Reverse Engineering Grammar Relationships

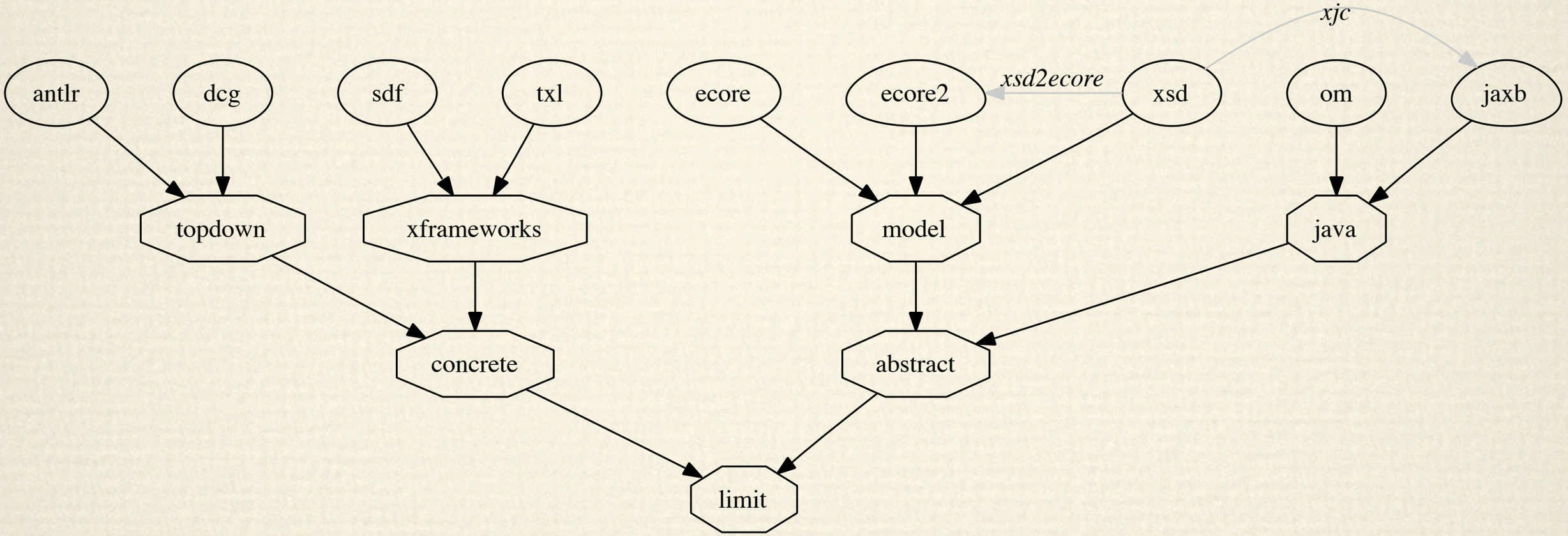
Ralf Lämmel and Vadim Zaytsev
Software Languages Team
Universität Koblenz-Landau

Grammar consistency checking

- ★ Co-existing grammars embedded in software artifacts
  - ✦ e.g. parser spec, data model, language standard
- ★ Question: do grammars describe the same language?
- ★ Goal: reliably establish & continuously maintain that
- ★ Not always BNF, and even then…
- ★ Not always meant to be equal, and even then…

