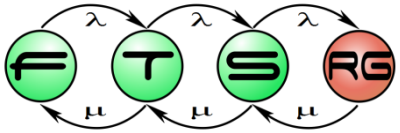
Szöveges editorok készítése II.

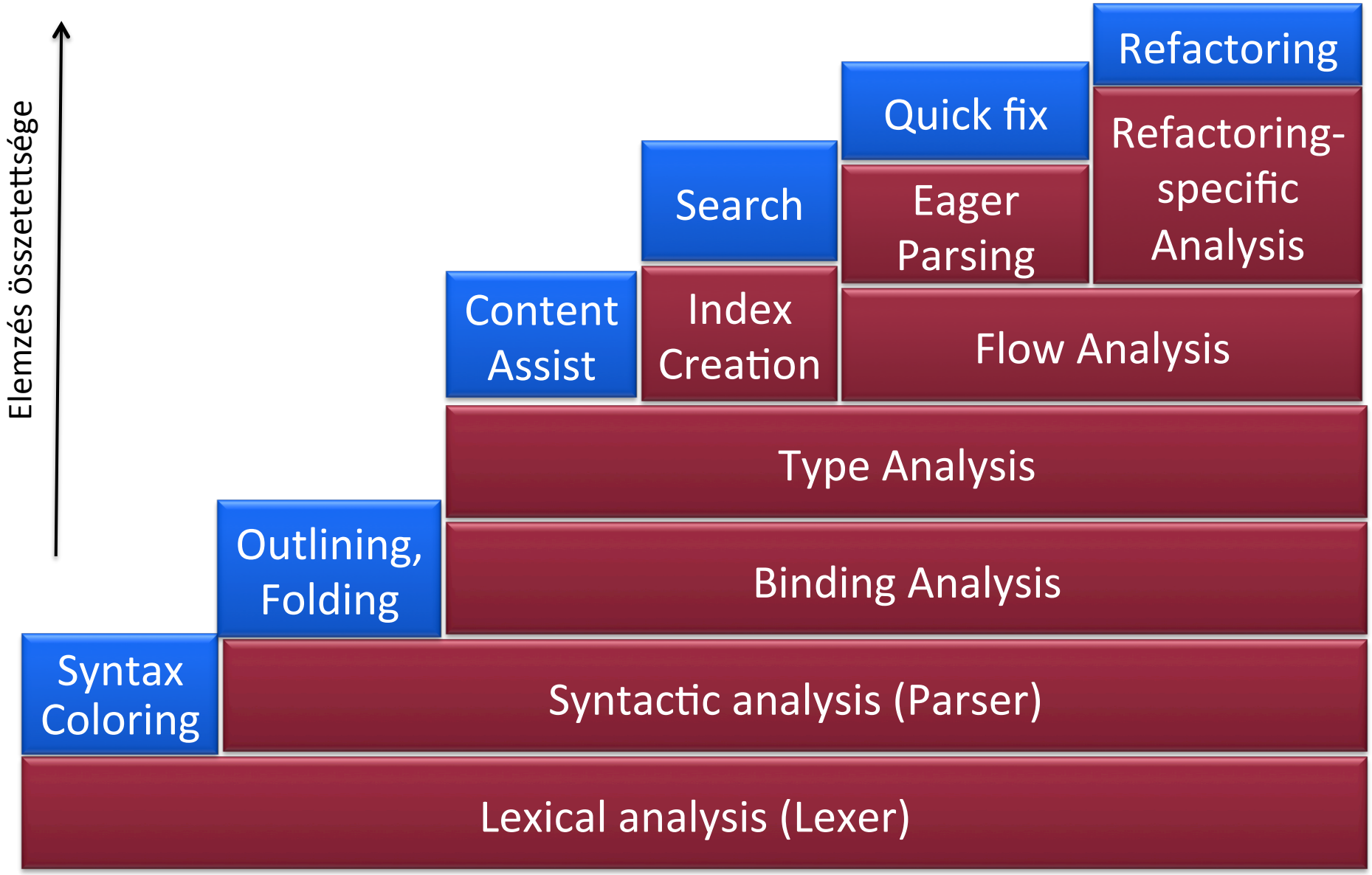
IMP, EMFText, Xtext



Ismétlés

- Szöveges editorok integrált fejlesztőeszközökben
  - Összetett funkcionalitás
  - Különböző szolgáltatások

Elemzés fázisai és editor szolgáltatások



Eszközök

- IDE Meta-tooling Platform (IMP)
- Xtext
- EMFText

IMP (IDE Meta-tooling Platform)

IMP (IDE Meta-tooling Platform)

- Eclipse alprojekt
- Cél
  - Tetszőleges szöveges nyelvhez IDE támogatás
- Megközelítés
  - Univerzális editor
  - Nyelv megadása szolgáltatásokkal
    - Minden szolgáltatás Eclipse kiterjesztés

