Language Convergence Infrastructure

Vadim Zaytsev
Software Languages Team
UNIVERSITÄT KOBLENZ•LANDAU
http://twitter.com/grammarware

Motivation

- Does Java source code relates correctly to the model?
- Is the class system serialisable to a standard schema?
- Do a code analyser and a compiler agree on a dialect?
- Which compiler compiler is better?
- Are language documentation claims true?
- Do two idiosyncratic grammars agree on a language?

Approach

- Grammar convergence idea:
  - extract grammars
  - compare grammars
  - transform grammars
- Grammar convergence methodology published as Software Languages Team work with Ralf Lämmel

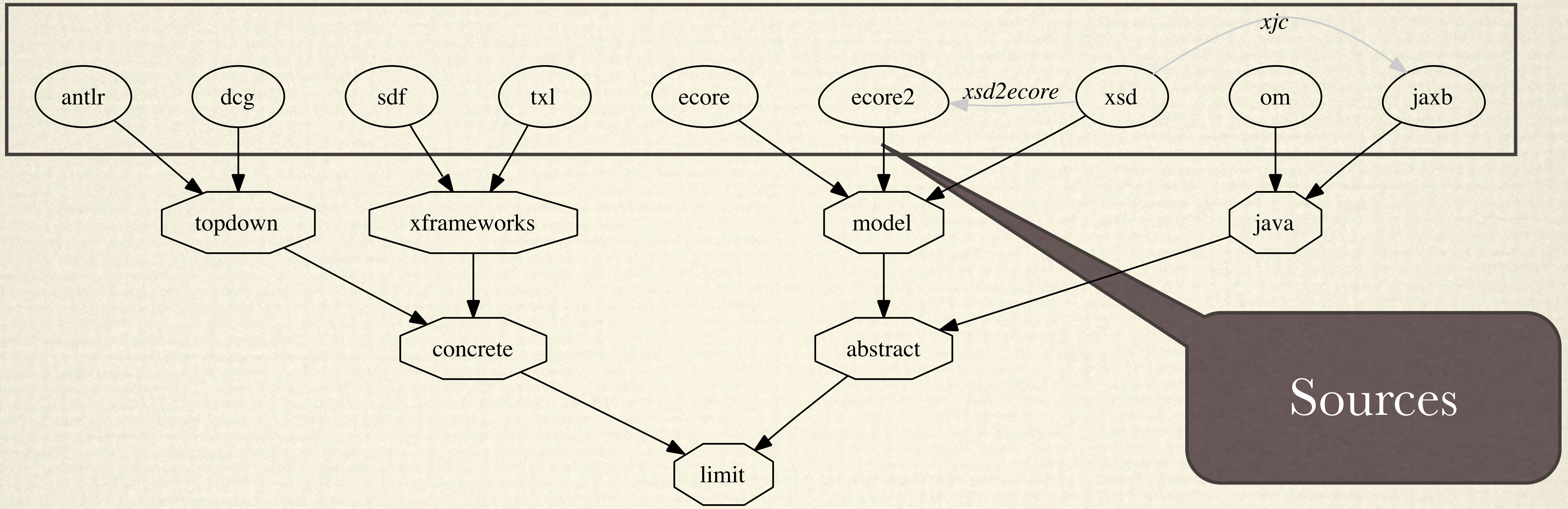
Technical side

- Software Language Processing Suite (SourceForge)
- Prolog
- Python
- Shell scripts
- XML Schema
- …

Core convergence tools

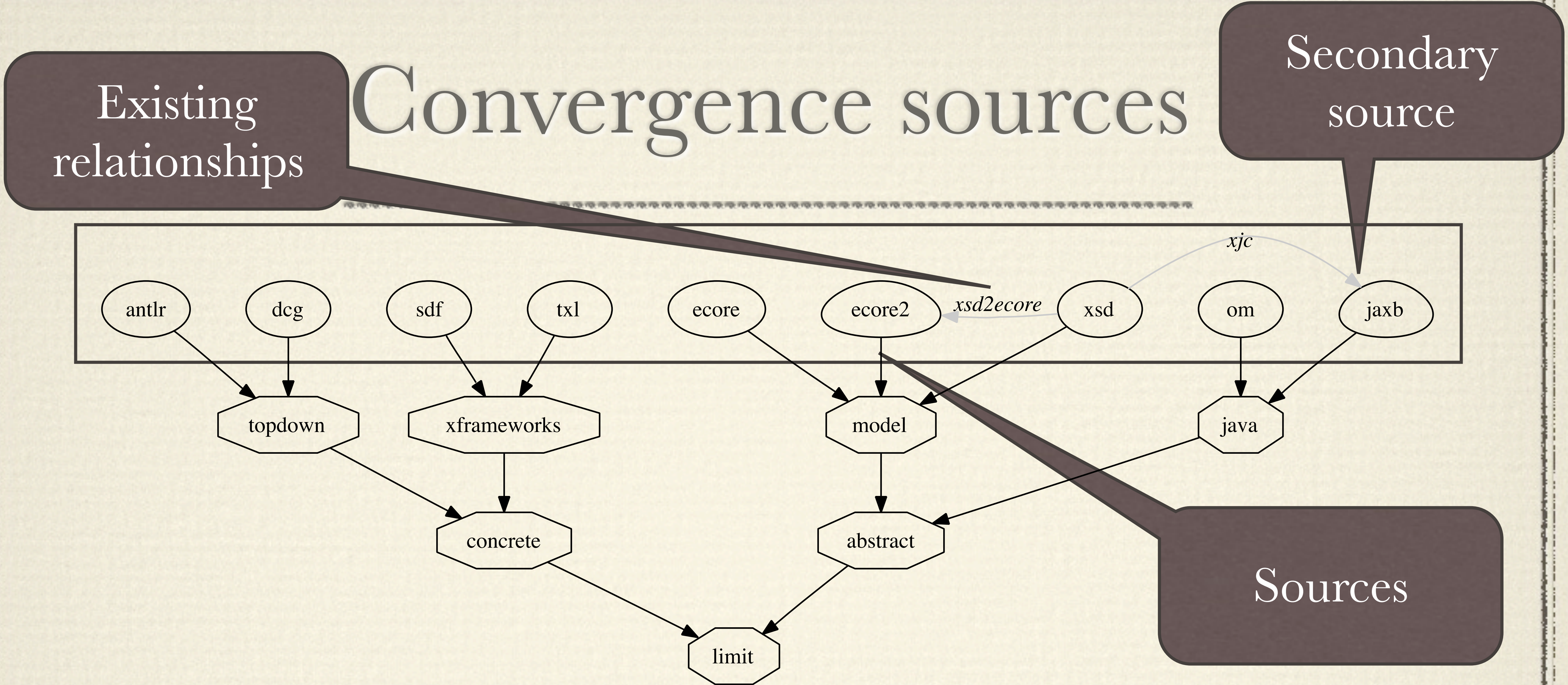
- Comparison
  - gdt left.bgf right.bgf
- Transformation
  - xbgf script.xbgf input.bgf output.bgf
- Validation
  - xmllint --noout --schema bgf.xsd input.bgf

