szöveges domain-specifikus nyelvek



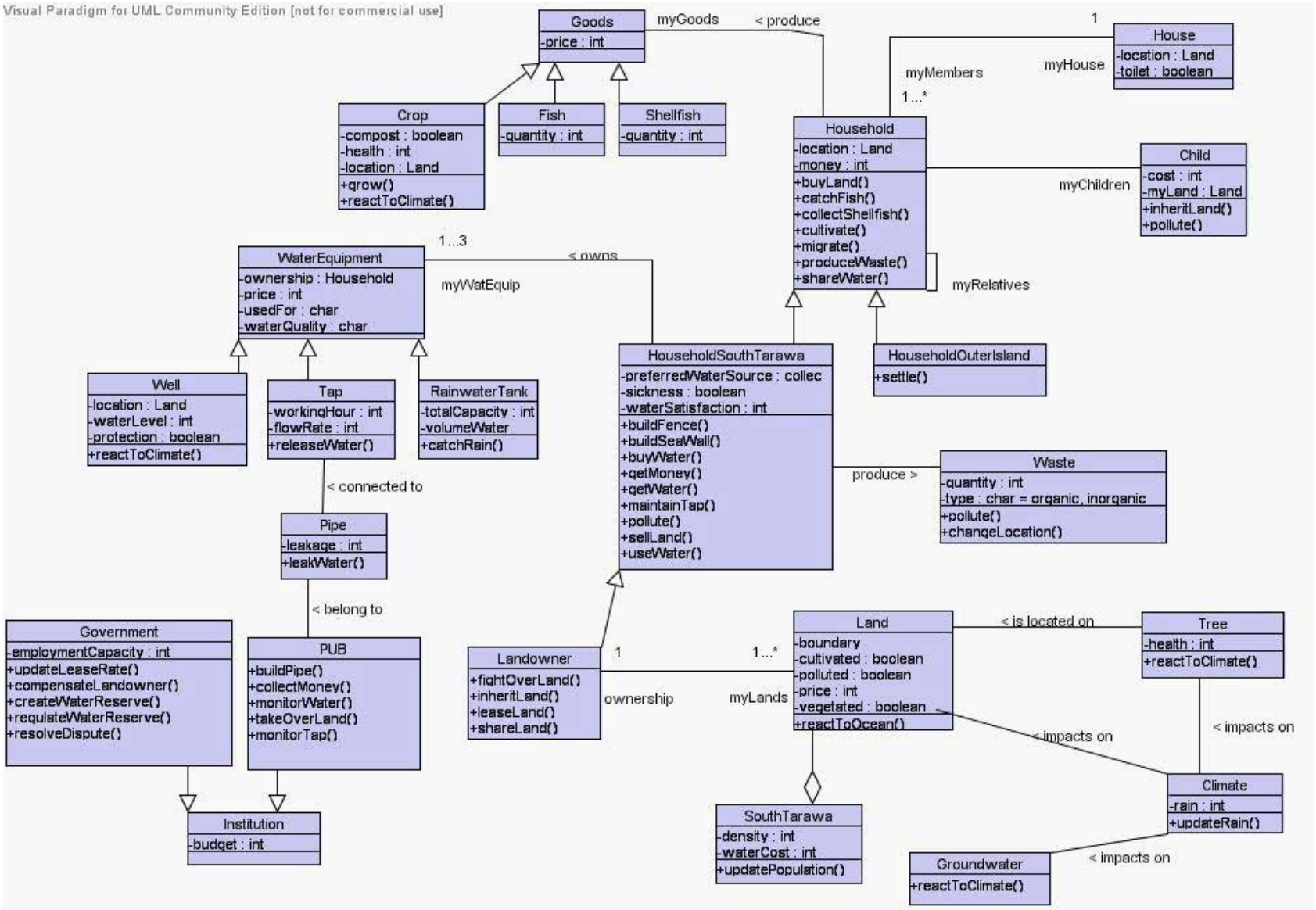
Összegzés

- Eddig
  - Metamodellek
  - Grafikus szerkesztők

Grafikus szerkesztők

- Átlátható, olvasható
- Nehézkes szerkesztés
- Nagyobb (20+ elem) modellek kezelése

Nagyobb modell



Forrás: http://epress.anu.edu.au/cs/html/ch12s04.html

Grafikus szerkesztők

- Átlátható, olvasható
- Nehézkes szerkesztés
- Nagyobb (20+ elem) modellek kezelése

Helyettük: Szöveges nyelvek

Szerkesztők szöveges nyelvekhez

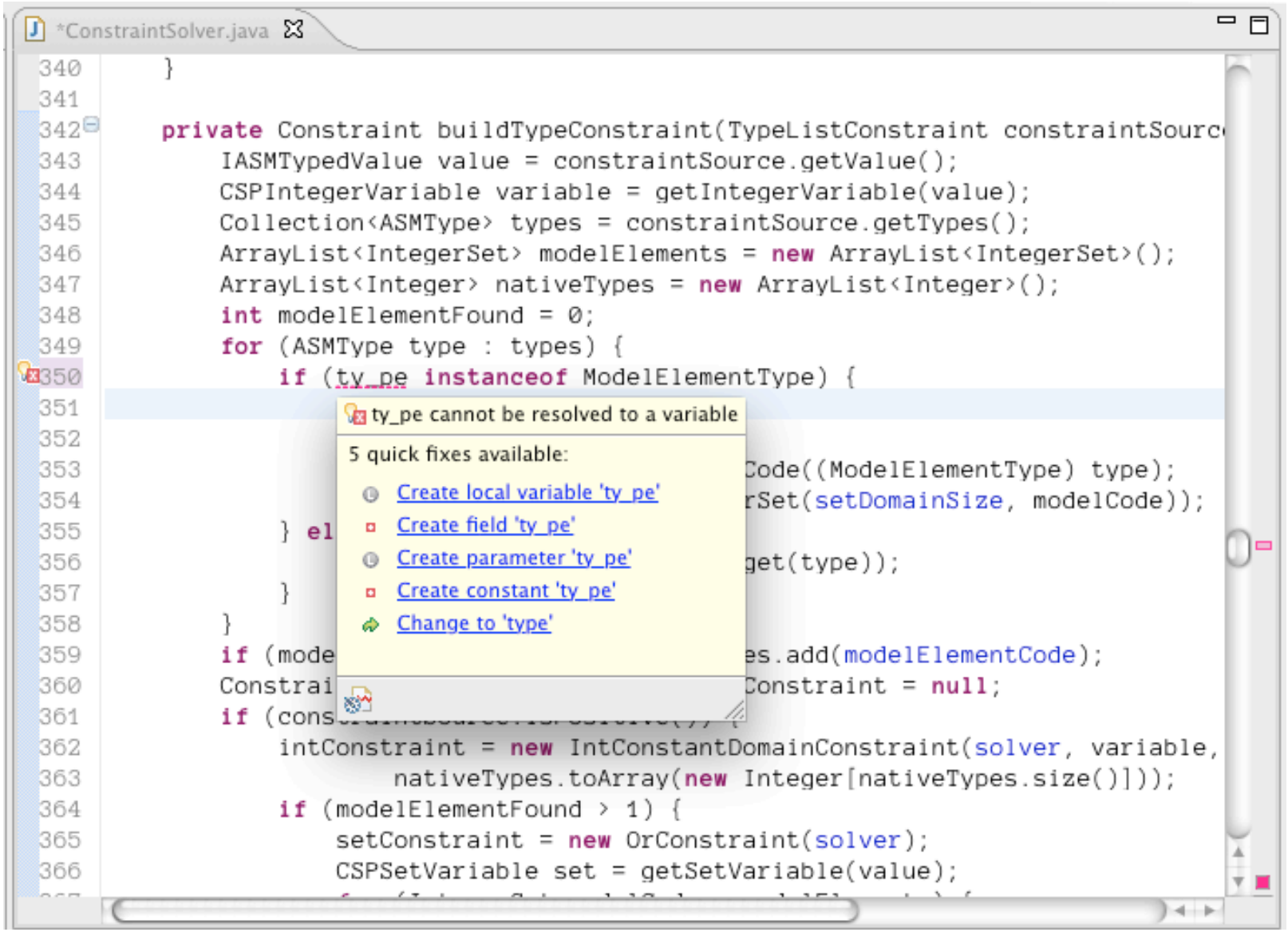
- Első körben
  - Ablak, amibe szöveget írunk
  - +támogató funkciók

Támogatás szintjei

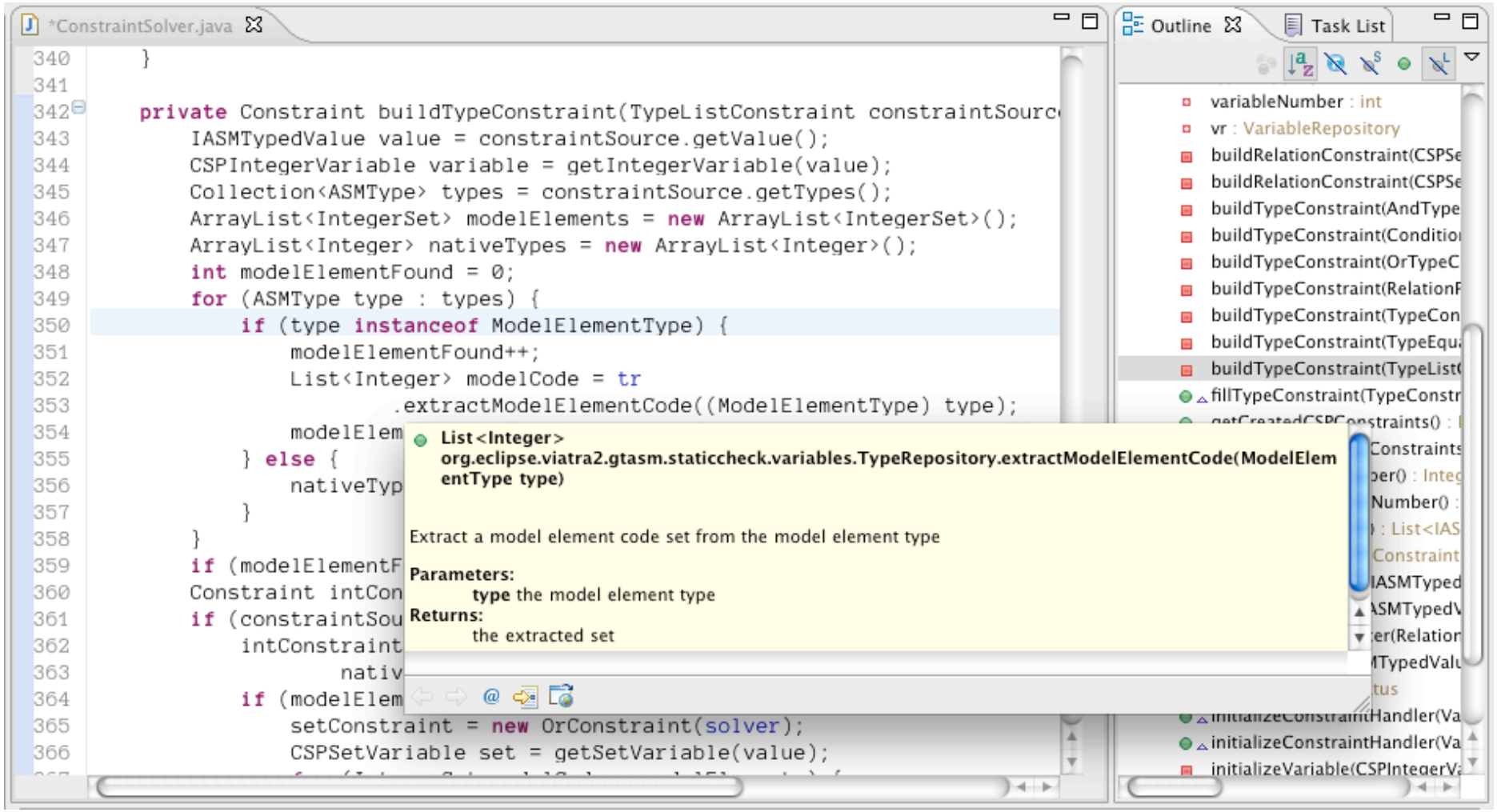
- Egyszerű szövegszerkesztő
  - Példa: Jegyzettömb
  - Alapműveletek (vágólapkezelés, keresés/csere, stb.)
  - Implementálva
- Programozói szövegszerkesztő
  - Példa: Notepad++, Emacs, Vim, …
  - Nyelvspecifikus szolgáltatások (forráskód színezés, …)
- Integrált szövegszerkesztő
  - Példa: Eclipse JDT, Visual Studio
  - Összetett szolgáltatások (projekt támogatás, autocomplete, dokumentáció, …)