Elemző, mint szolgáltatás

- Elemző feladata:
  - Karakterfolyam beolvasása
  - Kimenet:
    - AST
    - Hibaüzenetek
- Generikus kezelés
  - AST változtatás nélkül továbbadható
  - Hibaüzeneteket be kell csomagolni

AST kezelés

- Probléma
  - Honnan ismerik a szolgáltatások az AST szerkezetét?
- Válasz
  - Generikusan nem ismerik
  - A szolgáltatás implementációjakor a fejlesztő tudja
    - Eredmény:
      - API sokszor Java objektumokat vár
      - Kód pontos típuskonverziókkal indul (downcast)

IMP szolgáltatások

- Sourcelocator
  - AST -> forrás kapcsolat megadása
- Token Coloring
  - Forráskódszínezés
- Outline
  - Model builder
  - Label Provider
- Source folding

IMP szolgáltatások II.

- Reference resolution
  - Hivatkozásoknak kiszámolja a célját
  - Kritikus funkcionalitás
    - Sok szolgáltatás használja
    - Sokszor fut
  - Nincs különösebb segítség az implementációhoz!
    - AST bejárás

IMP szolgáltatások – III.

- Hover help
  - SourceLocatort használja
  - Referenciákat is követi
  - Kijelölt elemhez kiszámítja a súgó szövegét
- Hyperlinking
  - Összeköti a forrás- és cél elemeket
  - Csak referenciák esetén
- Content Assist
  - Kézzel össze kell gyűjteni a lehetséges folytatásokat

IMP szolgáltatások – IV.

- És még sok más
  - Pretty printing BOX nyelvű kényszerekkel
  - Builder (compiler) támogatás
  - Beállítások ablak

IMP tooling

- Van fejlesztési támogatás
  - LPG parser generátorra szabva
  - Viszonylag kényelmes szerkesztés
  - Van még néhány DSL a fejlesztés megkönnyítésére
  - … de alapvetően Java kódolás

IMP - Összefoglalás

- Univerzális editor koncepció
  - Nyelv megadása szolgáltatásokkal
  - Szolgáltatások megadása Java kóddal
- Probléma
  - A projekt lassan fejlődik
  - Utolsó kiadás 2010. augusztusi
    - Nem halott projekt – SVN repo frissül

EMFText

EMFText

- Szöveges editor technológia
  - http://www.emftext.org
  - Kiindulás: létező EMF metamodell (absztrakt szintaxis)
  - Cél: szöveges konkrét szintaxis
  - Képes automatikusan nyelvtant származtatni
    - Több stílusban (Java, HUTN…)
- Logika: nagyon hasonlít Xtexthez
  - Csak rövid ismertető