Convergence sources



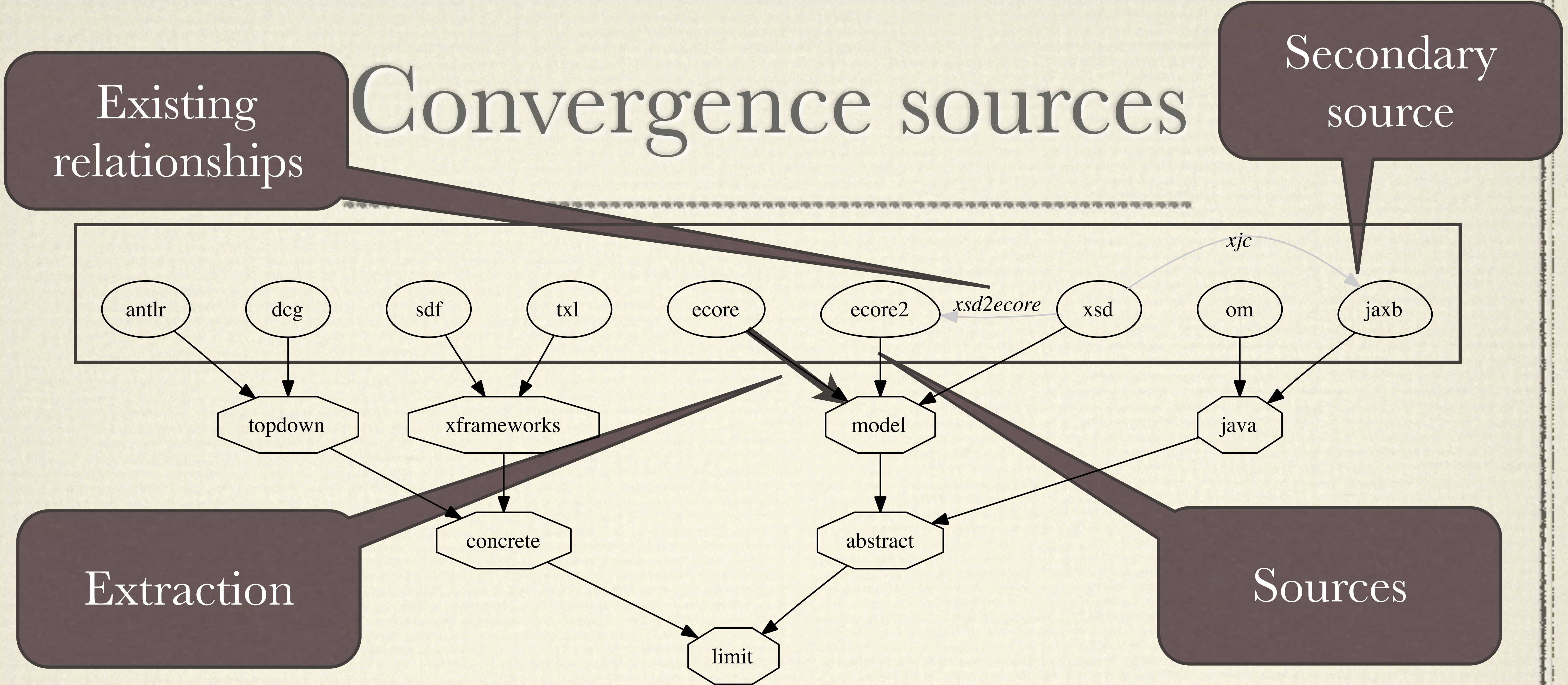
- Sources of grammar knowledge
-

Convergence sources



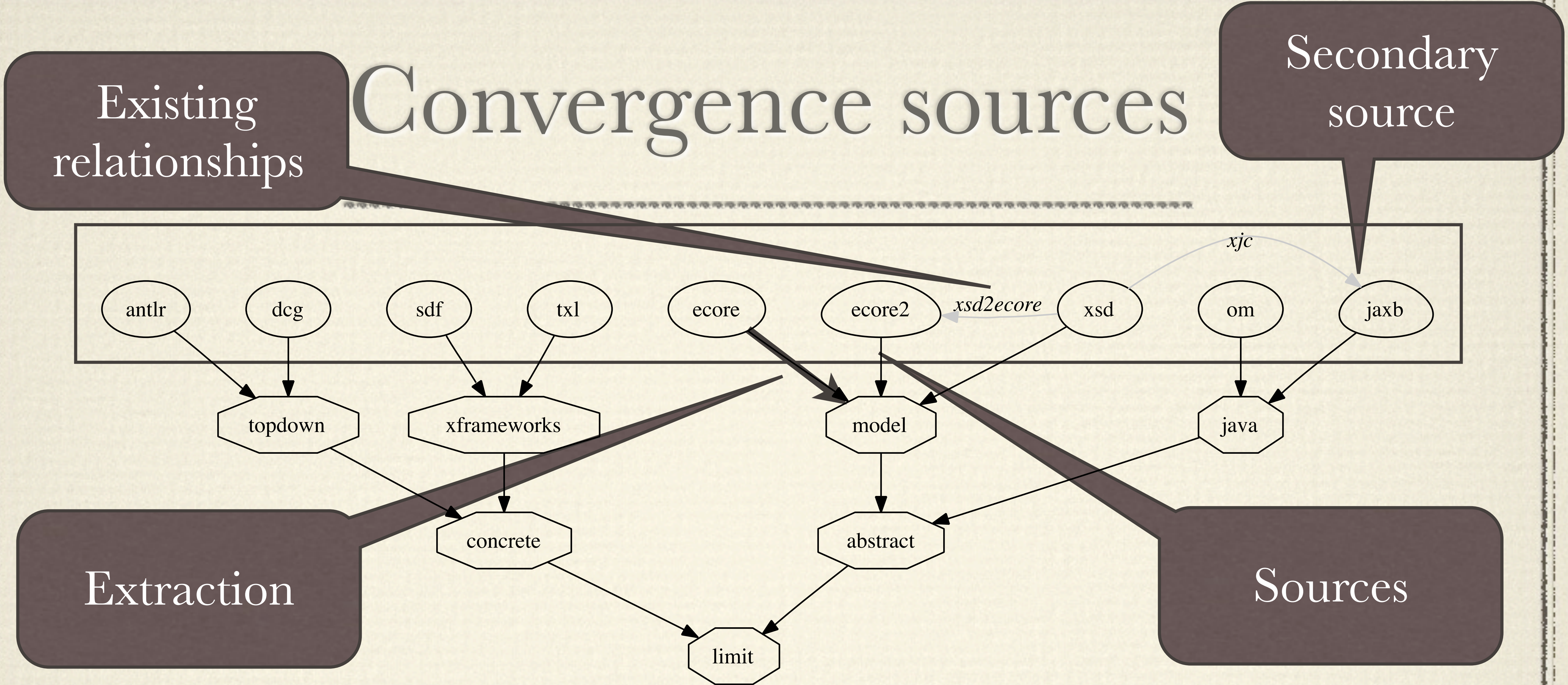
- Sources of grammar knowledge