JDT szolgáltatások

JDT szolgáltatások: Hibajelzés



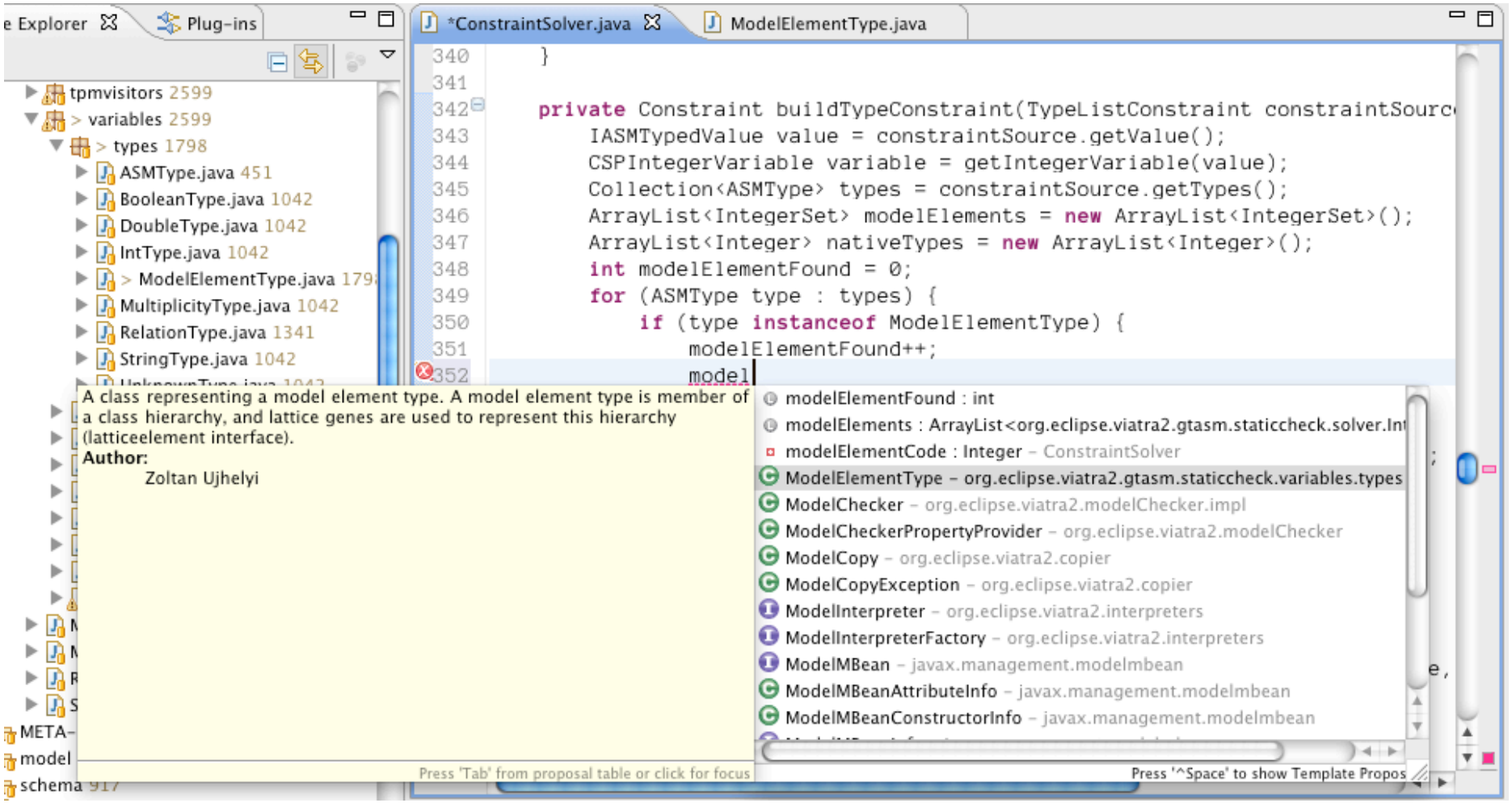
JDT szolgáltatások: Dokumentáció



JDT szolgáltatások: Occurrence marker

```
341
342    private Constraint buildTypeConstraint(TypeListConstraint constraintSourc
343        IASMTypedValue value = constraintSource.getValue();
344        CSPIntegerVariable variable = getIntegerVariable(value);
345        Collection<ASMType> types = constraintSource.getTypes();
346        ArrayList<IntegerSet> modelElements = new ArrayList<IntegerSet>();
347        ArrayList<Integer> nativeTypes = new ArrayList<Integer>();
348        int modelElementFound = 0;
349        for (ASMType type : types) {
350            if (type instanceof ModelElementType) {
351                modelElementFound++;
352                List<Integer> modelCode = tr
353                        .extractModelElementCode((ModelElementType) type);
354                modelElements.add(new IntegerSet(setDomainSize, modelCode));
355            } else {
356                nativeTypes.add(typeMapping.get(type));
357            }
358        }
359        if (modelElementFound > 0) nativeTypes.add(modelElementCode);
360        Constraint intConstraint = null, setConstraint = null;
361        if (constraintSource.isPositive()) {
362            intConstraint = new IntConstantDomainConstraint(solver, variable,
363                    nativeTypes.toArray(new Integer[nativeTypes.size()]));
364            if (modelElementFound > 1) {
365                setConstraint = new OrConstraint(solver);
366                CSPSetVariable set = getSetVariable(value);
```

JDT szolgáltatások: Content assist



JDT szolgáltatások

- Struktúra áttekintéséhez nézetek
- Navigációs linkek
- Varázslók
- Refactoring
- Inkrementális builder
- Futtatás/debug támogatás
- Bővíthetőség!

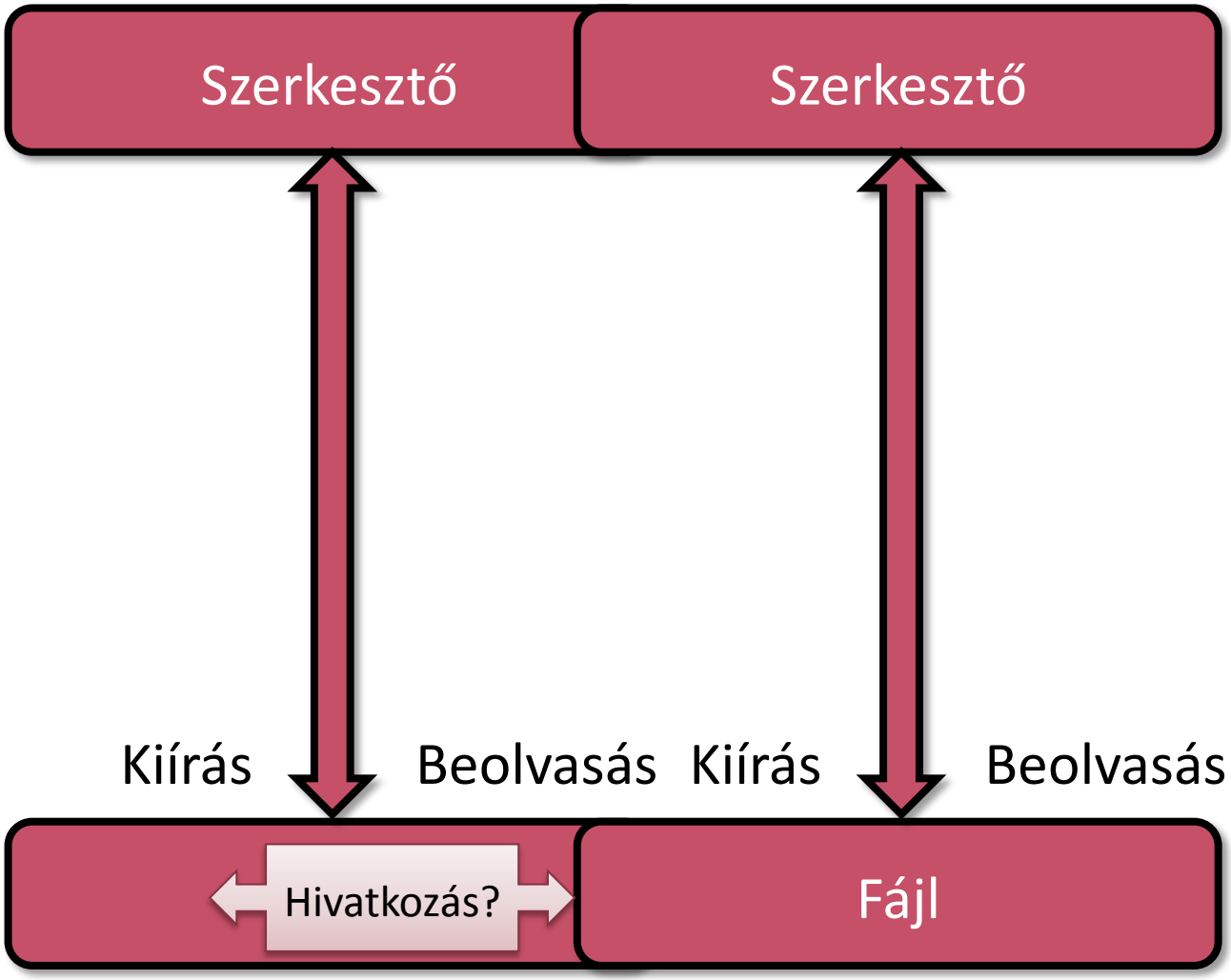
Összegzés

- Integrált szerkesztők szöveges nyelvekhez
  - Sokféle hasznos funkcionalitás
- Ebben a blokkban
  - Szerkesztők
    - Beolvasás, kiírás
    - Dokumentum modell
  - (Részben) később
    - Builder
    - Projektek