Nyelvtan megadása

- Minden metamodell elemhez egy szabály
  - Baloldal: metamodell elem
  - Jobboldal: karakterláncok és EMF feature nevek
- Pl.:
- ```
  SocialNetwork ::= "SocialNetwork"
  "{" ("entities" ":" entities |
  "acquaintances" ":" acquaintances)*
  "}";
  ```
  ```

Fejlettebb lehetőségek

- Quick fix támogatás
- Builder támogatás
- Verem-alapú interpreter váz
  - Debugger támogatással

JaMoPP

- Java 5 metamodell és parser
  - EMFText alapokon
- Java alapú nyelvekre
  - **Figyelem!** Elég komplex nyelvtan/modell

Syntax Zoo

- Bőséges nyelvdemonstráció a honlapon
  - Ontológia nyelvek
  - Modellező nyelvek
  - DSL-ek
  - …
  - (2012. 11. 05.: 100 nyelv)

Xtext

Xtext

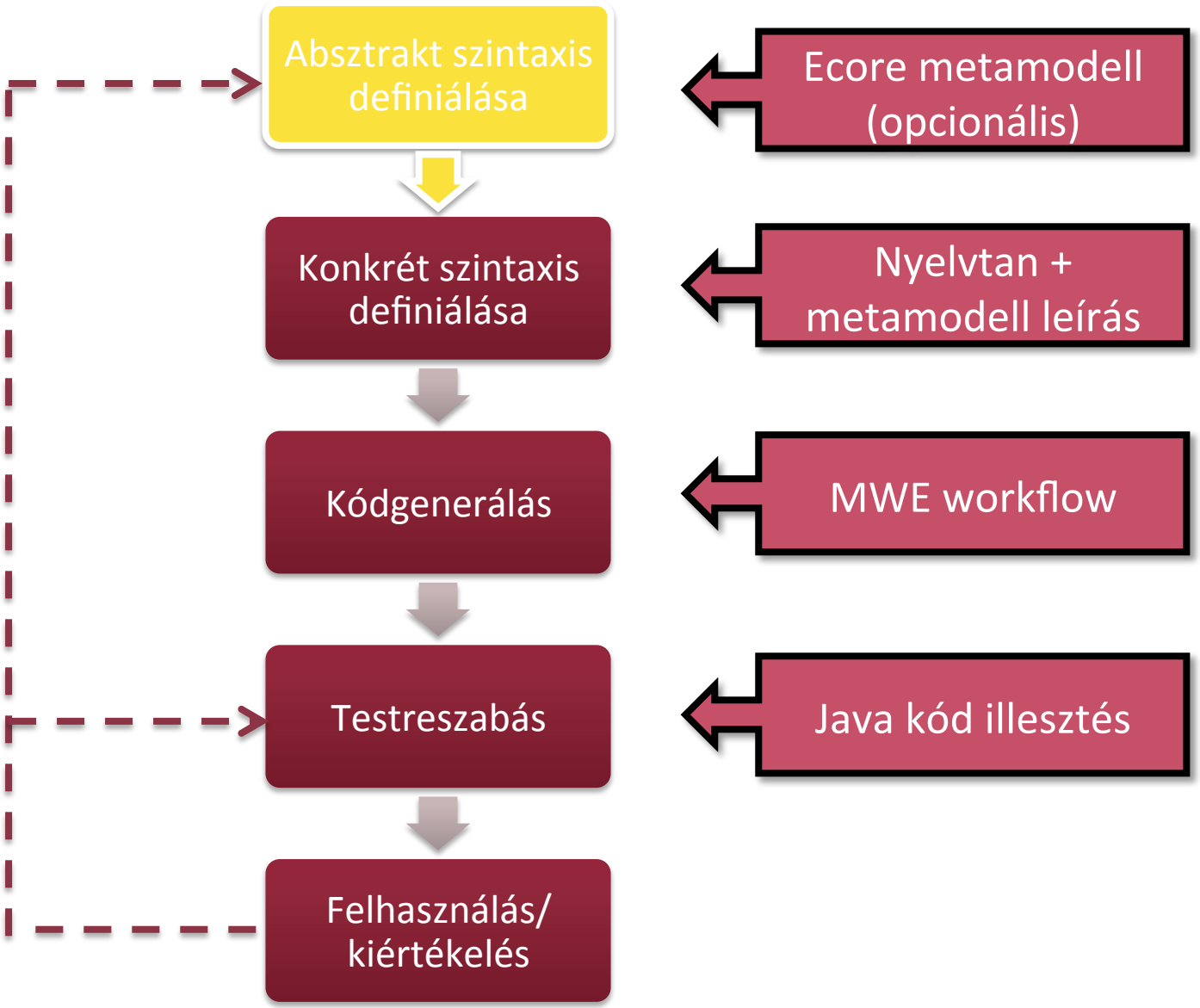
"A DSL for writing DSLs"

Peter Friese, Itemis

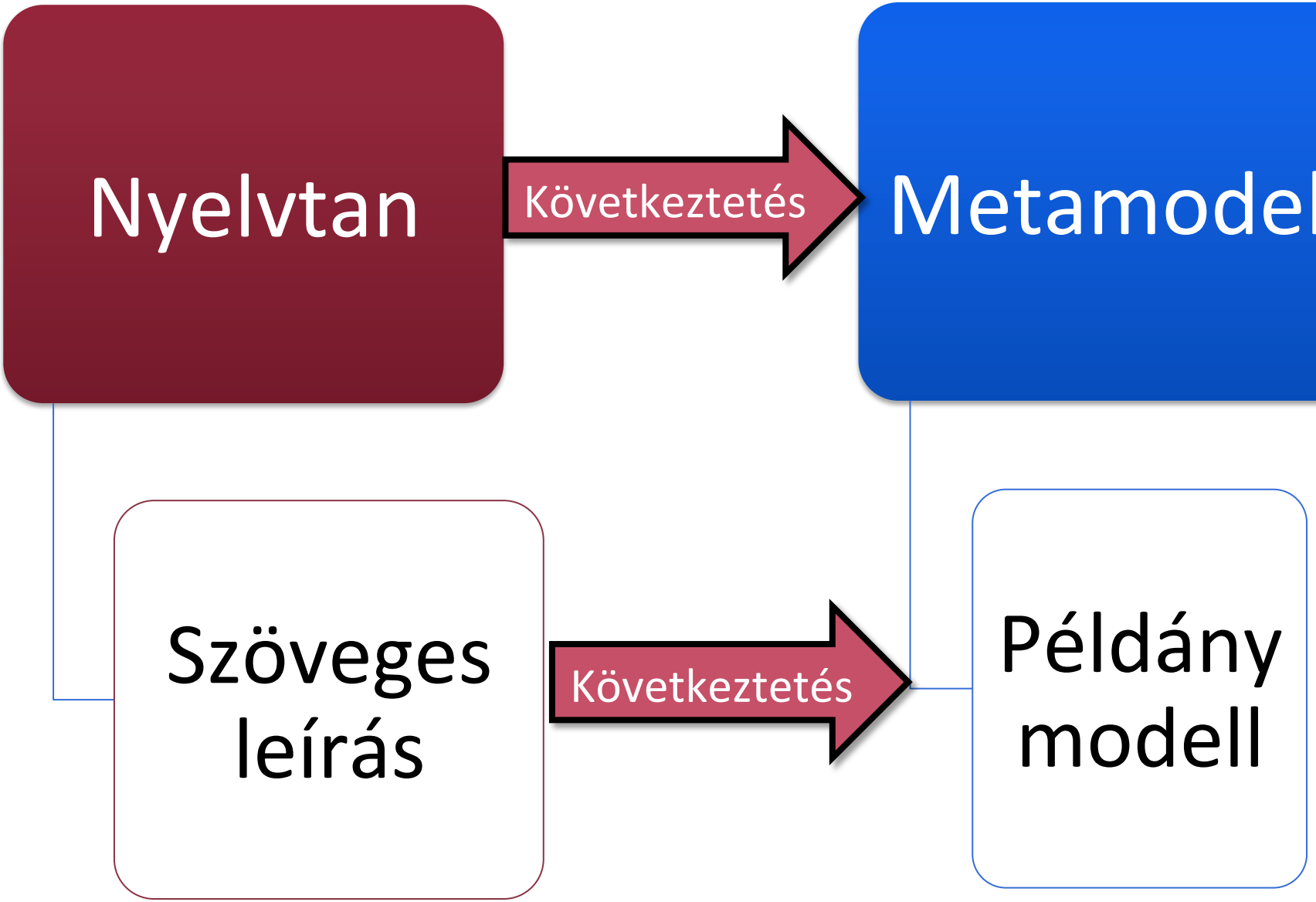
Xtext

- Cél
  - Integrált szöveges editor
  - EMF alapú AST
  - Kódgenerálási támogatás
    - Editorhoz
    - Nyelvhez

Felhasználás menete



Nyelvek és modellek



Konkrét szintaxis megadása

Nyelvtan leírása

- Domain-specifikus nyelv (.xtext fájl)
  - EBNF nyelvtanspecifikáció
  - Kiegészítésekkel
    - További nyelvtan elemek
    - Metamodell generálás pontosítása

Konkrét szintaxis megadása

- Parser generátor
  - Antlr
  - (Packrat parser generátor)
- LL(*) nyelvtanok támogatása
  - Jó hibajelző képesség
- Ecore metamodell automatikus származtatása

LL(*) nyelvtanok

- Múlt héten: LL(k)
  - Balról jobbra olvasás
  - Baloldali levezetés
  - k karakter előretekintés
- LL(*)
  - Balról jobbra olvasás
  - Baloldali levezetés
  - Korlátlan hosszú előretekintés
    - Véges automata használata

Elemzés menete

- Kiindulás
  - Mondatszimbólum
- Lépések
  - Szabályalkalmazások választása
    - Predict and match
    - Ehhez kell az előretekintés, hogy determinisztikus legyen
- Cél
  - A tényleges mondatig eljutni

Probléma: balrekurzív nyelvek

Probléma: balrekurzív nyelvek