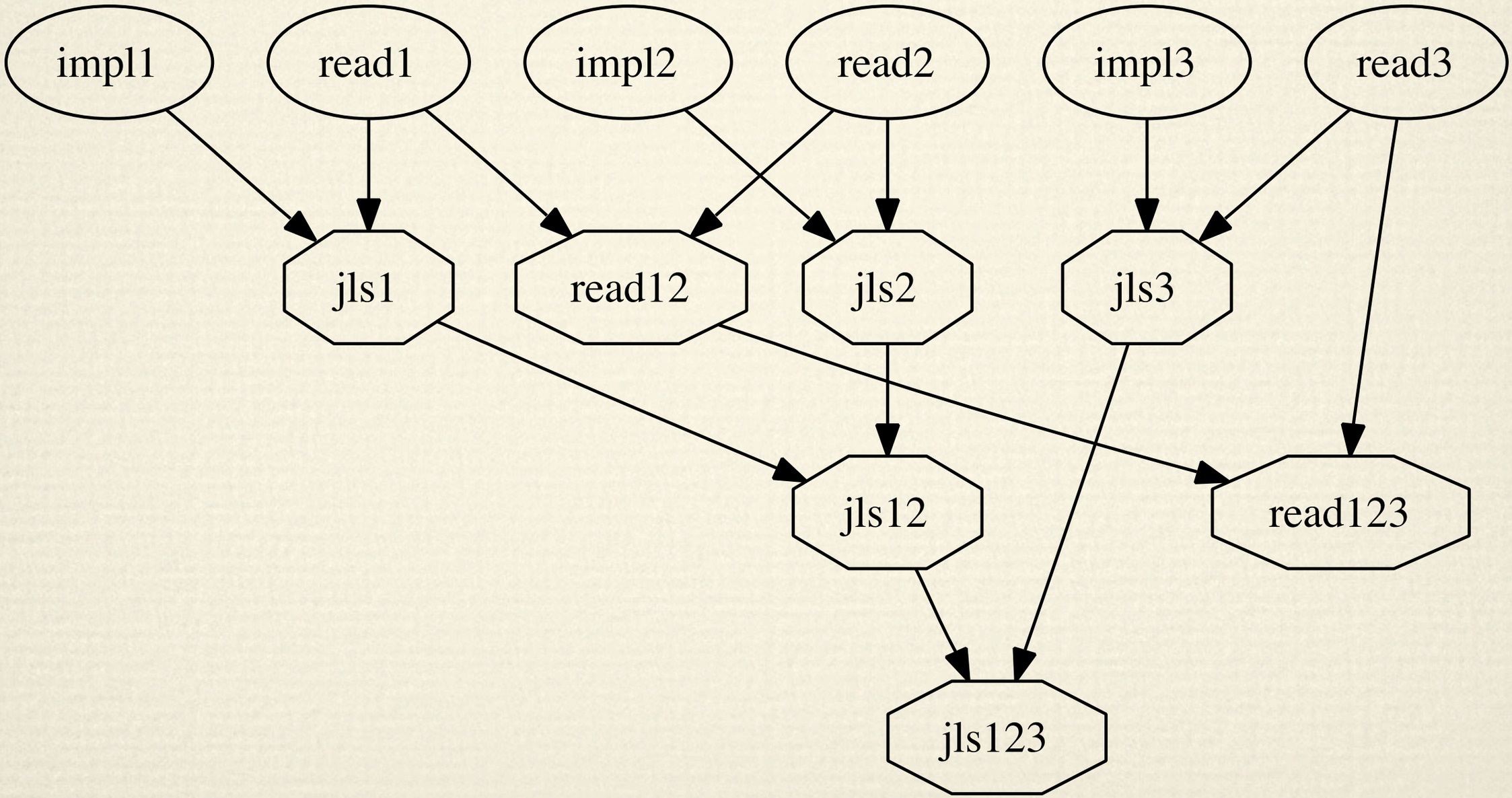
Example motivational scenario

Different implementations of the same language (parsers, data models, etc.)



