BX
&
Grammarware

Software Engineering Meeting
Vadim Zaytsev, SWAT, CWI
2012

Background

- ★ BX 2012: First International Workshop on Bidirectional Transformations (satellite event of ETAPS 2012)
- ★ "Language Evolution, Metasyntactically"
- ★ Submitted to BX 2012: 21 December 2011
- ★ Notification: 24 January 2012
- ★ Camera-ready copy: 5 February 2012
- ★ Conditionally accepted with 2×SA and 2×WR

Bidirectional transformations community

- A cross-discipline field:
    - Model-Driven Software Development: sync views
    - Graphical User Interfaces: model-view-controller
    - Visualization with Direct Manipulation: animation
    - Relational Databases: updatable views
    - Data Transformation, Integration and Exchange: map data and merge it
    - Data Synchronizers: bridge the gap between replicas in different formats
    - Macro Systems: give feedback in terms of original program elements
    - Domain-Specific Languages: runtime mapping in embedded interpreters
    - Structure Editors: interfaces for editing complicated data sources
    - Serializers: map external data to structured objects

Czarnecki, Foster, Hu, Lämmel, Schürr, Terwilliger, GRACE

Introduction

Introduction

- Every language document employs its own notation
- We focus on **metalanguage evolution**
    - the language itself does not evolve
    - the notation in which it is written, does
- We limit ourselves to grammarware technical space
- Working prototypes are a part of SLPS

Motivating example

LLL in itself [LDTA'02]

```
grammar      : rule+;
rule         : sort ":" alts ";";
alts         : alt alts-tail*;
alts-tail    : "|" alt;
alt          : term*;
term         : basis repetition?;
basis        : literal | sort;
repetition : "*" | "+" | "?";
```

LLL in itself [GDK]

```
specification : rule+;
rule          : ident ":" disjunction ";";
disjunction   : {conjunction "|"}+;
conjunction   : term*;
term          : basis repetition?;
basis         : ident | literal
              | alternation | group;
repetition    :"+"|"*"|"?";
alternation   : "{"basis basis "}" repetition;
group         : "(" disjunction ")" ;
```

LLL1 in EDD [Z12a]

defining metasymbol	:	definition separator metasymbol	|
terminator metasymbol	;	postfix optionality metasymbol	?
postfix star metasymbol	*	postfix plus metasymbol	+
start terminal metasymbol	"	end terminal metasymbol	"

Δ between LLL1 and LLL2

start group metasymbol	(	end group metasymbol	)
start separator list star metasymbol	{	end separator list star metasymbol	}*
start separator list plus metasymbol	{	end separator list plus metasymbol	}+

Metasyntactic evolution megamodel

Grammar internal representation

grammar(Rs,Ps) $\Leftarrow$ *mapoptlist(n,Rs), maplist(prod,Ps).*	grammar = start symbols + productions
prod(p(L,N,X)) $\Leftarrow$ *mapopt(label,L), atom(N), expr(X).*	production = label + lhs + rhs
label(l(X)) $\Leftarrow$ *atom(X).*	production labels
expr(true).	ε
expr(fail).	empty language
expr(a).	universal type
expr(t(T)) $\Leftarrow$ *atom(T).*	terminal symbols
expr(n(N)) $\Leftarrow$ *atom(N).*	nonterminal symbols
expr(’,’(Xs)) $\Leftarrow$ *maplist(expr,Xs).*	sequential composition
expr(’;’(Xs)) $\Leftarrow$ *maplist(expr,Xs).*	choice
expr(’?’(X)) $\Leftarrow$ *expr(X).*	optionality
expr(’∗’(X)) $\Leftarrow$ *expr(X).*	Kleene star
expr(’+’(X)) $\Leftarrow$ *expr(X).*	transitive closure
expr(slp(X,Y)) $\Leftarrow$ *expr(X), expr(Y).*	Y-separated list with 1 or more elements
expr(sls(X,Y)) $\Leftarrow$ *expr(X), expr(Y).*	Y-separated list with 0 or more elements
expr(s(S,X)) $\Leftarrow$ *atom(S), expr(X).*	selectable expressions