```
Expression:
 Expression ‘*’ Expression |
 Expression ‘+’ Expression |
 INT;
```

- Levezetés automatikusan
  - Expression
  - Expression ‘+’ Expression
  - Expression ‘+’ Expression ‘+’ Expression
  - …

Probléma: balrekurzív nyelvek

```
Expression:
 Expression ‘*’ Expression |
 Expression ‘+’ Expression |
 INT;
```

- Levezetés automatikusan
  - Expression
  - Expression ‘+’ Expression
  - Expression ‘+’ Expression ‘+’ Expression
  - …

Probléma: balrekurzív nyelvek

```
Expression:
 Expression ‘*’ Expression |
 Expression ‘+’ Expression |
 INT;
```

- Levezetés automatikusan
  - Expression
  - Expression ‘+’ Expression
  - Expression ‘+’ Expression ‘+’ Expression
  - …

Probléma: balrekurzív nyelvek

```
Expression:
 Expression ‘*’ Expression |
 Expression ‘+’ Expression |
 INT;
```

- Levezetés automatikusan
  - Expression
  - Expression ‘+’ Expression
  - Expression ‘+’ Expression ‘+’ Expression
  - …

Probléma: balrekurzív nyelvek

```
Expression:
 Expression ‘*’ Expression |
 Expression ‘+’ Expression |
 INT;
```

- Levezetés automatikusan
  - Expression
  - Expression ‘+’ Expression
  - Expression ‘+’ Expression ‘+’ Expression
  - …

Balrekurzív nyelvtanok

- Balrekurzív nyelvtanok
  - Végtelen hosszú szabályalkalmazássorozat lehetséges
  - Az elemző nem automatikusan tud választani
  - Nem bal-elemezhetőek
- Megoldás
  - Balrekurzió megszüntetése (left-factoring)

Balrekurzió megszüntetése

- További nem-terminálisok bevezetése
  - Szabályok során balrekurzió elkerülése

Példa: nem balrekurzív műveletek