Szöveges editorok Eclipse alatt

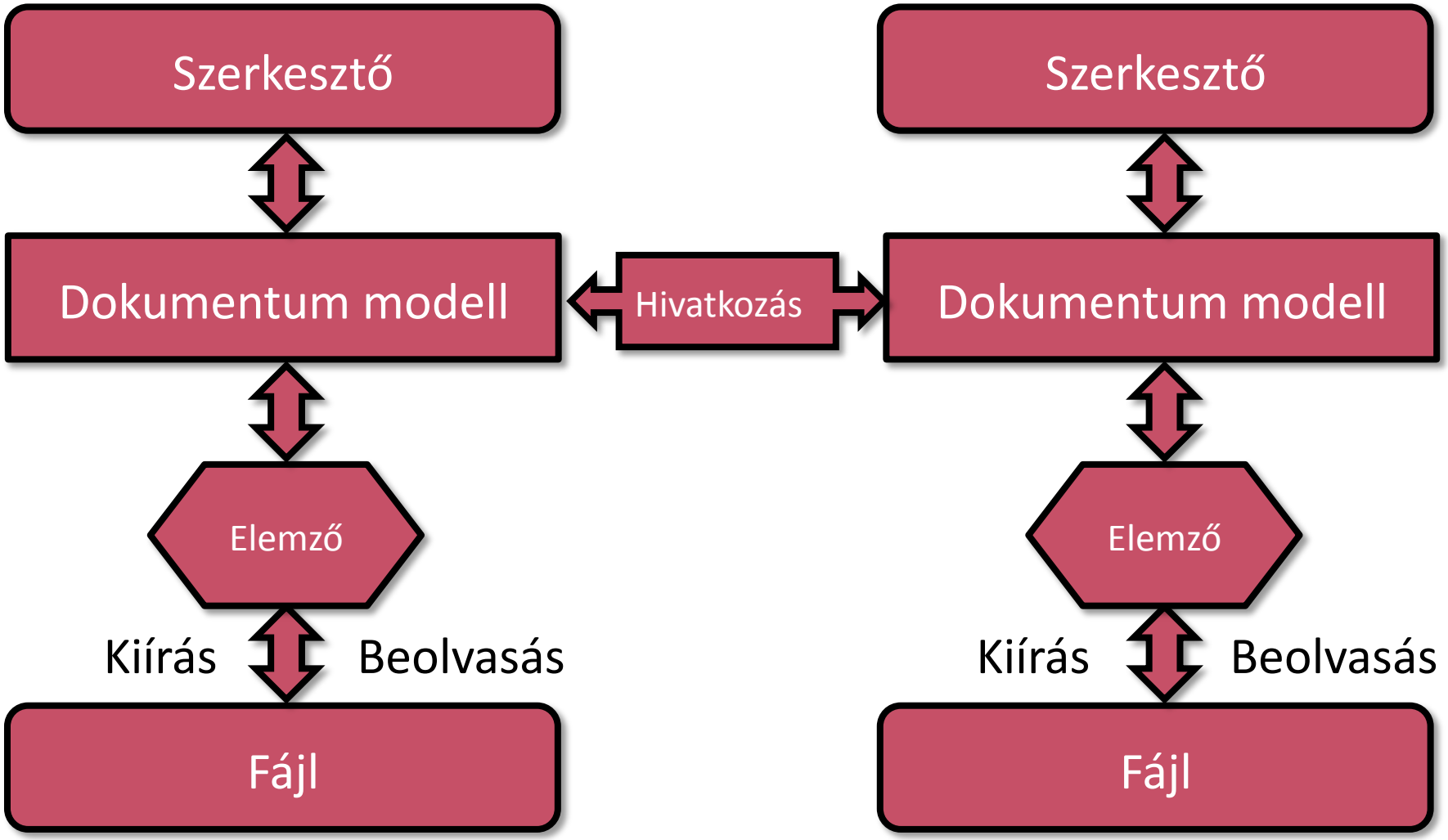
Architektúra

- Egyszerű szövegszerkesztő



Architektúra

- Dokumentum modell (MVC architektúra)



Dokumentum modell

- Objektum reprezentáció a forrásfájlhoz
- Mire használjuk?
  - Hivatkozások követése
    - Fájlon belül
    - Fájlok között
  - Editor szolgáltatások erre épülnek
    - Content assist
    - Outline
    - …

Dokumentum modell

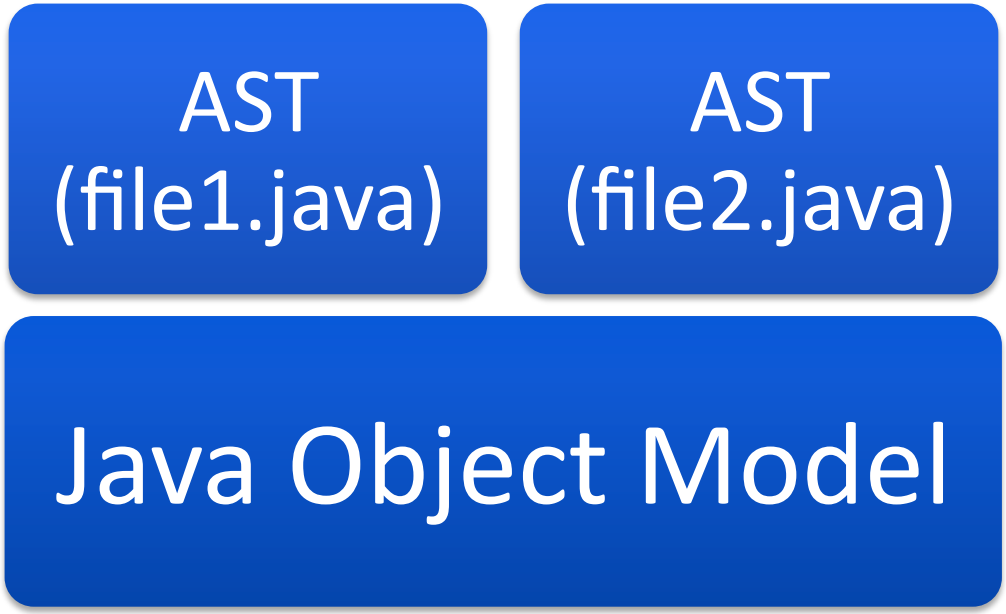
- Felépítés
  - Szöveg tördelése
  - Absztrakt szintaxis
    - Szoros kapcsolat nyers szöveggel
    - Azonosítható a szövegszakaszok
- Előállítás
  - Elemzőkkel

Dokumentum modell

- Részletesség
  - Teljes részletesség
    - 1:1 megfeleltetés modell és fájl tartalma között
    - Megvalósítás
      - AST
      - AST hivatkozáskövetéssel
  - Áttekintő szint
    - A modell kevésbé részletes, mint a fájl
    - Cél: több fájl tartalmának együttes áttekintése

Példa: JDT dokumentum modell

- Java Object Model
  - Osztályok
  - Metódusok
  - Attribútumok
  - Java projektekre
- AST
  - Teljes részletesség
  - Lusta felépítés



Dokumentum modell előállítása

- Nyelvspecifikus elemző segítségével
  - Beolvasás
  - Részekre bontás
  - Hivatkozás feloldás
  - AST/Dokumentum modell építés
- Hibakezelés!
- Később részletesebben

Jelölők

- Jelölők (markerek)
  - Platform támogatás információk hozzárendelésére
- Fajtái
  - Error marker
  - Bookmark
  - Task marker
  - …
  - Egyedi jelölők

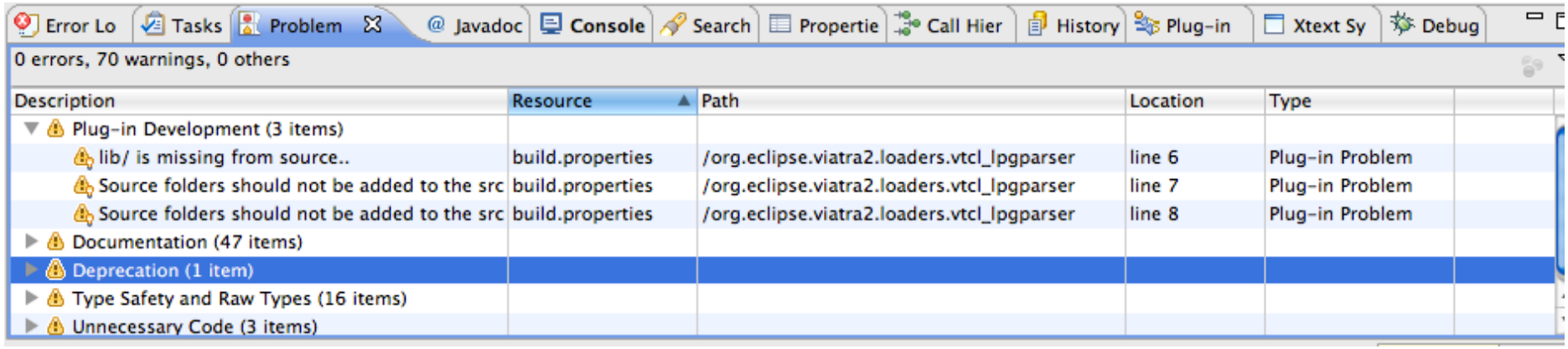
Jelölők

- Visszajelzés
  - Elemzők
  - Analízis
  - Felhasználói
- Szabad adatszerkezet
  - Kulcs-érték párok
  - Megjelenítés nincs definiálva!
  - Nézetek
    - Problems view
    - Bookmarks
    - Tasks
    - …