Toward bidirectional grammar transformation

- XBGF ⇒ ΞBGF:
  - renameN, factor, etc: flip arguments
  - addV/removeV, narrow/widen: form pairs
  - extract/inline, unlabel/designate: asymmetry
  - distribute: removed from the language
  - unite, equate: tricky, superposition of others
- BX is a stable way to represent grammar relationship

Toward transformable notation specification

- EDD [Z12a]
  - confix metaconstructs
  - infix, prefix, postfix metasymbols
  - predefined sets (e.g., built-in nonterminals)
  - conventions (e.g., naming, whitespace reliability)
- XEDD:
  - rename-metasymbol(s, v1, v2)
  - introduce-metasymbol(s, v)
  - eliminate-metasymbol(s)

Toward in-notation grammar transformation

- ★ Concrete syntax transformations
- ★ Avoiding discussion on propagation of CTS elements
- ★ bgfreformat tool:
  - ★ extract the grammar from the given notation
  - ★ manipulate (transform) the internal representation
  - ★ pretty-print the grammar in the desired notation
- ★ Alternatively, parse with grammar for grammars

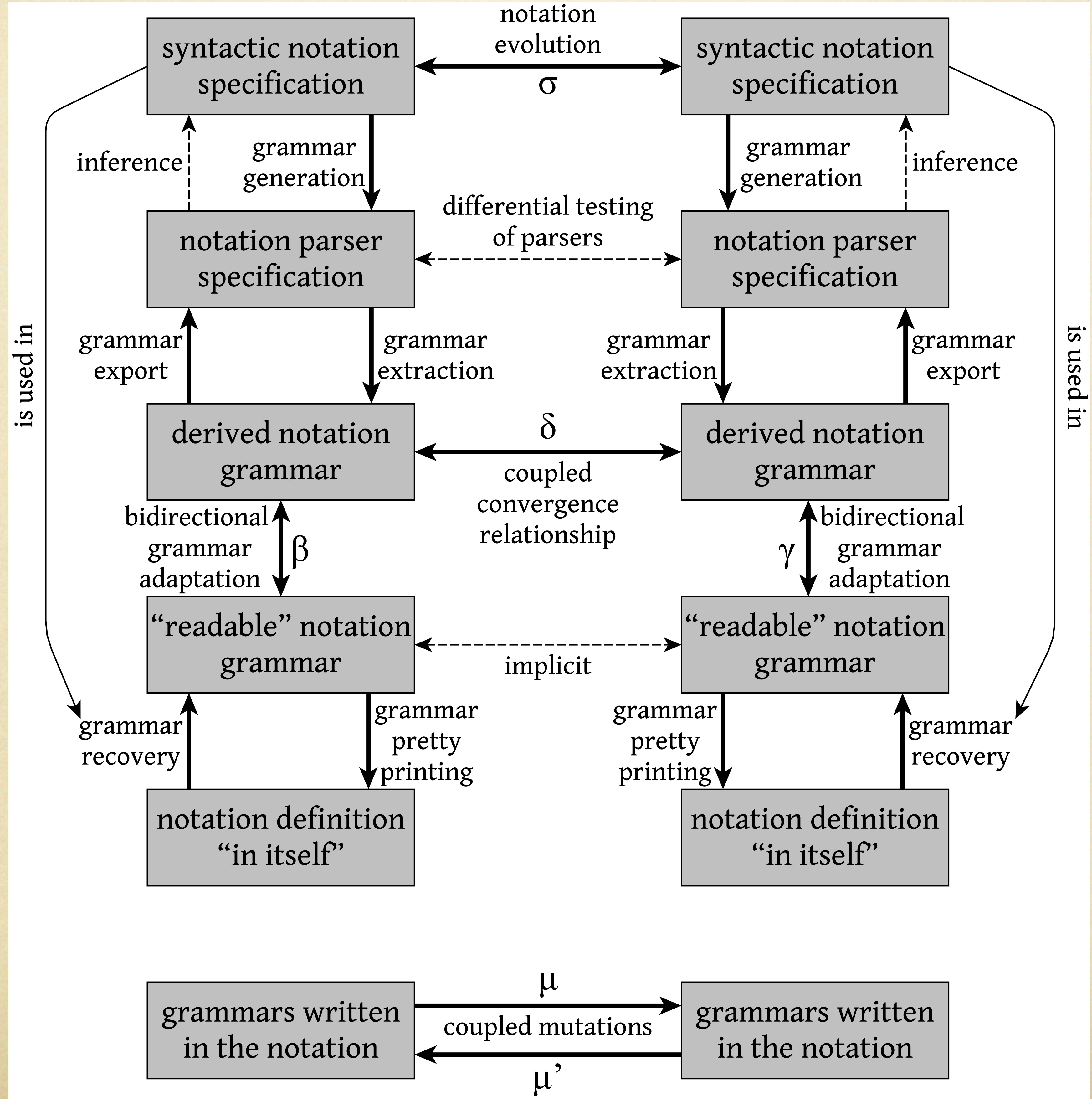
Grammar transformation vs. grammar mutation