```
Expression:

 Expression ‘*’ Expression |

 Expression ‘+’ Expression |

 INT;
```

Példa: nem balrekurzív műveletek

```
Addition:
 Multiplication ('+' right=Multiplication)*;

Multiplication:
 NumberLiteral ('*' right=NumberLiteral)*;

NumberLiteral:
 value=INT;
```

Mi változott még?

- Explicit műveleti sorrend!
  - Az új nyelvtanban a szorzás magasabb precedenciájú
  - Hozzá lehetne adni zárójelezést

Példa

```
Addition:
 Multiplication ('+' right=Multiplication)*;

Multiplication:
 Primary ('*' right=Primary)*;

Primary:
 NumberLiteral |
 '(' Addition ')';

NumberLiteral:
 value=INT;
```

Mi kell még

- Metamodell következtetés
- Visszaadott érték típusának megadása
- Átalakítási szabályok

AST típusok

- Ha nem adjuk meg, egyszerűen rövidítésnek veszi:
  - `NumberLiteral : … ;`
  - `NumberLiteral returns NumberLiteral : …;`

- Felhasználható arra, hogy több nyelvi elem azonos típust adjon vissza (rekurzív hierarchia)

Típusok

```
Addition returns Expression:
 Multiplication ('+' right=Multiplication)*;

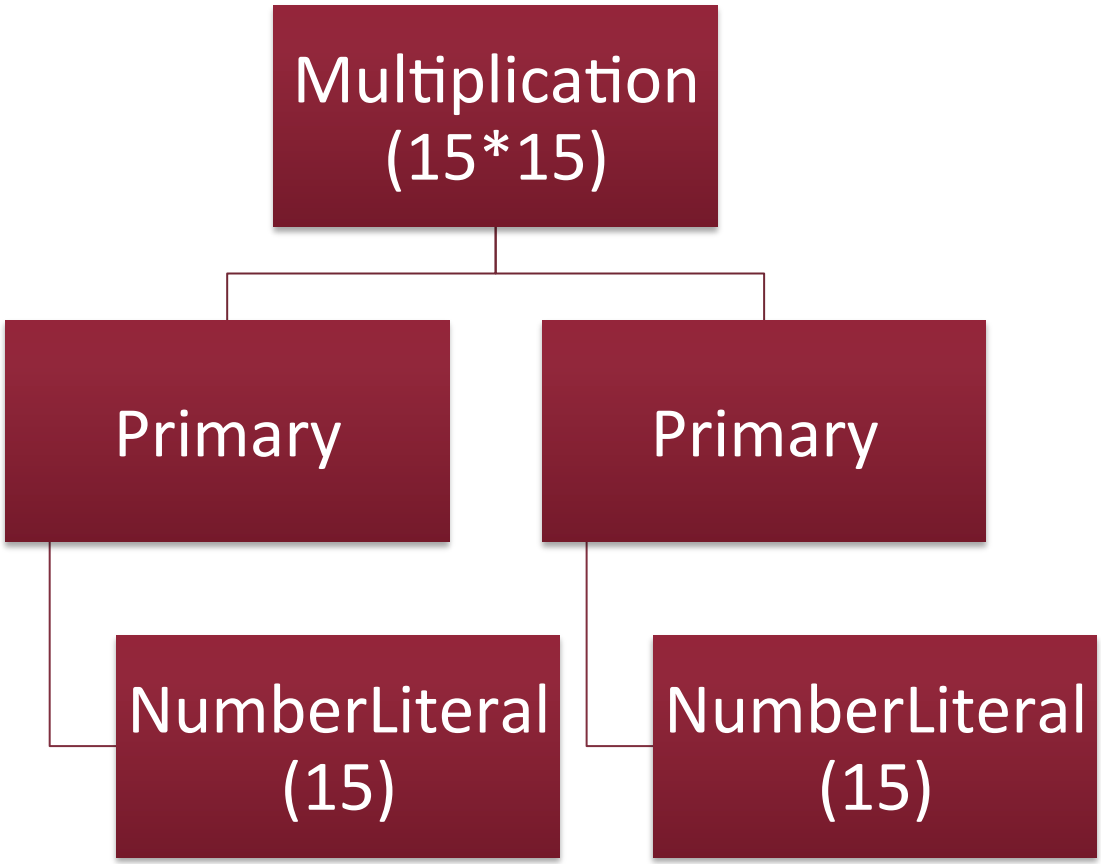
Multiplication returns Expression:
 Primary ('*' right=Primary)*;

Primary returns Expression:
 NumberLiteral |
 '(' Addition ')';