Jelölők

- Fájlokhoz rendelve
  - Ill. workspace erőforrásokhoz
- Tipikus értékek
  - Sorszám/terület

Problem markers



Dokumentum modell felhasználása

- Content Assist
  - Milyen elemek vannak már definiálva?
  - Modellspecifikus szűrés
- Outline készítés
  - Egyszerű szűrése a dokumentum modellnek
- Hivatkozások követése
  - Occurrence marking
  - Navigáció

Dokumentum modell felhasználása

- Refactoring
  - Modell szintjén célszerű elvégezni
    - Könnyebb értelmezni a változtatásokat
  - Modell támogatja a visszaírást
- Pretty printing
  - Modell kiírása
  - Tördelés

Szöveges nyelvek szerkesztése

Szöveges nyelvek feldolgozása

- Természetes nyelvek
- XML feldolgozás
- Nyelvtan alapú feldolgozás
  - Kézzel kódolt elemző
  - Reguláris kifejezések
  - Generált elemző

Természetes nyelvek feldolgozása

- Általánosan “nehéz”
  - Kontextus-információk
  - Nem (jól) definiált szemantika

Környezet

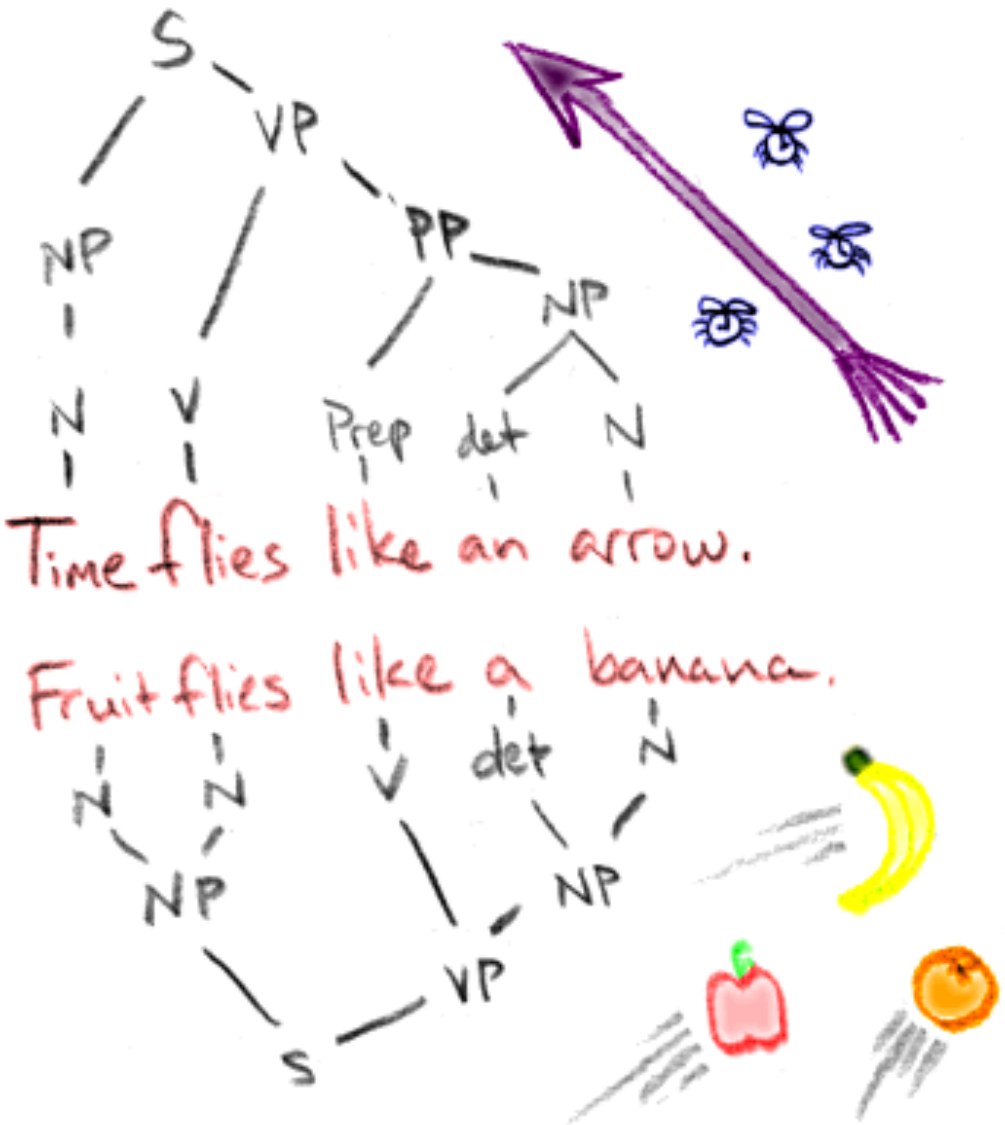
We gave the monkeys$_1$ the bananas$_2$

... because they$_1$ were hungry.

... because they$_2$ were ripe.

Forrás: http://specgram.com/CLIII.4/08.phlogiston.cartoon.zhe.html

Jelentések


S
VP
NP
PP
NP
N
V
Prep
det
N
Time flies like an arrow.
Fruit flies like a banana.
N
N
V
det
N
NP
NP
VP
S

Egyértelműség

XML feldolgozás

- XML: szabványos szöveges szintaxis
- Sémák definiálhatóak
  - Kb. metamodell
- Jó tooling
  - Többféle beolvasási technika
  - Validáció

Problémák az XML használatával

- Nehéz írni
  - Lezáró címkék
  - Entitások (pl. `&gt;` `&#8222;`)
- Nehéz olvasni
  - Szigorú fastruktúra
  - Hivatkozásfeloldás nem automatikus

Reguláris kifejezések

- Mintaillesztés karakterláncokban
  - Jó támogatás
    - Legtöbb programozási nyelven
    - Nagyjából közös szintaxis
  - Illeszkedés számítás/visszaadás
- Felhasználható szöveg elemzésére?

Some people, when confronted with a problem, think "I know, I'll use regular expressions." Now they have two problems.