- Heterogeneous artifacts
-

Convergence sources



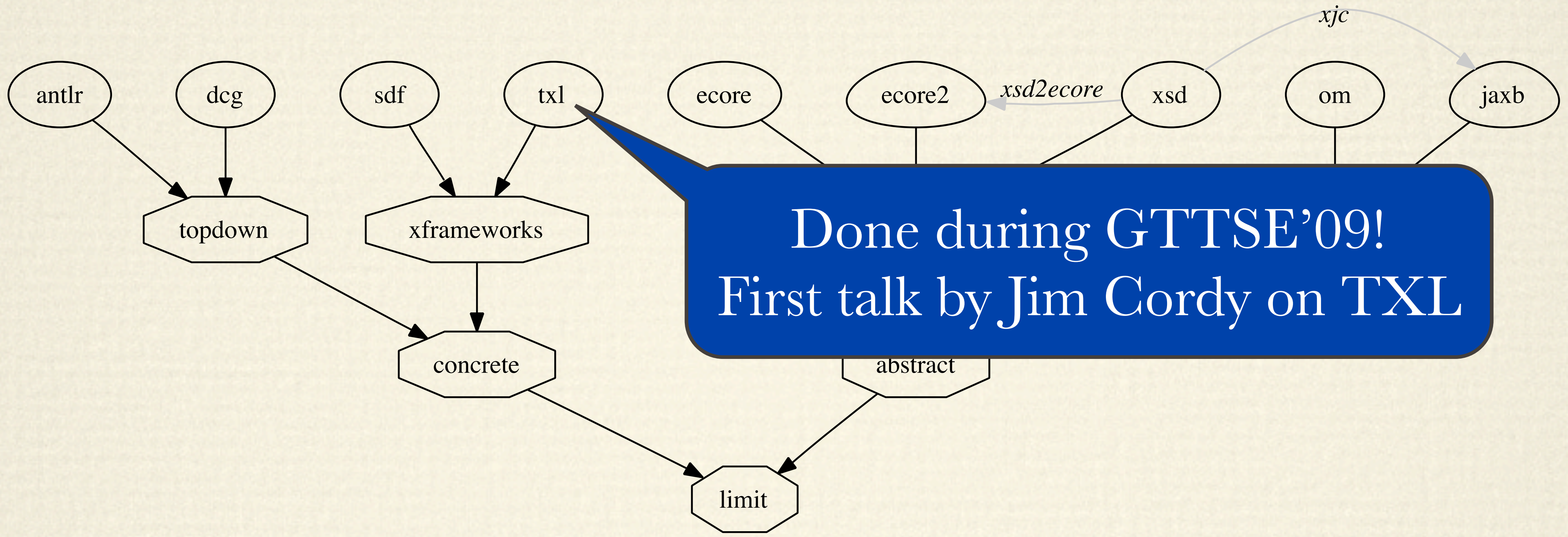
- Sources of grammar knowledge
- Heterogeneous artifacts
- Grammar properties: extraction, parsing, evaluation
-