NumberLiteral:
 value=INT;
```

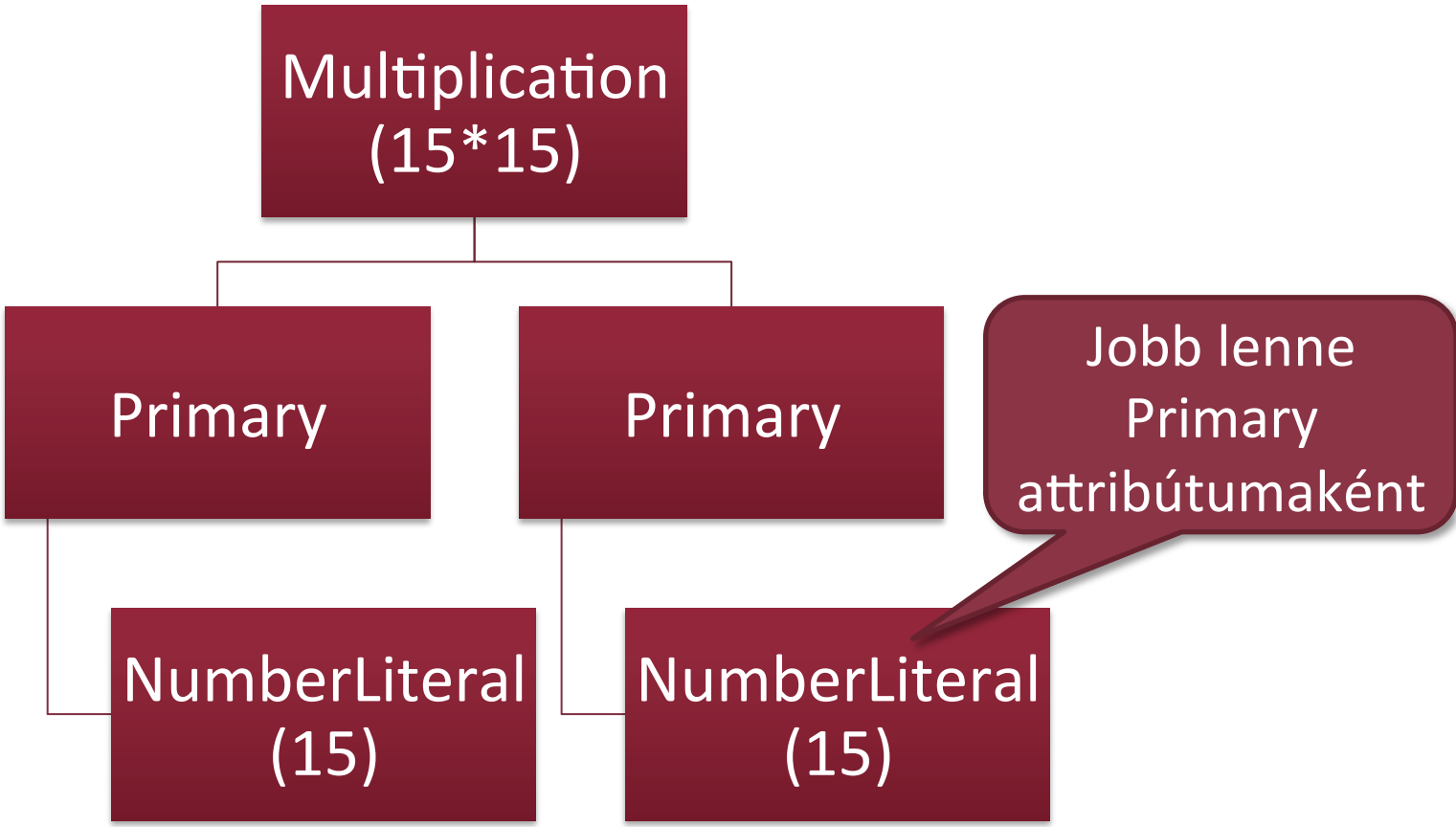
Átalakítási szabályok

- AST helyenként túl bőbeszédű
  - Left-factoring behoz ilyet



Átalakítási szabályok

- AST helyenként túl bőbeszédű
  - Left-factoring behoz ilyet

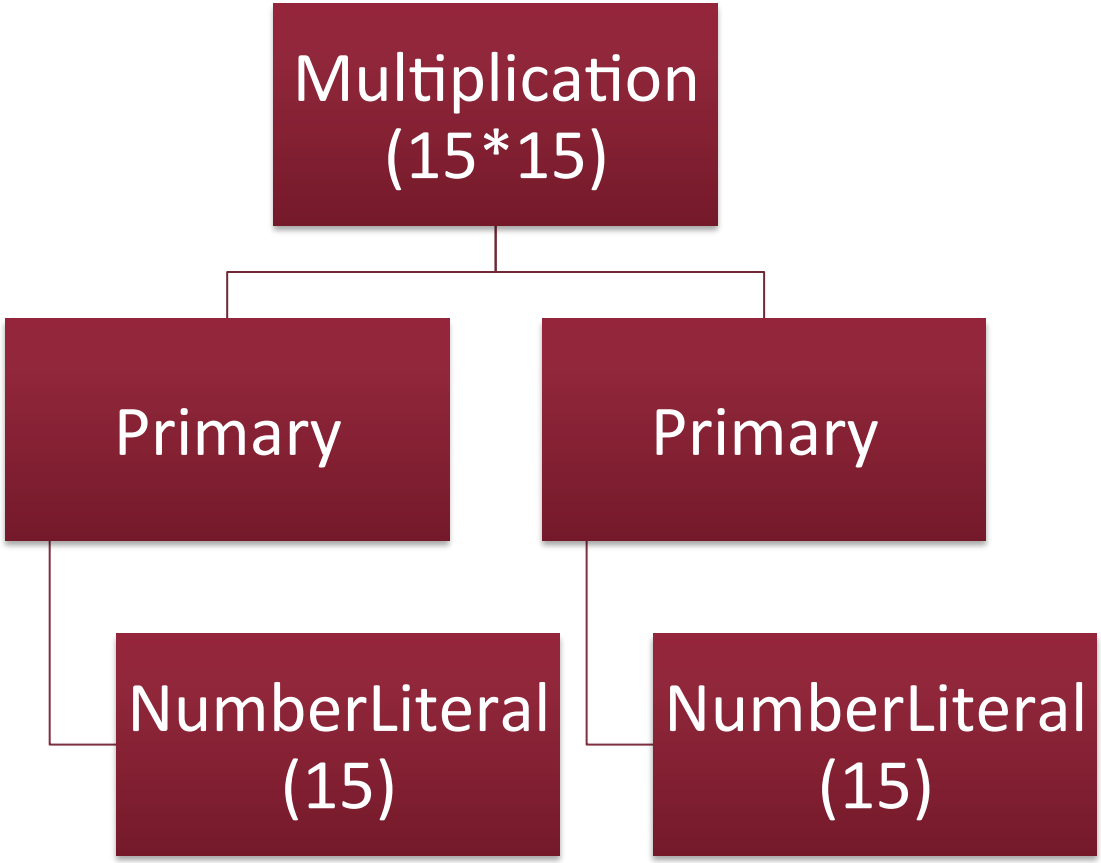