Jamie Zawinski
http://regex.info/blog/2006-09-15/247

RegExp: Email validáció

```
(?:(?:\r\n)?[ \t])*(?:(?:(?:[^()<>@,;:\\".\[\] \000-\031]+(?:(?:(?:\r\n)?[ \t] )+|\Z|(?=[\["()<>@,;:\\".\[\]]))|"(?:[^\"\r\\]|\\.|(?:(?:\r\n)?[ \t]))*"(?:(?: \r\n)?[ \t])*)(?:\.(?:(?:\r
\n)?[ \t])*(?:[^()<>@,;:\\".\[\] \000-\031]+(?:(?:( ?:\r\n)?[ \t])+|\Z|(?=[\["()<>@,;:\\".\[\]]))|"(?:[^\"\r\\]|\\.|(?:(?:\r\n)?[ \t]))*"(?:(?:\r\n)?[ \t])*))*@(?:(?:\r\n)?[ \t])*(?:
[^()<>@,;:\\".\[\] \000-\0 31]+(?:(?:(?:\r\n)?[ \t])+|\Z|(?=[\["()<>@,;:\\".\[\]]))|\[([^\[\]\r\\]|\\.)*\ ](?:(?:\r\n)?[ \t])*)(?:\.(?:(?:\r\n)?[ \t])*(?:[^()<>@,;:\\".\[\] \000-
\031]+ (?:(?:(?:\r\n)?[ \t])+|\Z|(?=[\["()<>@,;:\\".\[\]]))|\[([^\[\]\r\\]|\\.)*\](?: (?:\r\n)?[ \t])*))*|(?:[^()<>@,;:\\".\[\] \000-\031]+(?:(?:(?:\r\n)?[ \t])+|\Z |(?=[\["()<>@,;:
\\".\[\]]))|"(?:[^\"\r\\]|\\.|(?:(?:\r\n)?[ \t]))*"(?:(?:\r\n) ?[ \t])*)*\<(?:(?:\r\n)?[ \t])*(?:@(?:[^()<>@,;:\\".\[\] \000-\031]+(?:(?:(?:\ r\n)?[ \t])+|\Z|(?=[\["()<>@,;:\\".\
[\]]))|\[([^\[\]\r\\]|\\.)*\](?:(?:\r\n)?[ \t])*)(?:\.(?:(?:\r\n)?[ \t])*(?:[^()<>@,;:\\".\[\] \000-\031]+(?:(?:(?:\r\n) ?[ \t])+|\Z|(?=[\["()<>@,;:\\".\[\]]))|\[([^\[\]\r\\]|\\.)*\](?:
(?:\r\n)?[ \t] )*))*(?:,@(?:(?:\r\n)?[ \t])*(?:[^()<>@,;:\\".\[\] \000-\031]+(?:(?:(?:\r\n)?[ \t])+|\Z|(?=[\["()<>@,;:\\".\[\]]))|\[([^\[\]\r\\]|\\.)*\](?:(?:\r\n)?[ \t])* )(?:\.(?:(?:
\r\n)?[ \t])*(?:[^()<>@,;:\\".\[\] \000-\031]+(?:(?:(?:\r\n)?[ \t] )+|\Z|(?=[\["()<>@,;:\\".\[\]]))|\[([^\[\]\r\\]|\\.)*\](?:(?:\r\n)?[ \t])*))*) *:(?:(?:\r\n)?[ \t])*)?(?:[^()<>@,;:\
\".\[\] \000-\031]+(?:(?:(?:\r\n)?[ \t])+ |\Z|(?=[\["()<>@,;:\\".\[\]]))|"(?:[^\"\r\\]|\\.|(?:(?:\r\n)?[ \t]))*"(?:(?:\r \n)?[ \t])*)(?:\.(?:(?:\r\n)?[ \t])*(?:[^()<>@,;:\\".\[\] \000-
\031]+(?:(?:(?: \r\n)?[ \t])+|\Z|(?=[\["()<>@,;:\\".\[\]]))|"(?:[^\"\r\\]|\\.|(?:(?:\r\n)?[ \t ]))*"(?:(?:\r\n)?[ \t])*))*@(?:(?:\r\n)?[ \t])*(?:[^()<>@,;:\\".\[\] \000-\031 ]+(?:(?:
(?:\r\n)?[ \t])+|\Z|(?=[\["()<>@,;:\\".\[\]]))|\[([^\[\]\r\\]|\\.)*\]( ?:(?:\r\n)?[ \t])*)(?:\.(?:(?:\r\n)?[ \t])*(?:[^()<>@,;:\\".\[\] \000-\031]+(? :(?:(?:\r\n)?[ \t])+|\Z|(?=[\["()
<>@,;:\\".\[\]]))|\[([^\[\]\r\\]|\\.)*\](?:(? :\r\n)?[ \t])*))*\>(?:(?:\r\n)?[ \t])*)|(?:[^()<>@,;:\\".\[\] \000-\031]+(?:(? :(?:\r\n)?[ \t])+|\Z|(?=[\["()<>@,;:\\".\[\]]))|"(?:[^
\"\r\\]|\\.|(?:(?:\r\n)? [ \t]))*"(?:(?:\r\n)?[ \t])*)*:(?:(?:\r\n)?[ \t])*(?:(?:(?:[^()<>@,;:\\".\[\] \000-\031]+(?:(?:(?:\r\n)?[ \t])+|\Z|(?=[\["()<>@,;:\\".\[\]]))|"(?:[^\"\r\\]| \
\.|(?:(?:\r\n)?[ \t]))*"(?:(?:\r\n)?[ \t])*)(?:\.(?:(?:\r\n)?[ \t])*(?:[^()<> @,;:\\".\[\] \000-\031]+(?:(?:(?:\r\n)?[ \t])+|\Z|(?=[\["()<>@,;:\\".\[\]]))|" (?:[^\"\r\\]|\\.|(?:(?:\r
\n)?[ \t]))*"(?:(?:\r\n)?[ \t])*))*@(?:(?:\r\n)?[ \t] )*(?:[^()<>@,;:\\".\[\] \000-\031]+(?:(?:(?:\r\n)?[ \t])+|\Z|(?=[\["()<>@,;:\\ ".\[\]]))|\[([^\[\]\r\\]|\\.)*\](?:(?:\r\n)?[ \t])
*)(?:\.(?:(?:\r\n)?[ \t])*(? :[^()<>@,;:\\".\[\] \000-\031]+(?:(?:(?:\r\n)?[ \t])+|\Z|(?=[\["()<>@,;:\\".\[ \]]))|\[([^\[\]\r\\]|\\.)*\](?:(?:\r\n)?[ \t])*))*|(?:[^()<>@,;:\\".\[\]
\000- \031]+(?:(?:(?:\r\n)?[ \t])+|\Z|(?=[\["()<>@,;:\\".\[\]]))|"(?:[^\"\r\\]|\\.|( ?:(?:\r\n)?[ \t]))*"(?:(?:\r\n)?[ \t])*)*\<(?:(?:\r\n)?[ \t])*(?:@(?:[^()<>@,; :\\".\[\] \000-
\031]+(?:(?:(?:\r\n)?[ \t])+|\Z|(?=[\["()<>@,;:\\".\[\]]))|\[([ ^\[\]\r\\]|\\.)*\](?:(?:\r\n)?[ \t])*)(?:\.(?:(?:\r\n)?[ \t])*(?:[^()<>@,;:\\" .\[\] \000-\031]+(?:(?:(?:\r\n)?[ \t])+|
\Z|(?=[\["()<>@,;:\\".\[\]]))|\[([^\[\ ]\r\\]|\\.)*\](?:(?:\r\n)?[ \t])*))*(?:,@(?:(?:\r\n)?[ \t])*(?:[^()<>@,;:\\".\ [\] \000-\031]+(?:(?:(?:\r\n)?[ \t])+|\Z|(?=[\["()<>@,;:\\".\
[\]]))|\[([^\[\]\ r\\]|\\.)*\](?:(?:\r\n)?[ \t])*)(?:\.(?:(?:\r\n)?[ \t])*(?:[^()<>@,;:\\".\[\] \000-\031]+(?:(?:(?:\r\n)?[ \t])+|\Z|(?=[\["()<>@,;:\\".\[\]]))|\[([^\[\]\r\\] |\\.)*\](?:
(?:\r\n)?[ \t])*))*)*:(?:(?:\r\n)?[ \t])*)?(?:[^()<>@,;:\\".\[\] \0 00-\031]+(?:(?:(?:\r\n)?[ \t])+|\Z|(?=[\["()<>@,;:\\".\[\]]))|"(?:[^\"\r\\]|\\ .|(?:(?:\r\n)?[ \t]))*"(?:(?:\r\n)?
[ \t])*)(?:\.(?:(?:\r\n)?[ \t])*(?:[^()<>@, ;:\\".\[\] \000-\031]+(?:(?:(?:\r\n)?[ \t])+|\Z|(?=[\["()<>@,;:\\".\[\]]))|"(? :[^\"\r\\]|\\.|(?:(?:\r\n)?[ \t]))*"(?:(?:\r\n)?[ \t])*))*@
(?:(?:\r\n)?[ \t])* (?:[^()<>@,;:\\".\[\] \000-\031]+(?:(?:(?:\r\n)?[ \t])+|\Z|(?=[\["()<>@,;:\\". \[\]]))|\[([^\[\]\r\\]|\\.)*\](?:(?:\r\n)?[ \t])*)(?:\.(?:(?:\r\n)?[ \t])*(?:[ ^()
<>@,;:\\".\[\] \000-\031]+(?:(?:(?:\r\n)?[ \t])+|\Z|(?=[\["()<>@,;:\\".\[\] ]))|\[([^\[\]\r\\]|\\.)*\](?:(?:\r\n)?[ \t])*))*\>(?:(?:\r\n)?[ \t])*)(?:,\s*( ?:(?:[^()<>@,;:\\".\[\]
\000-\031]+(?:(?:(?:\r\n)?[ \t])+|\Z|(?=[\["()<>@,;:\\ ".\[\]]))|"(?:[^\"\r\\]|\\.|(?:(?:\r\n)?[ \t]))*"(?:(?:\r\n)?[ \t])*)(?:\.(?:( ?:\r\n)?[ \t])*(?:[^()<>@,;:\\".\[\] \000-\031]+
(?:(?:(?:\r\n)?[ \t])+|\Z|(?=[ \["()<>@,;:\\".\[\]]))|"(?:[^\"\r\\]|\\.|(?:(?:\r\n)?[ \t]))*"(?:(?:\r\n)?[ \t ])*))*@(?:(?:\r\n)?[ \t])*(?:[^()<>@,;:\\".\[\] \000-\031]+(?:(?:(?:\r
\n)?[ \t ])+|\Z|(?=[\["()<>@,;:\\".\[\]]))|\[([^\[\]\r\\]|\\.)*\](?:(?:\r\n)?[ \t])*)(? :\.(?:(?:\r\n)?[ \t])*(?:[^()<>@,;:\\".\[\] \000-\031]+(?:(?:(?:\r\n)?[ \t])+| \Z|(?=[\["()
<>@,;:\\".\[\]]))|\[([^\[\]\r\\]|\\.)*\](?:(?:\r\n)?[ \t])*))*|(?: [^()<>@,;:\\".\[\] \000-\031]+(?:(?:(?:\r\n)?[ \t])+|\Z|(?=[\["()<>@,;:\\".\[\ ]]))|"(?:[^\"\r\\]|\\.|(?:(?:\r\n)?
[ \t]))*"(?:(?:\r\n)?[ \t])*)*\<(?:(?:\r\n) ?[ \t])*(?:@(?:[^()<>@,;:\\".\[\] \000-\031]+(?:(?:(?:\r\n)?[ \t])+|\Z|(?=[\[" ()<>@,;:\\".\[\]]))|\[([^\[\]\r\\]|\\.)*\](?:(?:\r\n)?[ \t])
*)(?:\.(?:(?:\r\n) ?[ \t])*(?:[^()<>@,;:\\".\[\] \000-\031]+(?:(?:(?:\r\n)?[ \t])+|\Z|(?=[\["()<> @,;:\\".\[\]]))|\[([^\[\]\r\\]|\\.)*\](?:(?:\r\n)?[ \t])*))*(?:,@(?:(?:\r\n)?[ \t])*
(?:[^()<>@,;:\\".\[\] \000-\031]+(?:(?:(?:\r\n)?[ \t])+|\Z|(?=[\["()<>@, ;:\\".\[\]]))|\[([^\[\]\r\\]|\\.)*\](?:(?:\r\n)?[ \t])*)(?:\.(?:(?:\r\n)?[ \t] )*(?:[^()<>@,;:\\".\[\] \000-
\031]+(?:(?:(?:\r\n)?[ \t])+|\Z|(?=[\["()<>@,;:\\ ".\[\]]))|\[([^\[\]\r\\]|\\.)*\](?:(?:\r\n)?[ \t])*))*)*:(?:(?:\r\n)?[ \t])*)? (?:[^()<>@,;:\\".\[\] \000-\031]+(?:(?:(?:\r\n)?[ \t])
+|\Z|(?=[\["()<>@,;:\\". \[\]]))|"(?:[^\"\r\\]|\\.|(?:(?:\r\n)?[ \t]))*"(?:(?:\r\n)?[ \t])*)(?:\.(?:(?: \r\n)?[ \t])*(?:[^()<>@,;:\\".\[\] \000-\031]+(?:(?:(?:\r\n)?[ \t])+|\Z|(?=[\
["()<>@,;:\\".\[\]]))|"(?:[^\"\r\\]|\\.|(?:(?:\r\n)?[ \t]))*"(?:(?:\r\n)?[ \t]) *))*@(?:(?:\r\n)?[ \t])*(?:[^()<>@,;:\\".\[\] \000-\031]+(?:(?:(?:\r\n)?[ \t]) +|\Z|(?=[\["()<>@,;:\
\".\[\]]))|\[([^\[\]\r\\]|\\.)*\](?:(?:\r\n)?[ \t])*)(?:\ .(?:(?:\r\n)?[ \t])*(?:[^()<>@,;:\\".\[\] \000-\031]+(?:(?:(?:\r\n)?[ \t])+|\Z |(?=[\["()<>@,;:\\".\[\]]))|\[([^\[\]\r\\]|\\.)*
\](?:(?:\r\n)?[ \t])*))*\>(?:( ?:\r\n)?[ \t])*))*)?;\s*)
```

This regular expression will only validate addresses that have had any comments stripped and replaced with whitespace

További probléma

- Kimenet egyetlen logikai változó
  - Illeszkedik-e vagy sem
- Mi hiányzik?

Hibajelzés!