Convergence sources



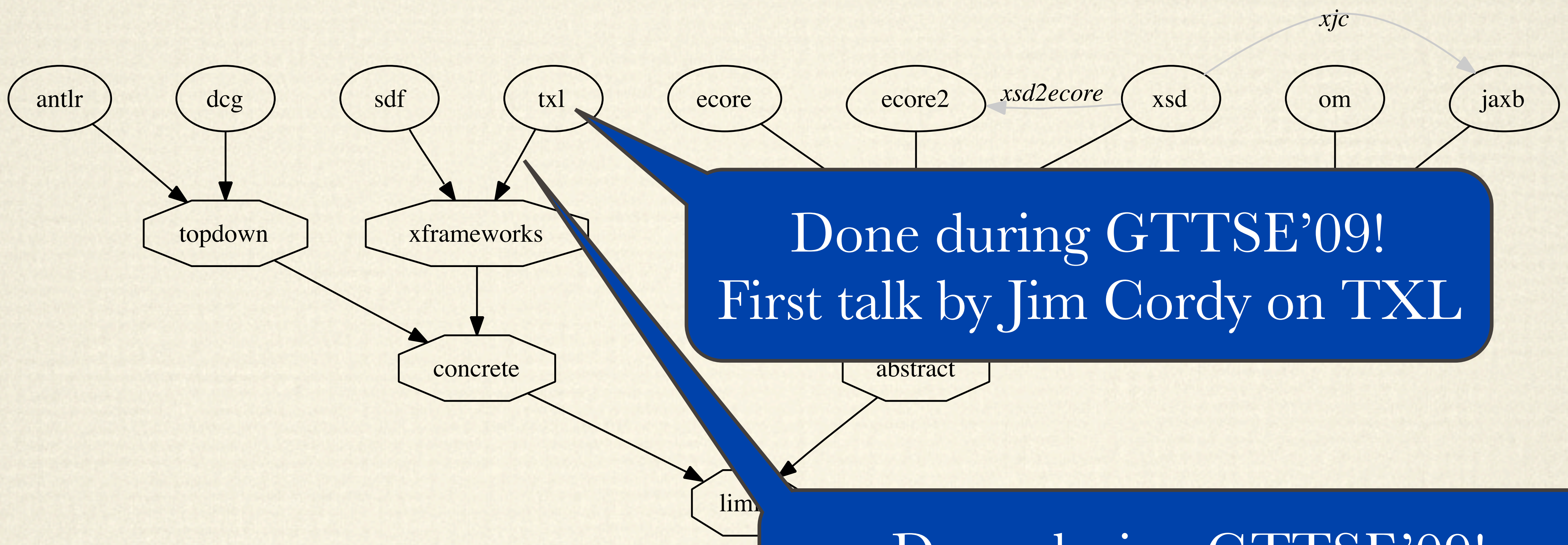
- Sources of grammar knowledge
- Heterogeneous artifacts
- Grammar properties: extraction, parsing, evaluation
- Instance properties; testing properties