Alternative scenario

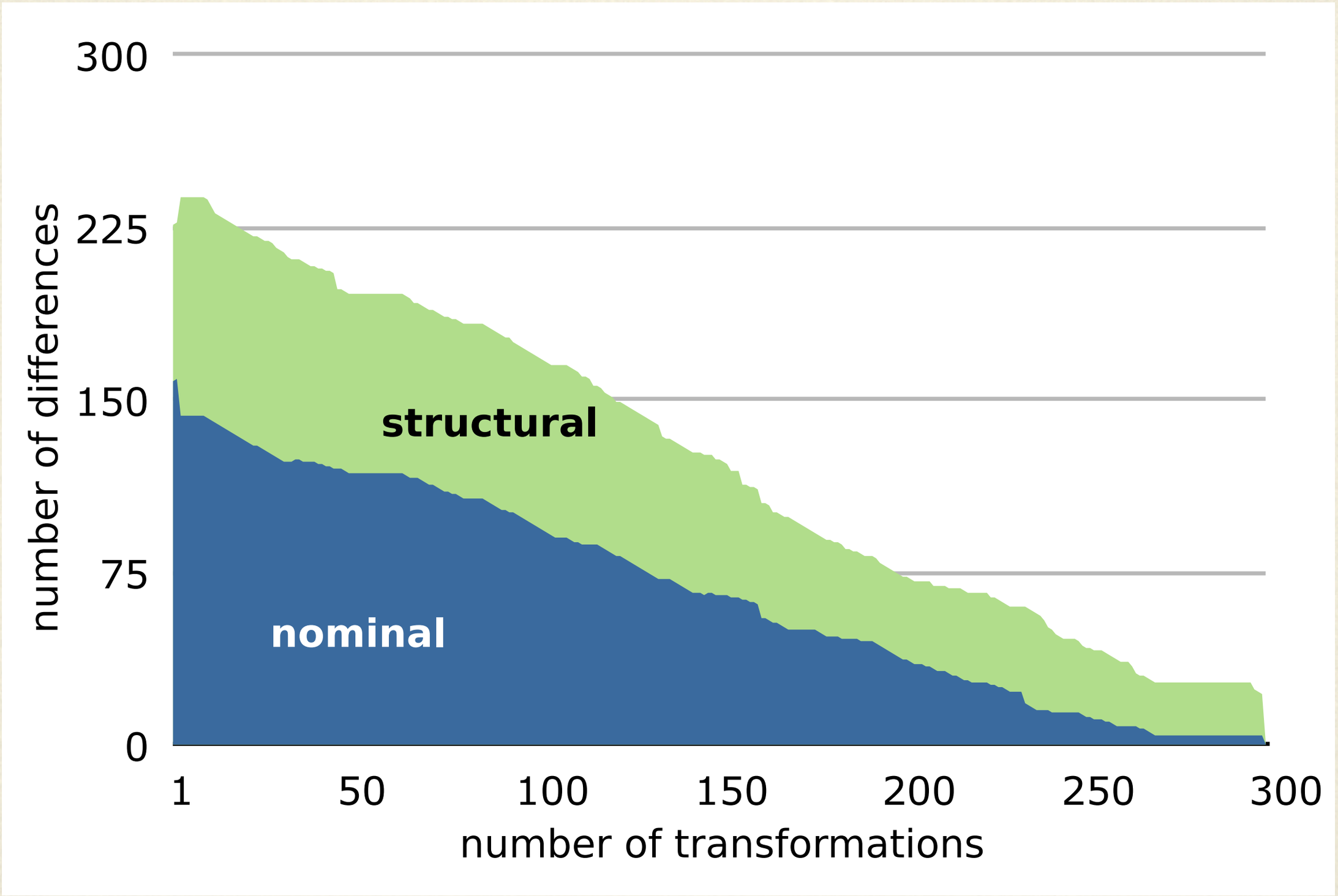
Different versions of a language as documented by specifications



Grammar differences

- ★ intended vs. accidental
- ★ result of grammar adaptation
- ★ result of grammar evolution
- ★ idiosyncrasies
- ★ presentation and understandability
- ★ misspelling
- ★ …etc
- ★ **nominal** & **structural**