Nyelvek és nyelvtanok

Formális nyelvek gyorstalpaló

Formális nyelv

- Fogalmak
  - Nyelv: lehetséges mondatok halmaza
  - Mondat: szimbólumok sorozata
  - Ábécé: lehetséges szimbólumok halmaza

Példa: Természetes számok

- Ábécé: {0, 1, 2, 3, 4, 5, 6, 7, 8, 9}
- Egyjegyű számok nyelve:
  - «0», «1», «2», «3», «4», «5», «6», «7», «8», «9»
- 15-nél kisebb páros számok nyelve:
  - «0», «2», «4», «6», «8», «10», «12», «14»
- Pozitív páros számok nyelve:
  - «0», «2», «4», «6», «8», «10», «12», «14», «16», «18», «20», «22», «24», «26», «28», …

Nyelvek megadása

- Elemek felsorolása
  - Csak véges elemszám esetén működik
  - Ld. páros számok nyelve
- Véges algoritmus felhasználása
  - Lépések, melynek a kimenetei a nyelv mondatai

Példa: Nevek felsorolása

- Nevek felsorolása:
  - Neveket vesszővel válasszuk el;
    - Kivéve az utolsót, ott és legyen közben
    - Pl.: Zoltán, István és Dániel
  - Ismétlődés megengedett
    - Pl.: Zoltán, Zoltán és Dániel

Példa: Nevek felsorolása

- Algoritmus
  1. Dániel, István, Zoltán nevek
  2. Egy név érvényes mondat
  3. Egy mondat, amelyet vessző és egy név követ, értelmes mondat
  4. Ha a befejezés előtt a mondat “, «név»” alakú, cseréljük ki “ és «név»” formájúra

«név» nem fog szerepelni a szövegben!

Probléma megoldás

- Kétféle szimbólum
  - Megjelenhet a szövegben (ábécéhez tartozik)
    - Terminális szimbólum
  - Nem jelenhet meg a szövegben
    - Nem-terminális szimbólum
- Nem-terminálisok
  - Cserélhetőek más szimbólumokkal (➔)
  - Algoritmus vége
    - Csak terminális szimbólum van
    - Nem cserélhetünk

Módosított algoritmus

- Ábécé
  - Dániel, István, Zoltán, és, “,”, sorvége
- Nem-terminális szimbólumok
  - «Név», «Mondat»

① Dániel ➔«Név»; István ➔«Név»; Zoltán ➔«Név»

② «Mondat» ➔ «Név»

③ «Mondat» ➔ «Mondat», «Né

④ «Mondat» végén , «Név» ➔ és «Név»

⑤ Kiindulás: «Mondat»

Bonyolult elemzési szabály

Nyelvek osztályozása

Nyelvek osztályozása

- Elemző bonyolult
  - Nem is lehetséges minden nyelvtanhoz
- Nyelvtanok kategorizálhatóak
  - Chomsky-féle nyelvosztályok

Rekurzívan felsorolható nyelvek (Type 0)

- Korlátozás nélküli szabályok
- Általános elemzés lehetetlen
  - Eldönthetetlenségi problémák
  - Speciális esetek működhetnek

Környezetfüggő nyelvek (Type 1)

- Kétféle definíció
  - Környezetfüggetlen
    - αAβ➔αγβ
    - α és β: kontextus
    - A: nem-terminális
    - γ: cserélendő érték
  - Monoton
    - Szabály jobboldala nem rövidebb, mint a baloldal
    - Szabály baloldala tartalmaz nem-terminálist

Az A nem-terminális kicserélhető γ-ra

Környezetfüggő nyelvek (Type 1)

- Elemzés
  - Monoton nyelvtanok
    - Felsorolhatjuk az összes mondatot
    - Adott hosszig
  - Lassú, de eldönthető

Környezetfüggetlen nyelvek (Type 2)

- Baloldal
  - A➔γ
    - Egy darab nem-terminális
    - Jobb oldal: terminális és nemterminálisok sorozata
  - Környezetfüggő nyelv, de kontextus üres

Környezetfüggetlen nyelvtanok (Type 2)

- Elemzés
  - Többféle megközelítés létezik
  - Jól ismert előnyök/hátrányok
  - Később részletesebben

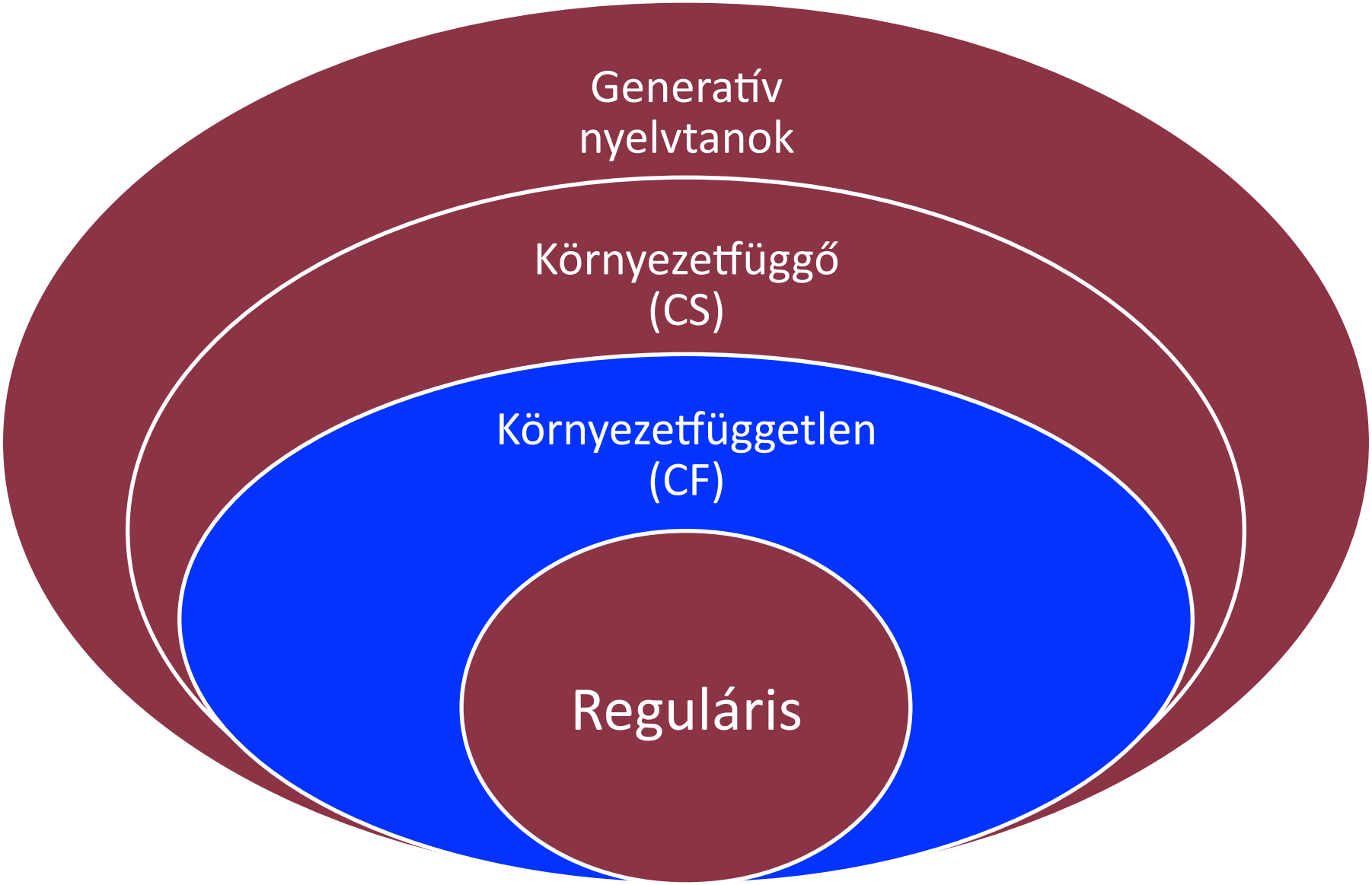
Reguláris nyelvek (Type 3)

- Kétféle szabály:
  - A→α
  - A→αB
- Kb. reguláris kifejezéseknek megfelelő
  - Implementációk valamivel többet engednek

Reguláris nyelvek (Type 3)

- Elemző: véges automata
- Egyszerű nyelvekre jó
- DE:
  - Beágyazás (nesting) nem megy
  - Ez tipikus probléma elemzők esetén

Nyelvosztályok hierarchiája



Nyelvek osztályozása

- Eddig:
  - Nyelvtanok osztályozása
- Nyelv:
  - Lehetséges nyelvtanok szerint
  - Legszigorúbb osztálynak megfelelő nyelvtan

Példa: Nevek felsorolása

- Ábécé
  - Dániel, István, Zoltán, és, “,”, sorvége
- Nem-terminális szimbólumok
  - «Név», «Mondat»

① Dániel ➔«Név»; István ➔«Név»;
Zoltán ➔«Név»

② «Mondat» ➔ «Név»

③ «Mondat» ➔ «Mondat», «Név»

④ «Mondat» végén , «Név» ➔ és «Név»

⑤ Kiindulás: «Mondat»