Convergence sources



Done during GTTSE'09!
First talk by Jim Cordy on TXL

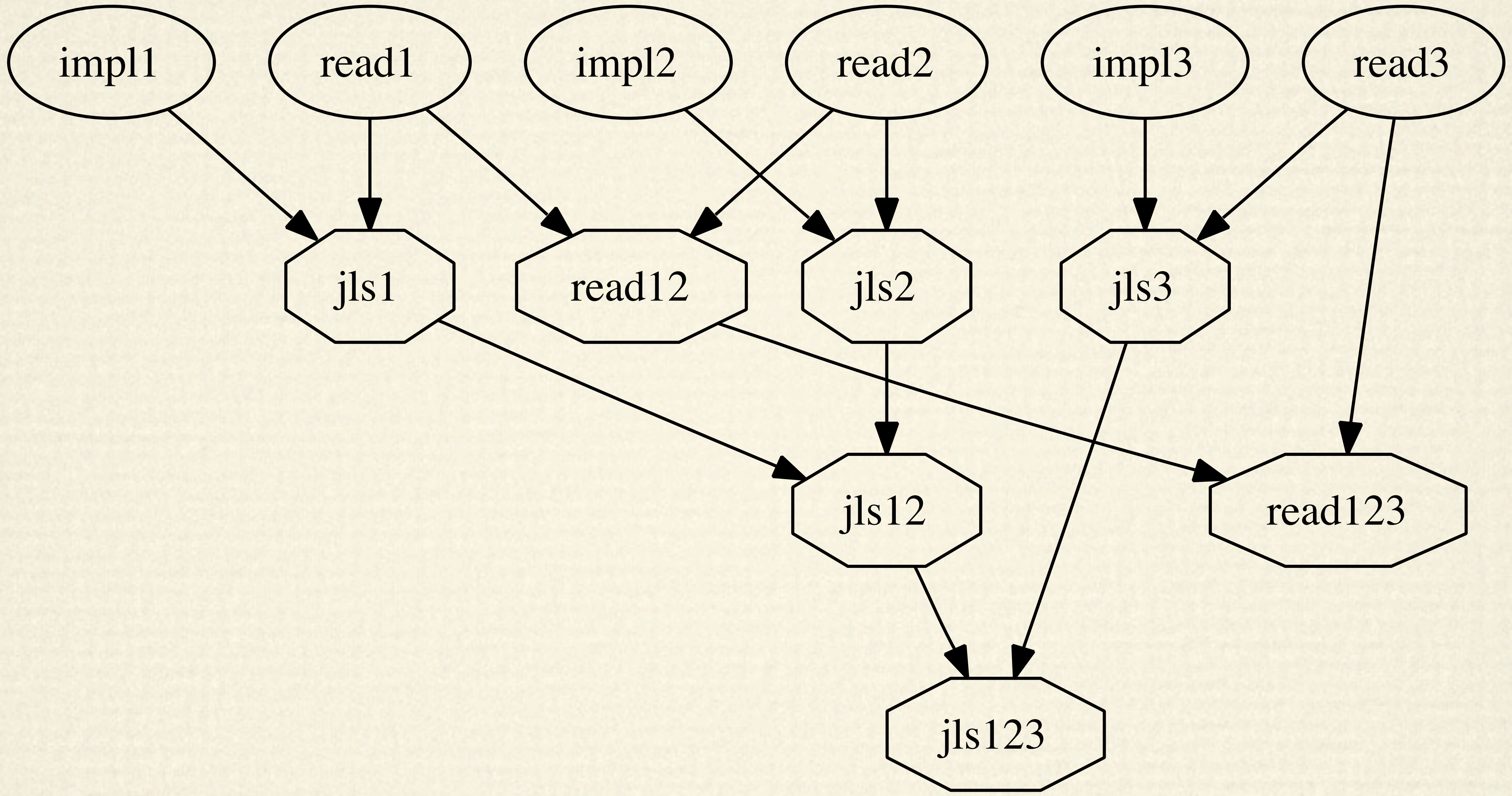
Convergence sources



Done during GTTSE'09!
First talk by Jim Cordy on TXL

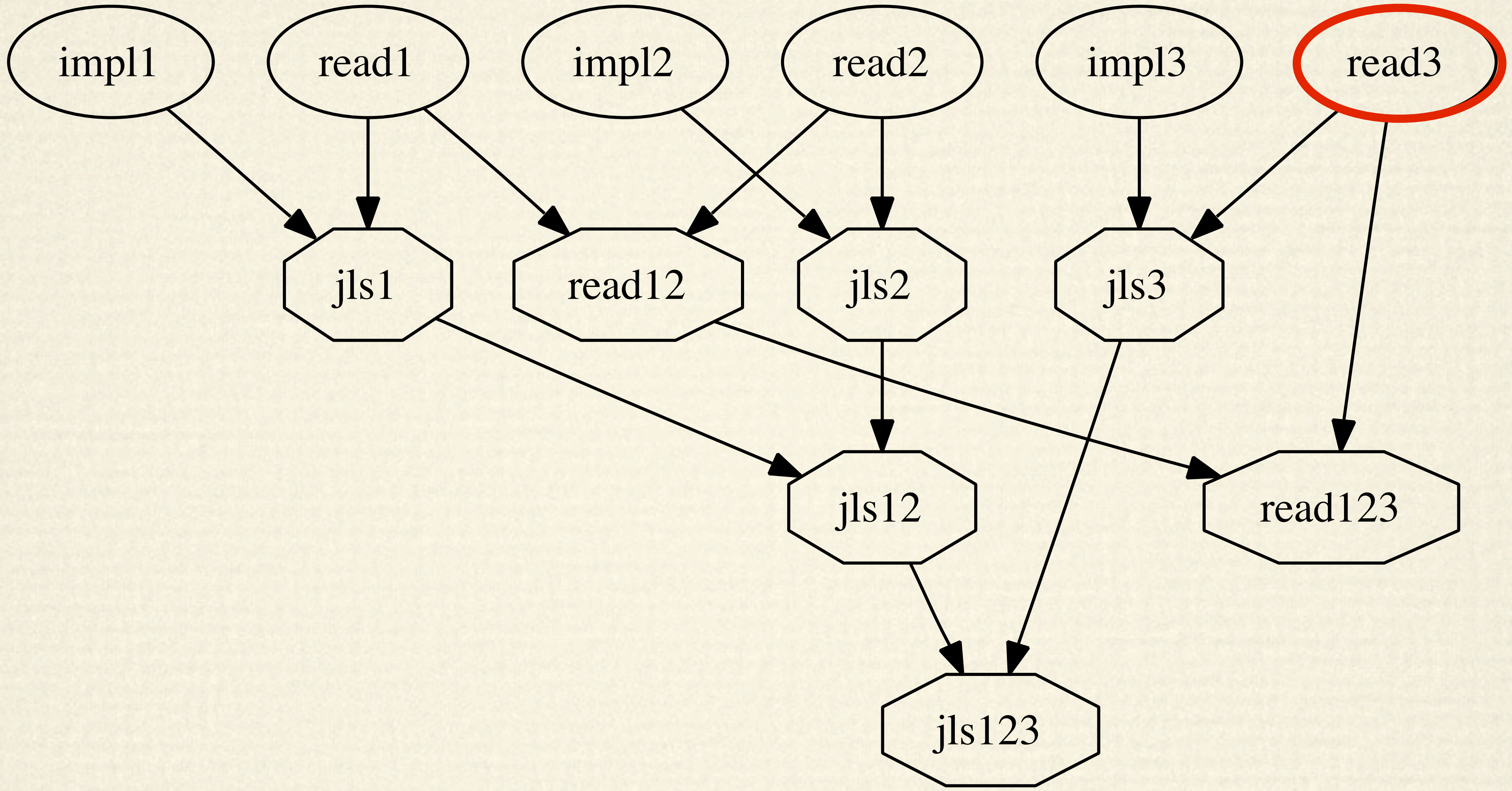
Done during GTTSE'09!
Second talk by Jim Cordy on TXL

Convergence sources (2)