Grammar measurement



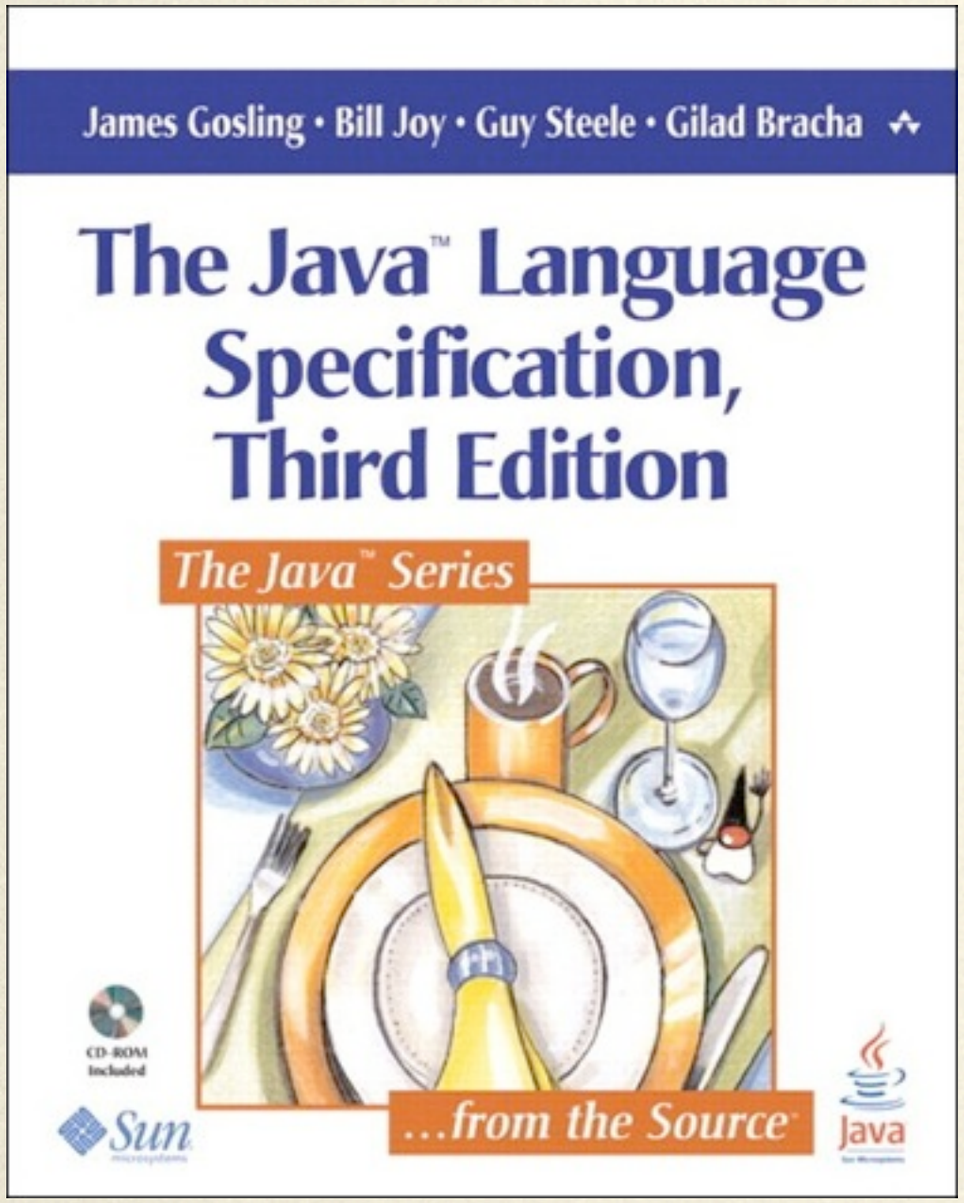
Language convergence method

- ★ Grammar *format* free from idiosyncrasies
- ★ Grammar *extraction* for notation mapping
- ★ Grammar *comparison* for spotting grammar differences
- ★ Grammar *transformation*:
  - ✦ Refactoring; extension / restriction; revision
- ★ Grammar *measurement*:
  - ✦ Nominal differences; structural differences

Language evolution

- ★ Just checking for equivalence is not enough
- ★ Languages evolve
- ★ IDEs/compilers/tools evolve
- ★ Documentation evolves
- ★ Evolution can be independent
- ★ Consistency control must account for this

Case study:
Java Language Specification



★ The official language definition

★ Keeps up with language evolution

★ Foundation for compilers, pretty-printers, IDEs, …

★ Freely accessible in three versions

JLS irregularities in extraction

	impl1	**impl2**	**impl3**	**read1**	**read2**	**read3**	**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