- A **grammar transformation operator** τ can be formalised as a triplet: $\tau = \langle c\_pre, t, c\_post \rangle$.
- A **grammar transformation** then is τ_a_i (G), resulting in G'.
  - if a_i are of incorrect types and quantity than expected by t ⇒ τ is **incorrectly called**;
  - if the constraint c_pre does not hold on G ⇒ τ_a_i is **inapplicable** to G;
  - if the constraint c_post holds on G ⇒ τ_a_i is **vacuous** on G;
  - if the constraint c_pre holds on G and c_post does not hold on G' ⇒ t is **incorrectly implemented**;
  - if c_pre holds on G, c_post holds on G' ⇒ τ has been **applied correctly** with arguments a_i to G resulting in G'.

Grammar transformation vs. grammar mutation

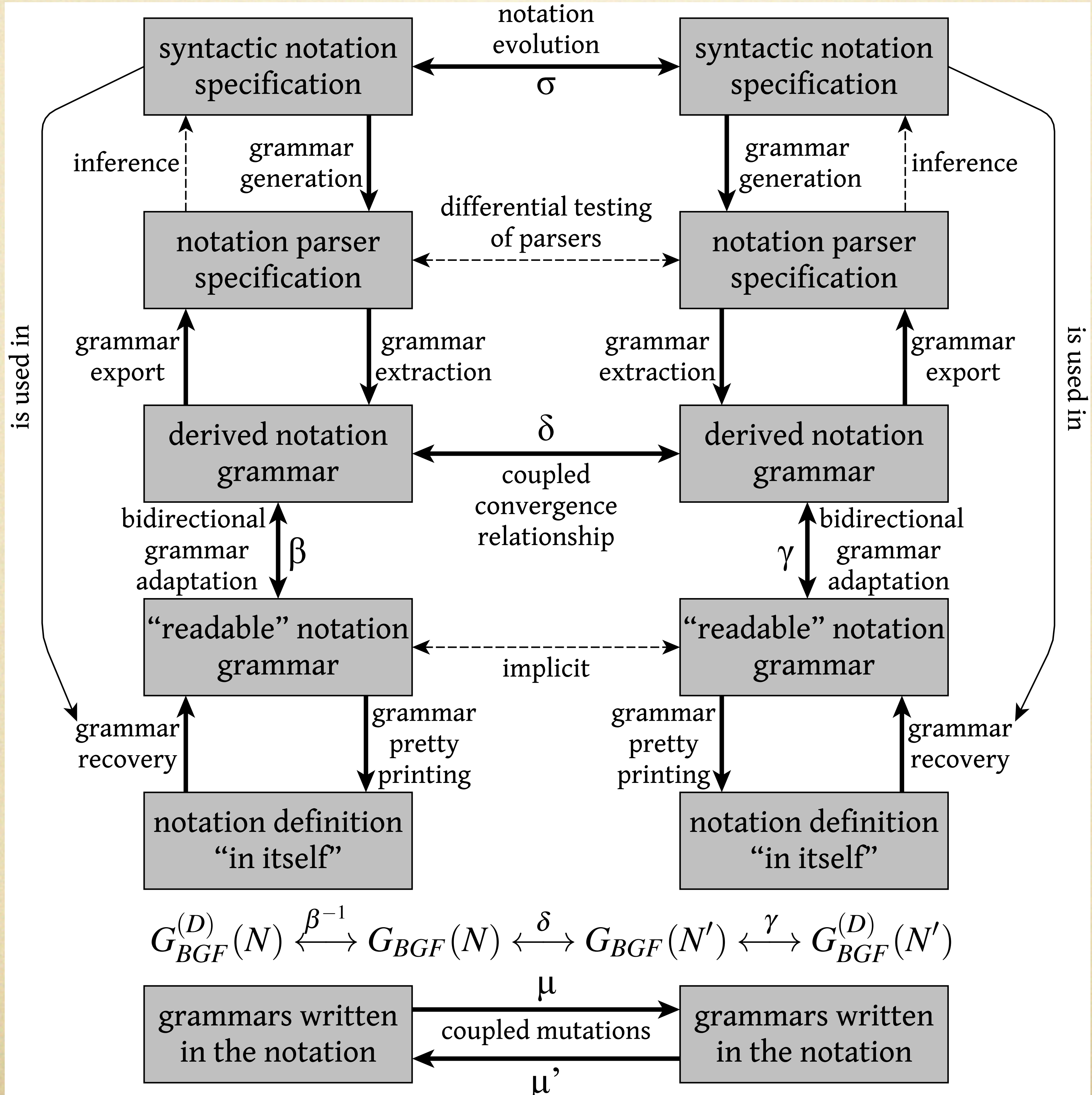
- ★ A **grammar mutation** does not have a single precondition
- ★ It has a set of preconditions that serve as **triggers**: $\mu = \langle \{c\_i\}, \{t\_i\}, c\_post \rangle$.
- ★ The mutation **terminates** once no trigger c_i holds and the postcondition c_post is met.
- ★ A **bidirectional** grammar mutation: $\mu\_bx = \langle c\_pre, \{c\_i\}, \{t\_i\}, c\_post \rangle$ will be an instantiation of a grammar mutation
- ★ The family of spawned BMs does **not** define the original: i.e., $\forall \mu\ \exists G\ \exists G'\ \nexists \mu\_bx,\ G' = \mu(G) \wedge G' = \mu\_bx(G) \wedge G = \mu^{-1}(G')$.

The megamodel



The megamodel

$$S(N) \longrightarrow G_{Rascal}(N) \longleftrightarrow G_{BGF}(N) \overset{\beta}{\longleftrightarrow} G^{(D)}_{BGF}(N) \longleftrightarrow G_N(N)$$