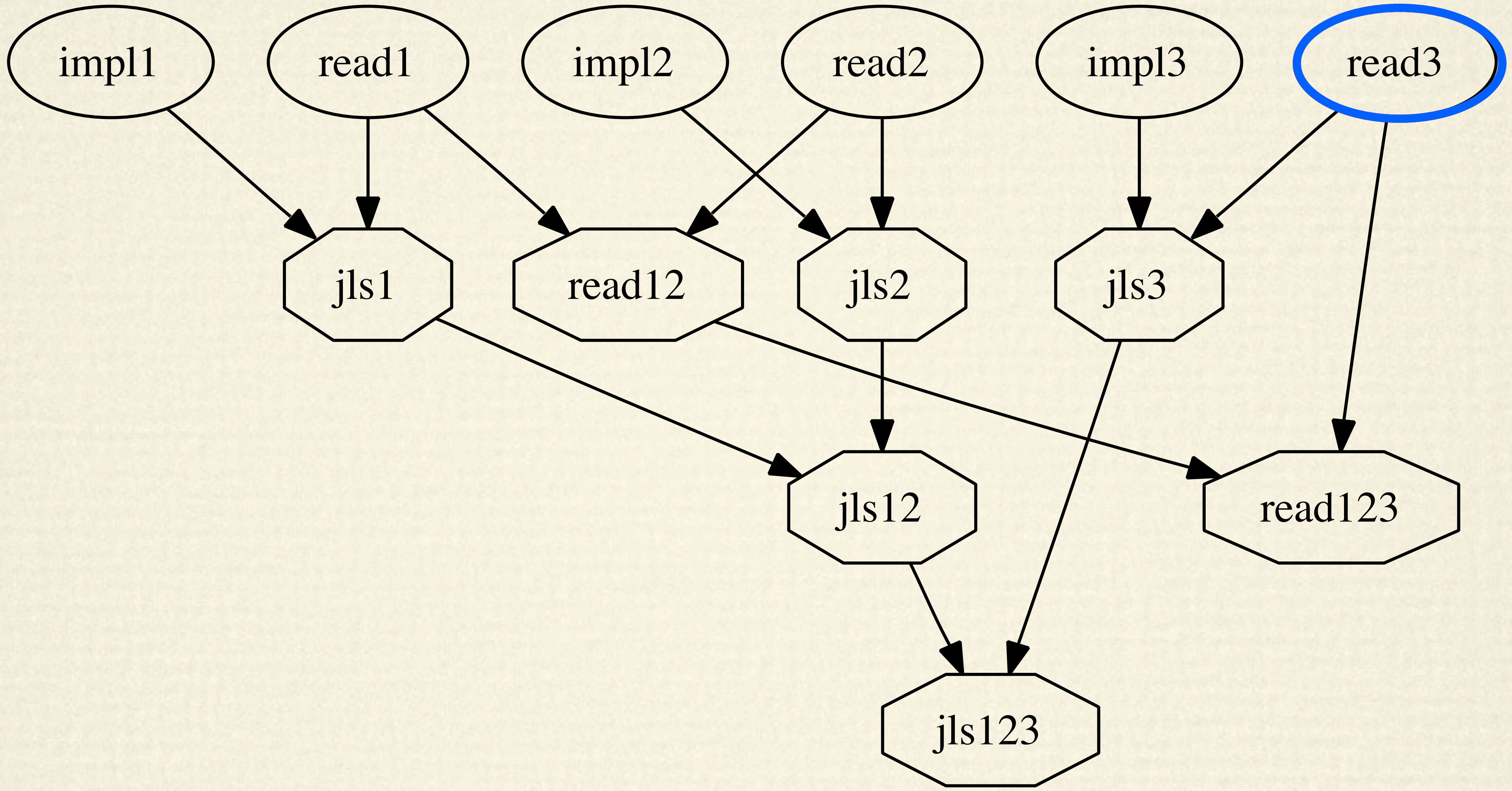
- Extraction happens once per source
-

Convergence sources (2)



- Extraction happens once per source
- Failed extraction are marked on the diagram
-

Convergence sources (2)



- Extraction happens once per source
- Failed extraction are marked on the diagram
- Snapshots and fallback

Grammar definition formalism

- BGF: BNF-like Grammar Format
- BNF: symbols, composition
- EBNF: *, +, ?
- Production labels and expression selectors
- …
- XML

A word on extractors

- Source format → unified format
- Abstraction from idiosyncrasies
- Can be intricate
- Specific for the source type, not the source

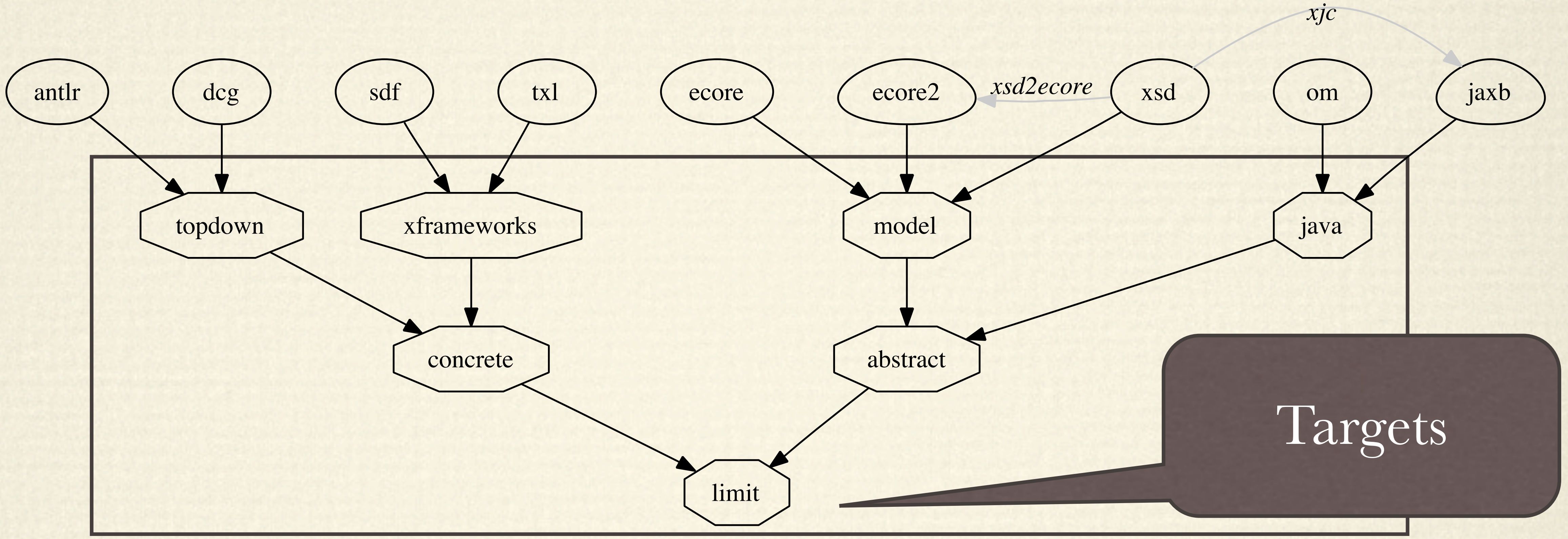
Nontrivial extraction in numbers

	app1	**app2**	**app3**	**doc1**	**doc2**	**doc3**	**Total**
Arbitrary lexical decisions	2	109	60	1	90	161	423
Well-formedness violations	5	0	7	4	11	4	31
Indentation violations	1	2	7	1	4	8	23
Recovery rules	3	12	18	2	59	47	141
◦ Match parentheses	0	3	6	0	0	0	9
◦ Metasymbol to terminal	0	1	7	0	27	7	42
◦ Merge adjacent symbols	1	0	0	1	1	0	3
◦ Split compound symbol	0	1	1	0	3	8	13
◦ Nonterminal to terminal	0	7	3	0	8	11	29
◦ Terminal to nonterminal	1	0	1	1	17	13	33
◦ Recover optionality	1	0	0	0	3	8	12
Purge duplicate definitions	0	0	0	16	17	18	51
Total	11	123	92	24	181	238	669