Átalakítási szabályok

- Hivatkozásokhoz szabályok illeszthetőek
  - Metamodell következtetést befolyásolják
  - Egyszerű átalakításokra elegendőek
  - DE: bonyolítják a nyelvtant
- Tipikus szabály
  - Megkapunk egy node-ot, és felhasználjuk

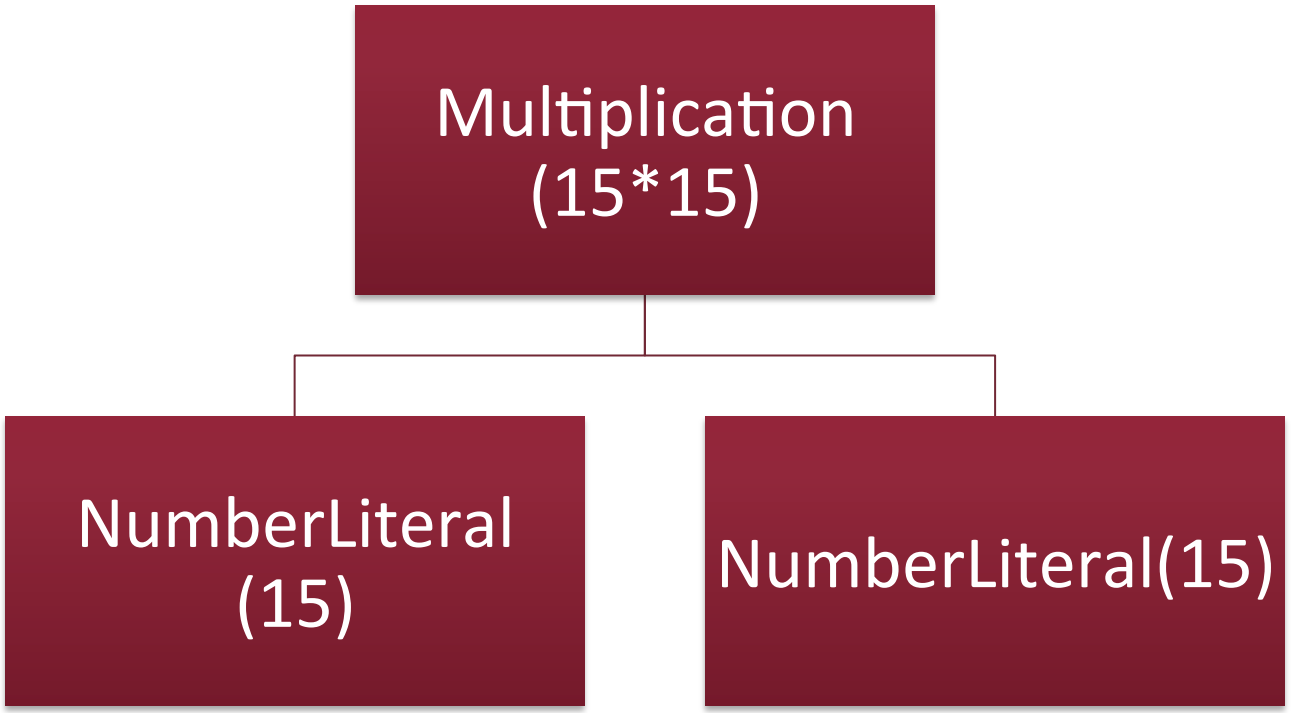
Átalakítási szabályok