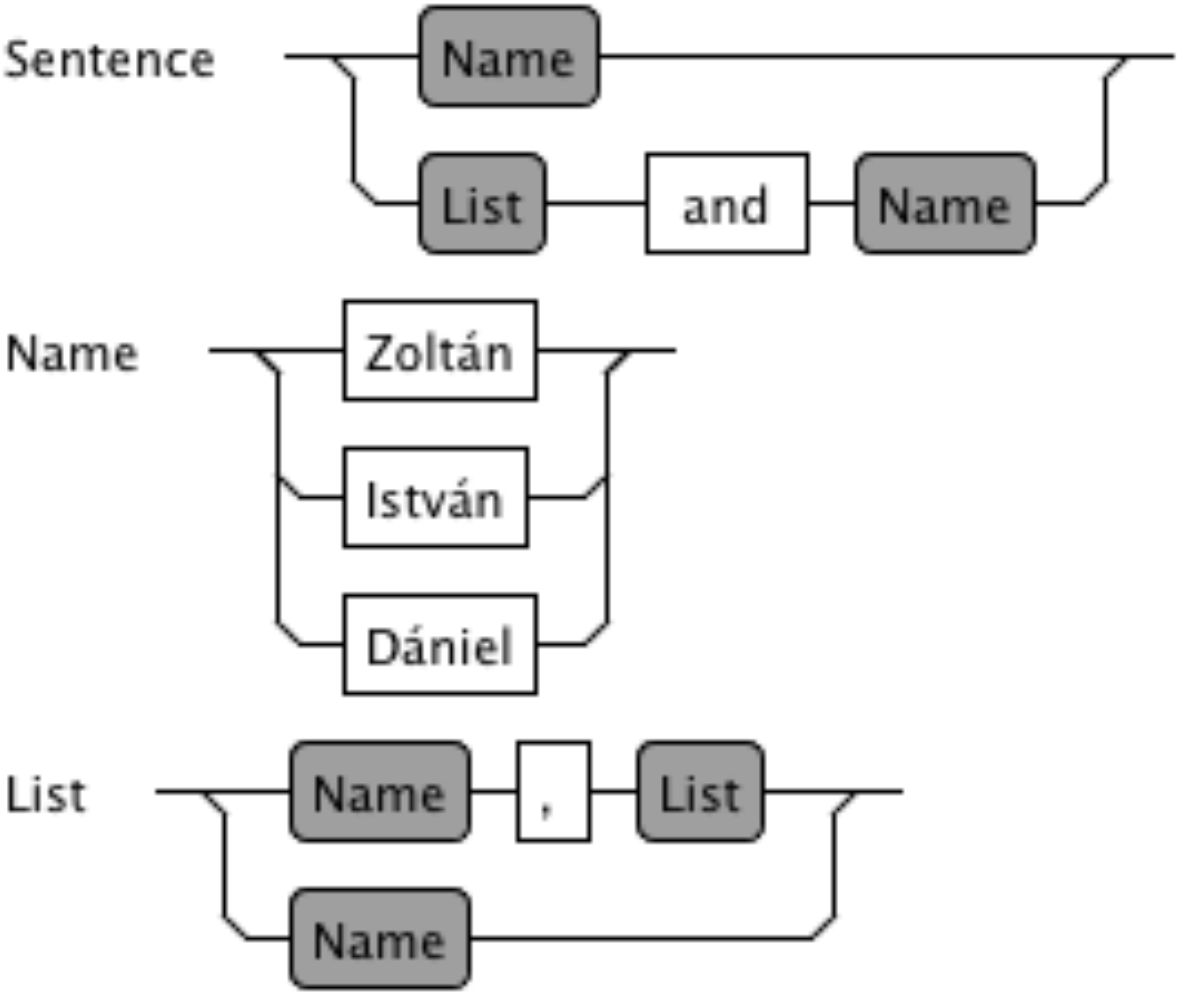
4. szabály:
0. nyelvosztály

Példa: CF nyelvtan a nevekhez

- Ábécé
  - Dániel, István, Zoltán, és, “,”, sorvége
- Nem-terminális szimbólumok
  - «Név», «Mondat», «Lista»

```
«Név»::=  Zoltán | István | Dániel
«Mondat»::= «Név» | «Lista» és «Név»
«Lista»::= «Lista», «Név»| «Név»
```

Railroad diagram



CF nyelvek elemzése

Nyelvtanok alkalmazása

- Szabályok alapján behelyettesítés
- Indeterminisztikus választás
  - Elemzési algoritmusok leköthetik!
- Cél:
  - Szabályok sorozatával előállítani a bemenetet

Példa: Levezetés

«Név»::= Zoltán | István | Dániel

«Mondat»::= «Név» | «Lista» és «Név»

«Lista»::= «Lista», «Név» | «Név»

Mondatszerű forma	Felhasznált szabály
«Mondat»	

Példa: Levezetés

«Név»::= Zoltán | István | Dániel

«Mondat»::= «Név» | «Lista» és «Név»

«Lista»::= «Lista», «Név» | «Név»

Mondatszerű forma	Felhasznált szabály
«Mondat»	
«Lista» és «Név»	«Mondat» ->«Lista» és «Név»

Példa: Levezetés

«Név»::= Zoltán | István | Dániel

«Mondat»::= «Név» | «Lista» és «Név»

«Lista»::= «Lista», «Név» | «Név»

Mondatszerű forma	Felhasznált szabály
«Mondat»	
«Lista» és «Név»	«Mondat» ->«Lista» és «Név»
«Lista», «Név» és «Név»	«Lista» -> «Lista», «Név»

Példa: Levezetés

«Név»::= Zoltán | István | Dániel

«Mondat»::= «Név» | «Lista» és «Név»

«Lista»::= «Lista», «Név» | «Név»

Mondatszerű forma	Felhasznált szabály
«Mondat»	
«Lista» és «Név»	«Mondat» ->«Lista» és «Név»
«Lista», «Név» és «Név»	«Lista» -> «Lista», «Név»
«Név», «Név» és «Név»	«Lista» -> «Név»

Példa: Levezetés

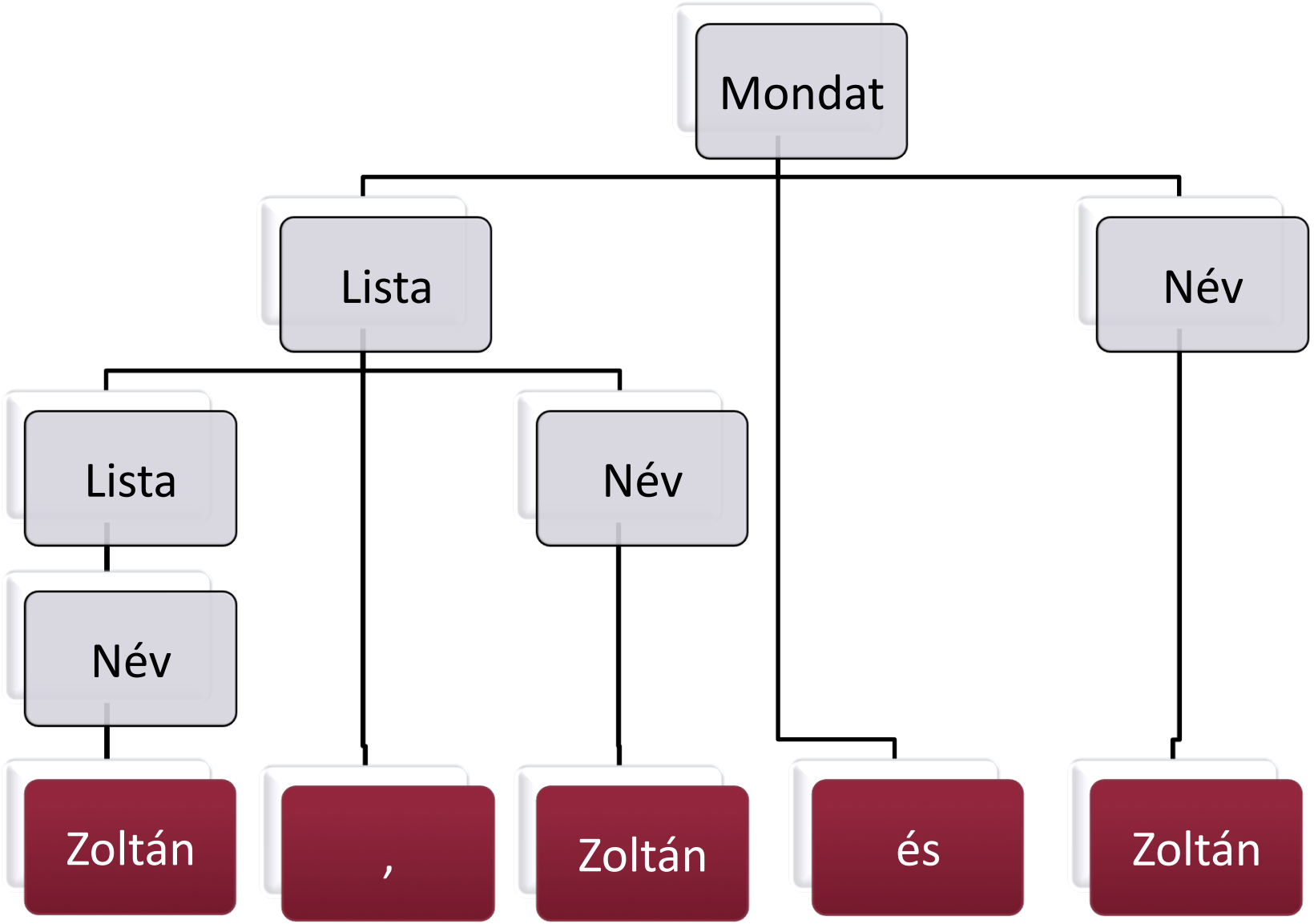
«Név»::= Zoltán | István | Dániel

«Mondat»::= «Név» | «Lista» és «Név»

«Lista»::= «Lista», «Név» | «Név»

Mondatszerű forma	Felhasznált szabály
«Mondat»	
«Lista» és «Név»	«Mondat» ->«Lista» és «Név»
«Lista», «Név» és «Név»	«Lista» -> «Lista», «Név»
«Név», «Név» és «Név»	«Lista» -> «Név»
Zoltán, Zoltán és Zoltán	

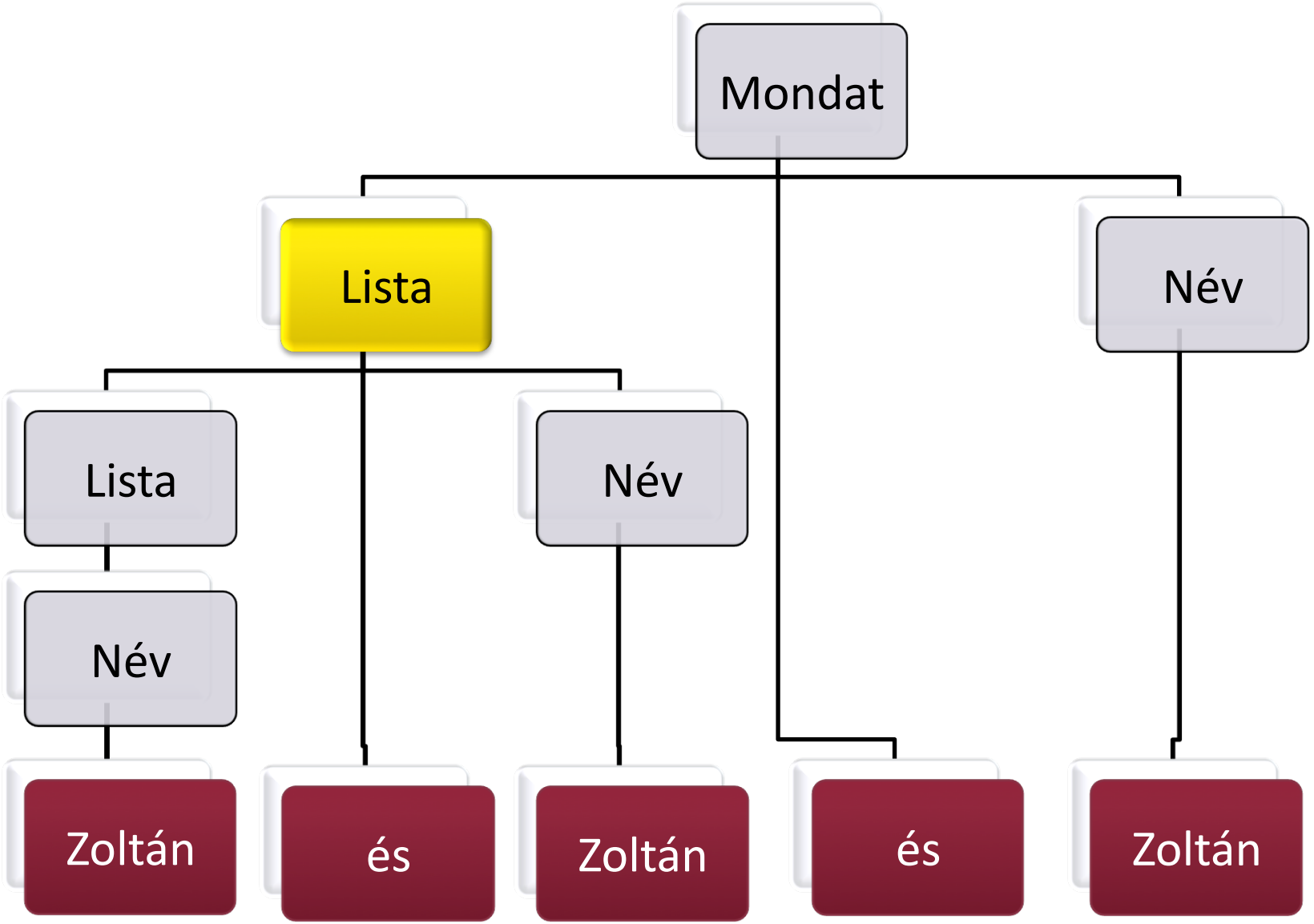
Levezetési fa



Elemzés

- Cél:
  - Levezetési fa meghatározása
  - Miért kell a levezetési fa?