Available extractors

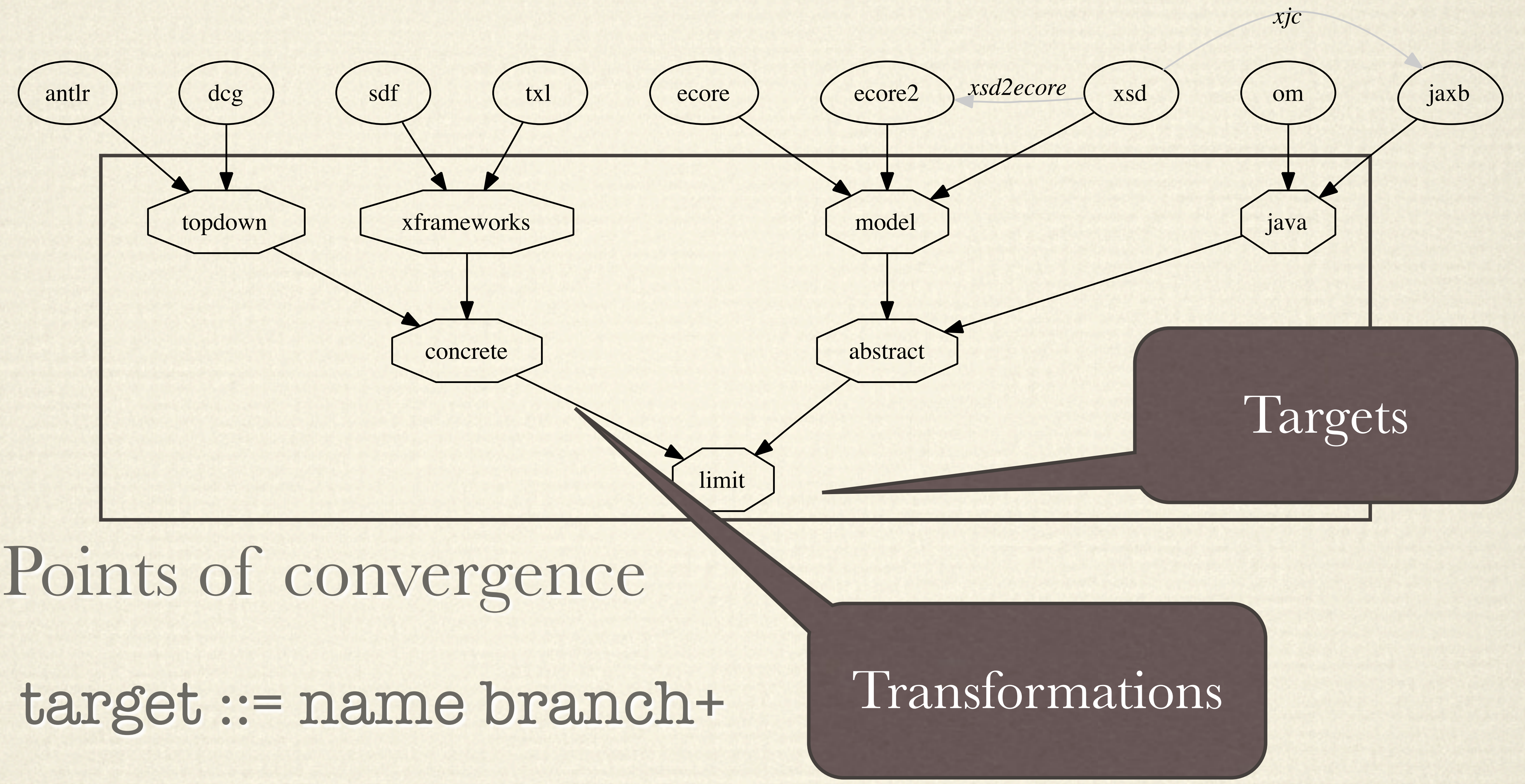
- ✓ ANTLR parser definitions
  - ➡ ANTLR self-application
- ✓ Syntax Defnition Formalism
  - ➡ ASF+SDF MetaEnvironment or Stratego/XT
- ✓ Definite clause grammars in Prolog
  - ➡ Prolog
- ✓ Java object models
  - ➡ reflection with `java.lang.reflect` or `com.sun.source.tree`
- ✓ ECore models in XMI
  - ➡ XSLT
- ✓ XML Schema schemata
- ✓ Language specifications

Convergence targets



- Points of convergence
- target ::= name branch+
- branch ::= input phase*
- Use comparison tool at the end

Convergence targets



- Points of convergence
- target ::= name branch+
- branch ::= input phase*
- Use comparison tool at the end

Grammar trasformation

- Initial corrections
- Nominal matching
- Structured matching by refactoring
- Relaxation/restriction
- Extension
- Correction