```
Multiplication returns Expression:
 Primary ({Multiplication.left=current} '*'
right=Primary)*;
```



Átalakítási szabályok

```
Multiplication returns Expression:
 Primary ({Multiplication.left=current} '*'
right=Primary)*;
```



Példa

```
Addition returns Expression:
 Multiplication ({Addition.left=current} '+'
right=Multiplication)*;

Multiplication returns Expression:
 Primary ({Multiplication.left=current} '*'
right=Primary)*;

Primary returns Expression:
 NumberLiteral |
 '(' Addition ')';

NumberLiteral:
 value=INT;
```

További nyelvi elemek

- Felsorolás (EEnum)

```
enum Sex returns Sex:
     male = 'male' | female = 'female';
```

- Szabad sorrendű elemek

```
Modifier:
static?='static'? & final?='final'? &
visibility=Visibility;

enum Visibility:
PUBLIC='public' | PRIVATE='private' |
PROTECTED='protected';
```

További nyelvtan elemek

- Until token
  - terminal ML_COMMENT : '/*' -> '*/';
- Wildcard
  - FOO : 'f' . 'o';
- Tartomány
  - terminal INT returns ecore::EInt: ('0'..'9')+;
- ... (dokumentáció)

Kódgenerálás

Kódgenerálás

- Van egy nyelvtanunk
  - +metamodell információk
- Kell egy szerkesztő
  - Kódgenerálás
    - Ecore metamodell generálás
    - Ecore generátor modellből kód generálás
    - Parser generálás
    - Editor komponensek generálása
    - …

- Bonyolult folyamat
  - Lépésekre bontható
  - Újabb DSL
    - Modellezési munkafolyamat leírására
    - Modeling Workflow Engine (MWE)

Az Xtext workflow

- Generátor fragmensek adhatóak meg
  - Ecore generátor
  - Ecore importer
  - Antlr generátor hívása
  - …
  - Egyesével paraméterezhető
- Jó alapértelmezett generátort kapunk
  - Ritkán kell hozzányúlni

Példa

```
component = Generator {
        pathRtProject = runtimeProject
        pathUiProject = "${runtimeProject}.ui"
        projectNameRt = projectName
        projectNameUi = "${projectName}.ui"
        language = {
                uri = grammarURI
                fileExtensions = file.extensions

                // Java API to access grammar elements
(required by several other fragments)
                fragment = grammarAccess.GrammarAccessFragment {}
                // generates Java API for the generated EPackages
                fragment = ecore.EcoreGeneratorFragment {
                // referencedGenModels = "uri to genmodel, uri
to next genmodel"
                }
```

Példa - Folytatás

```
                    // a custom ResourceFactory for use with EMF
                    fragment = resourceFactory.ResourceFactoryFragment {
                            fileExtensions = file.extensions
                    }

                    // The antlr parser generator fragment.
                    fragment =
parser.antlr.XtextAntlrGeneratorFragment {
                    //  options = {
                    //              backtrack = true
                    //      }
                    }
...
```

Generátor futása után

- Kapunk egy működő szerkesztőt
  - Forráskód színezés
  - Outline
  - Content assist
  - Hivatkozásfeloldás
  - …
- Mindenből alapértelmezett implementáció
  - Implementációs részletekről két hét múlva

Összegzés

Melyiket használjuk?

	IMP	EMFText	Xtext
Elemző/AST	Tetszőleges parser technológia; tetszőleges AST	Antlr – LL(*); EMF alapú AST	Antlr – LL(*); EMF alapú AST
Megközelítés	Univerzális editor	Generált editor	Generált editor
Előny	Széles körű felhasználhatóság; létező parser újrahasznosítása	Language Zoo Pl. Java vagy ontológia nyelvek	Nagyon jól bővíthető; nagyon aktív fejlesztés
Hátrány	Minimális automatizáció	Korlátozott nyelvtan	Bonyolult lehet