Hibajelzés



Elemzés

- Cél:
  - Levezetési fa meghatározása
  - Miért kell a levezetési fa?
- Megközelítések
  - Nem irányított módszerek (Non-directed methods)
  - Irányított módszerek (Directed methods)
  - Keresési technikák
  - **Felülről lefelé**
  - **Alulról felfelé**

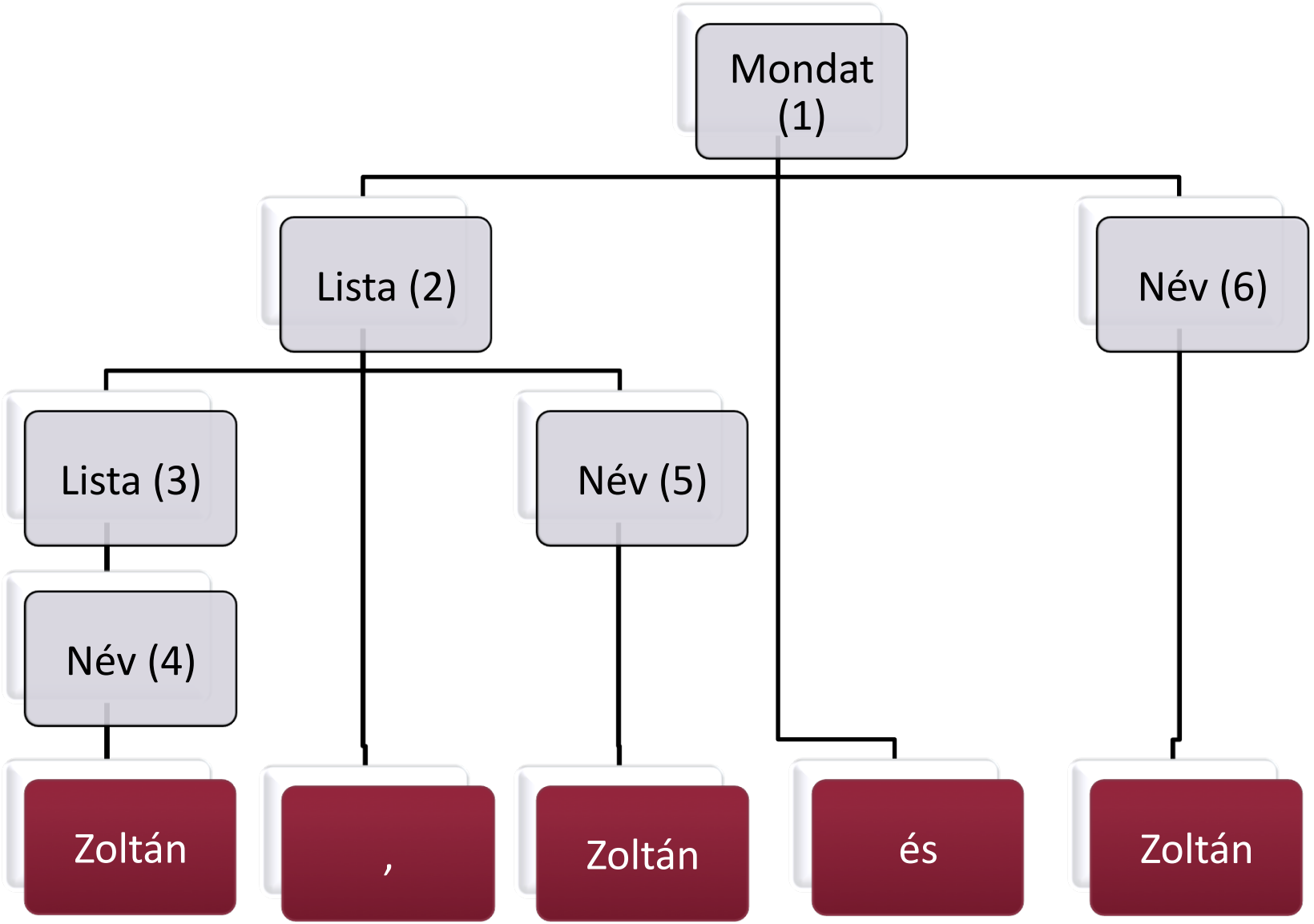
Felülről lefelé elemzés

- Kiindulás: Mondatszimbólum
- Cél: szabályalkalmazásokkal eljutni a mondatig
- Elágazási lehetőségek
  - Predict and match
  - Backtracking szükség esetén

LL(k) elemzők

- Tulajdonságok
  - Left-to-right (balról jobbra olvasás)
  - Leftmost derivation (jobboldali levezetés)
  - k character look-ahead (előretekintés k karakterrel)

Baloldali levezetés



Alulról felfelé elemzés

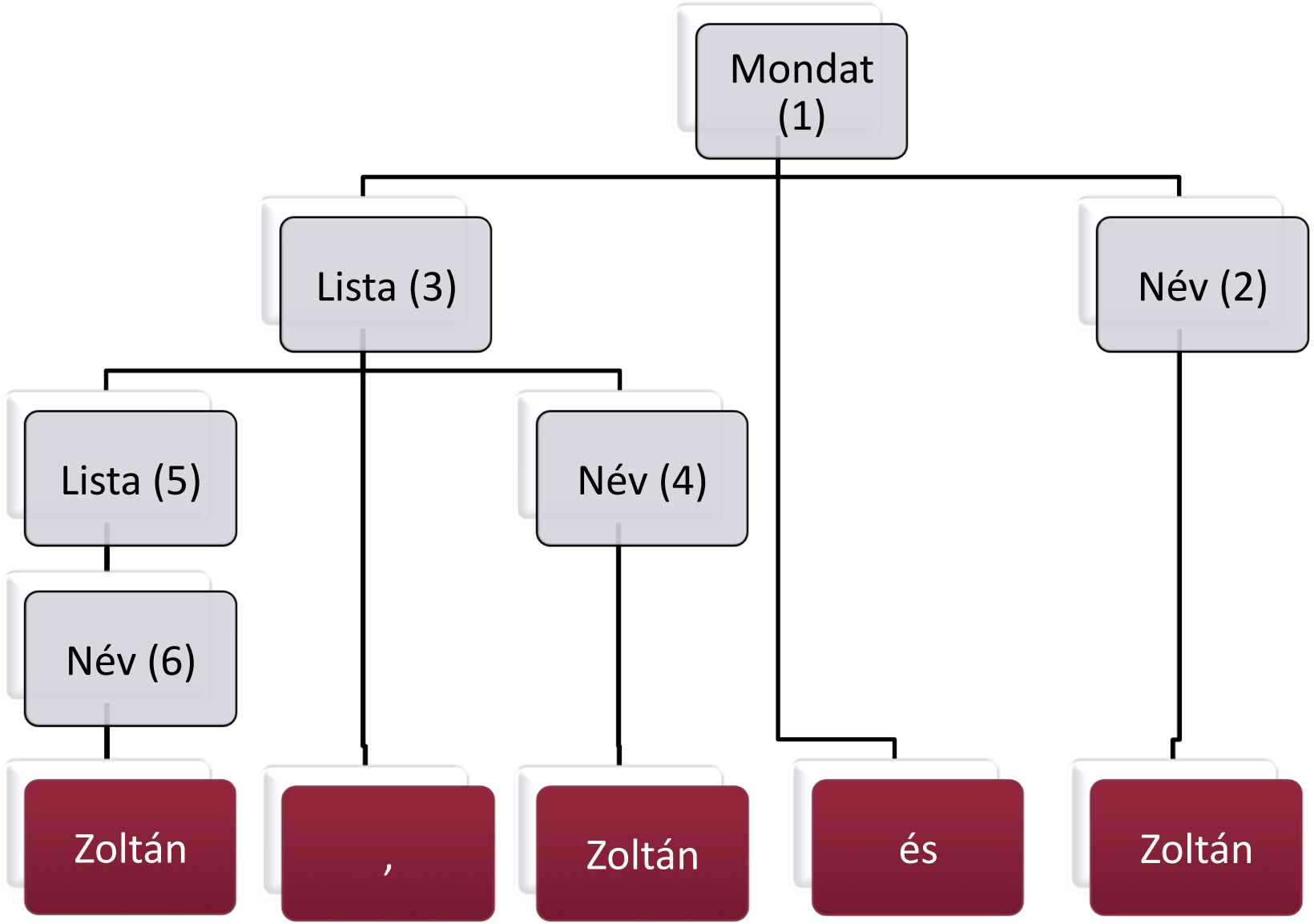
- Kiindulás: Mondat
- Cél: Eljutni a Mondatszimbólumig

- Szabály jobb oldalát illesztjük a szövegre
  - Illeszkedés esetén alkalmazhatjuk a szabályt
  - Visszafelé!

LR(k) elemzők

- Tulajdonságok
  - Left-to-right
  - Rightmost derivation
  - k character look-ahead

Jobboldali levezetés



Elemzők választása

- $LR(k) \subset LL(k)$
- $LR(k) \subset CF$

- Ötletek
  - LL(k) egyszerűbb algoritmus
  - LR(k) nagyobb kifejezőerő