Transformations for JLS

	jls1	**jls12**	**jls123**	**jls2**	**jls3**	**read12**	**read123**	**Total**
Number of lines	682	5114	2847	6774	10721	1639	3082	30859
Number of transformations	67	290	111	387	544	77	135	1611
○ Semantics-preserving (§4.2.2)	45	231	80	275	381	31	78	1121
○ Semantics-increasing/-decreasing	22	58	31	102	150	39	53	455
○ Semantics-revising	—	1	—	10	13	7	4	35
Preparation phase (§4.2.1)	1	—	—	15	24	11	14	65
○ Known bugs	—	—	—	1	11	—	4	16
○ Post-extraction	—	—	—	7	8	7	5	27
○ Initial correction	1	—	—	7	5	4	5	22
Resolution phase	21	59	31	97	139	35	43	425
○ Extension (§4.2.3)	—	17	26	—	—	31	38	112
○ Relaxation (§4.2.4)	18	39	5	75	112	—	2	251
○ Correction (§4.2.5)	3	3	—	22	27	4	3	62

Where are grammar relationships?

	jls1	jls12	jls123	jls2	jls3	read12	read123	Total
○ *rename*	9	4	2	9	10	—	2	36
○ *reroot*	2	—	—	2	2	2	1	9
○ *unfold*	1	10	8	11	13	2	3	48
○ *fold*	4	11	4	11	13	2	5	50
○ *inline*	3	67	8	71	100	—	1	250
○ *extract*	—	17	5	18	30	—	5	75
○ *chain*	1	—	2	—	—	1	4	8
○ *massage*	2	13	—	15	32	5	3	70
○ *distribute*	3	4	2	3	6	—	—	18
○ *factor*	1	7	3	5	24	3	1	44
○ *deyaccify*	2	20	—	25	33	4	3	87
○ *yaccify*	—	—	—	—	1	—	1	2
○ *eliminate*	1	8	1	14	22	—	—	46
○ *introduce*	—	1	30	4	13	3	34	85
○ *import*	—	—	2	—	—	—	1	3
○ *vertical*	5	7	7	8	22	5	8	62
○ *horizontal*	4	19	5	17	31	4	4	84
○ *add*	1	14	13	7	20	28	20	103
○ *appear*	—	8	11	8	25	2	17	71
○ *widen*	1	3	—	1	8	1	3	17
○ *upgrade*	—	8	—	14	20	2	2	46
○ *unite*	18	2	—	18	21	5	4	68
○ *remove*	—	10	1	11	18	—	1	41
○ *disappear*	—	7	4	11	11	—	—	33
○ *narrow*	—	—	1	—	4	—	—	5
○ *downgrade*	—	2	—	8	3	—	—	13
○ *define*	—	6	—	4	9	1	6	26
○ *undefine*	—	3	—	5	3	—	—	11
○ *redefine*	—	3	—	8	7	6	2	26
○ *inject*	—	—	—	2	4	—	1	7
○ *project*	—	1	—	1	2	—	—	4
○ *replace*	3	1	2	3	6	1	1	17
○ *unlabel*	—	—	—	—	—	—	2	2

Further reading

- ★ Ralf Lämmel and Vadim Zaytsev,
  An Introduction to Grammar Convergence, iFM 2009
- ★ Ralf Lämmel and Vadim Zaytsev,
  Recovering Grammar Relationships for the JLS, SCAM 2009
- ★ Vadim Zaytsev,
  Language Convergence Infrastructure, GTTSE 2009
- ★ Ralf Lämmel and Vadim Zaytsev,
  Recovering Grammar Relationships for the JLS, SQJ
- ★ Software Language Processing Suite:
  http://slps.sf.net/

http://uni-koblenz.de/~laemmel/convergence & http://uni-koblenz.de/~laemmel/jls

Conclusion / Discussion

★ Reverse engineering grammar relationships

★ Straightforward analysis not possible

★ Straightforward reverse engineering not possible

★ We perform a transformation…

★ …and reverse engineer our actions.

★ Language convergence is the name

Questions?

Thank you!