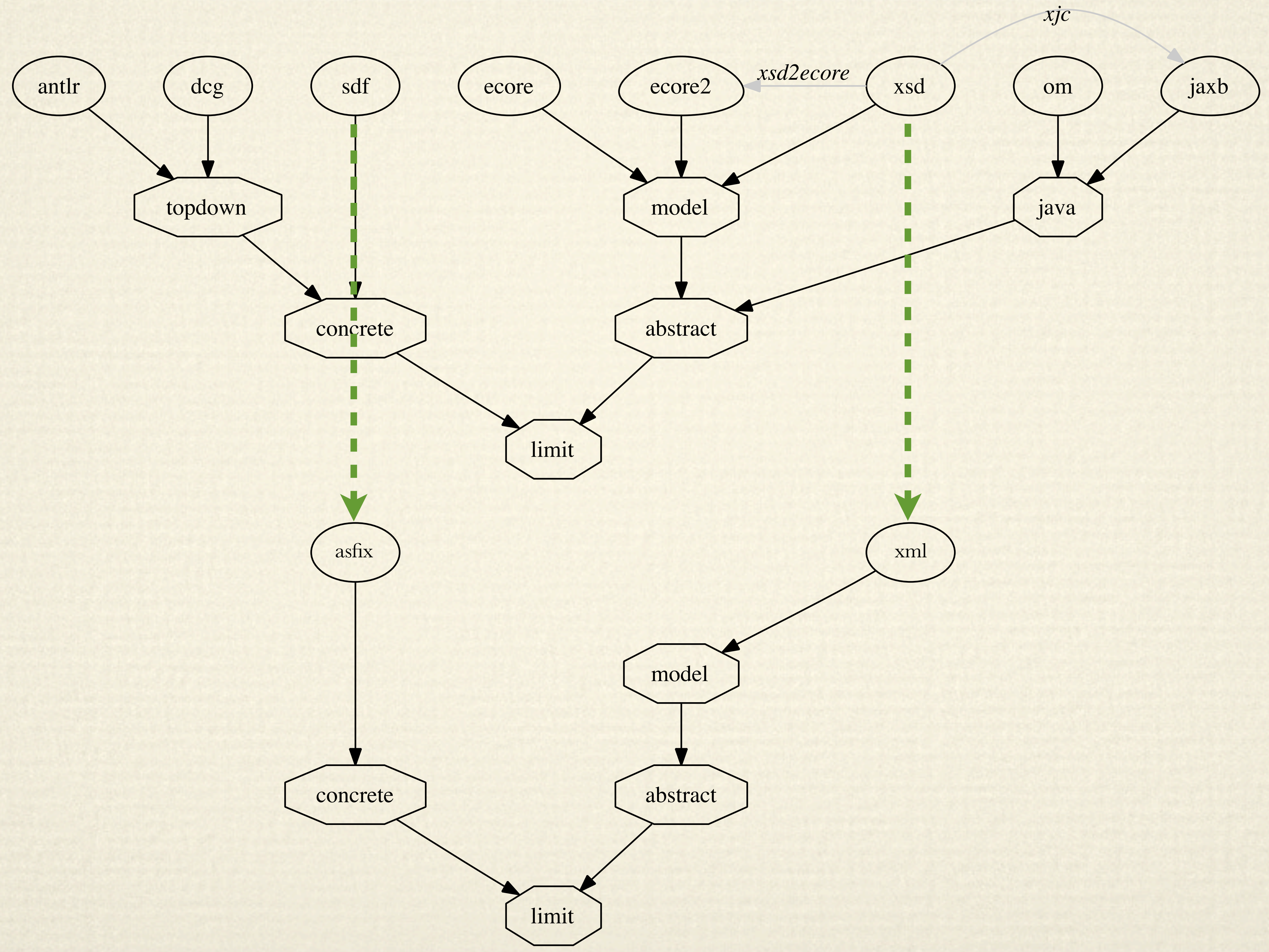
Grammar transformation (2)

- Static XBGF script
  - can be reused
- Transformation generator
  - strip away all terminal symbols
  - eliminate unused nonterminal symbols
  - apply naming convention

Transformation statistics for JLS

	jls1	**jls12**	**jls123**	**jls2**	**jls3**	**read12**	**read123**	**Total**
Number of lines	682	5116	2847	6772	10715	1639	3082	30853
Number of transformations	67	298	111	395	544	77	135	1627
◦ Semantics-preserving	45	239	80	283	381	31	78	1137
◦ Semantics-increasing or -decreasing	22	58	31	102	150	39	53	455
◦ Semantics-revising	—	1	—	10	13	7	4	35
Preparation phase	1	—	—	15	24	11	14	65
◦ Known bugs (Ex. 3.7)	—	—	—	1	11	—	4	16
◦ Post-extraction (Ex. 3.8)	—	—	—	7	8	7	5	27
◦ Initial correction (Ex. 3.9)	1	—	—	7	5	4	5	22
Resolution phase	21	59	31	97	139	35	43	425
◦ Extension (Ex. 3.4)	—	17	26	—	—	31	38	112
◦ Relaxation (Ex. 3.5)	18	39	5	75	112	—	2	251
◦ Correction (Ex. 3.6)	3	3	—	22	27	4	3	62

Coupled transformations



Language documentation

- Language document is…
  - a (sliced) (formal) grammar
  - textual annotations for human understanding
  - source code samples
- Language evolution vs
  Language documentation evolution

Language documentation

- Language document is…
  - a (sliced) (formal) grammar
  - textual annotations for human understanding
  - source code samples
- Language evolution vs
  Language documentation evolution

Extract

Language documentation

- Language document is…
    - a (sliced) (formal) grammar
    - textual annotations for human understanding
    - source code samples
- Language evolution vs
  Language documentation evolution

Extract

Test

Language documentation

Converge!

Extract

- Language document is…
  - a (sliced) (formal) grammar
  - textual annotations for human understanding
  - source code samples
- Language evolution vs
  Language documentation evolution

Test

The end.

- More questions?
- Suggestions?
- Related work advice?

Discussion topics

- Transformation scripts reengineering and maintenance (XXBGF?)
- Transformation generators — what input?
- Extended comparison results (advice)
- Defining metrics and benchmarking
- Extending the infrastructure for documentation
- Formal algebraic proof for operator semantics