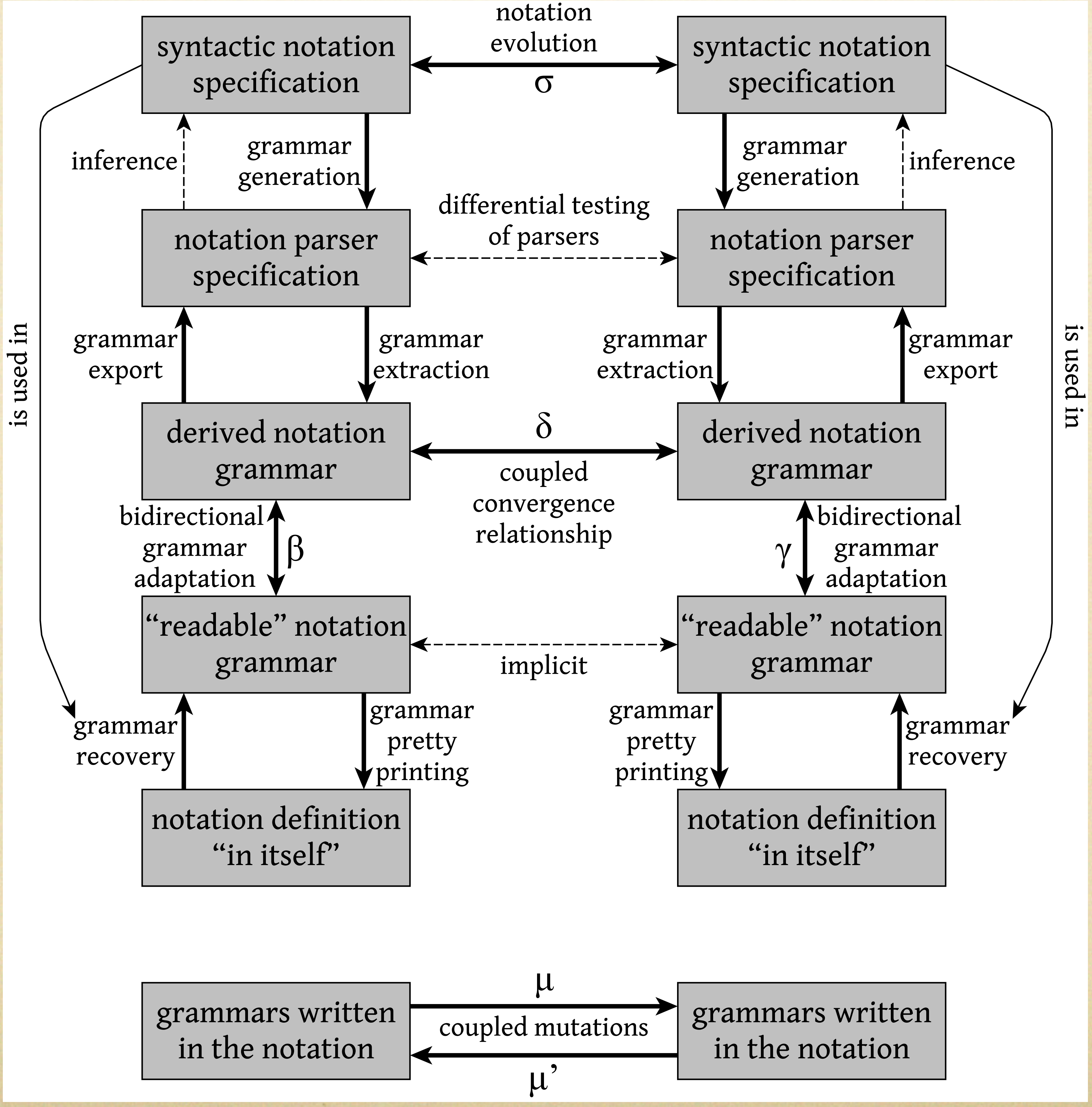
Notation evolution summary

To conclude, a notation evolution step Δ consists of the following components:

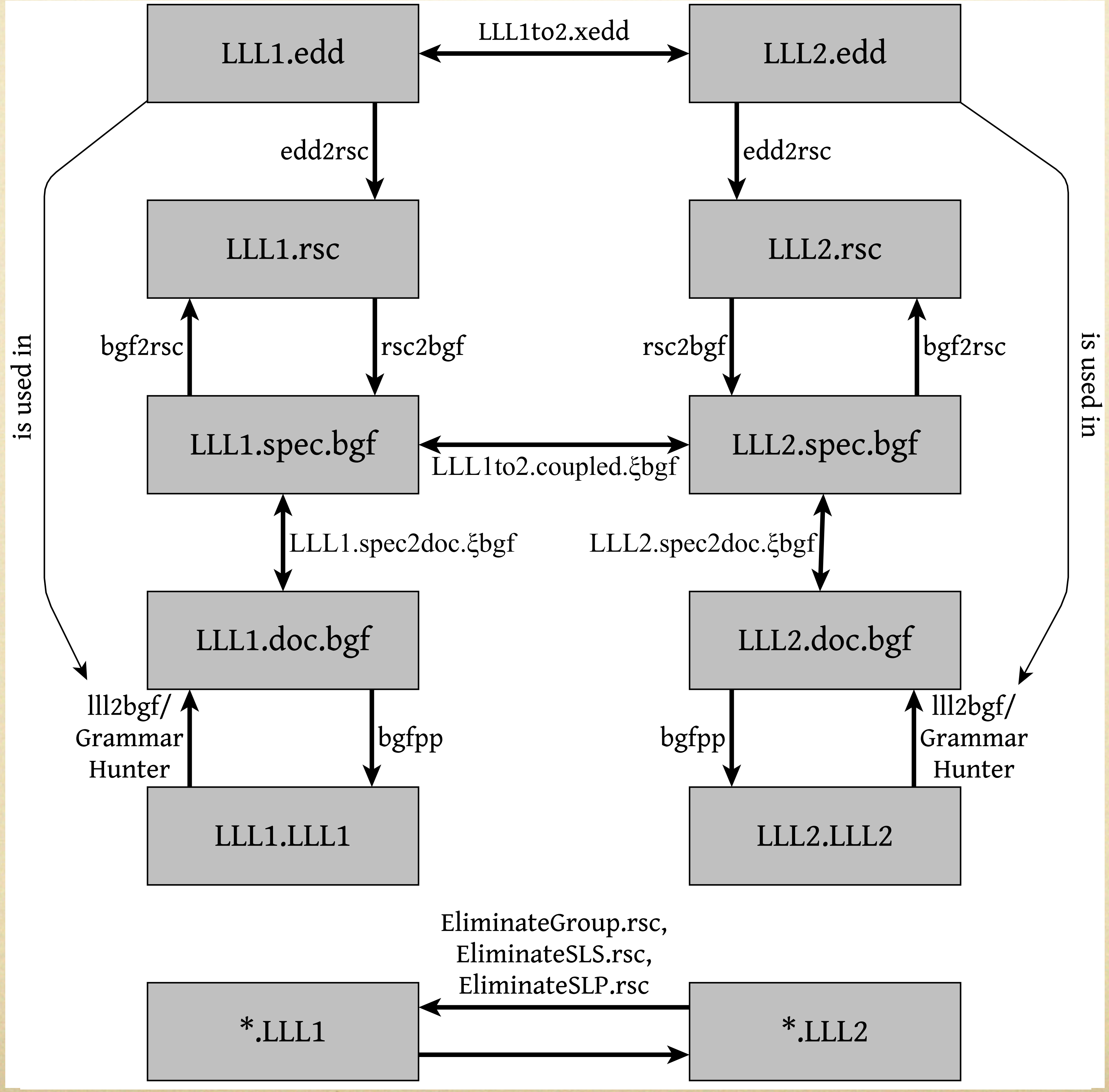
- σ, a bidirectional notation specification transformation that changes the notation itself
- δ, a bidirectional coupled grammar transformation that converges the notation grammars
- μ, an unidirectional coupled grammar mutation that migrates the grammarbase according to notation changes.
- a mechanism to propagate naming changes to form $\gamma = \delta_n^{-1} \circ \beta$

LLL case study

The megamodel



The megamodel



$\Delta = \langle\sigma,\delta,\mu\rangle$

The σ between $\hat{L}_1^3$ and $\hat{L}_2^3$, expressed in XEDD, looks like this (see `lll1to2.xedd`):
introduce-metasymbol(*group*, '(', ')');
introduce-metasymbol(*seplist-star*, '{', '}$\star$');
introduce-metasymbol(*seplist-plus*, '{', '}+');
The coupled δ generated by the `xedd` processor produces the following ΞBGF:
rename-rename(*LLL1Grammar*, *LLL2genGrammar*);
rename-rename(*LLL1Production*, *LLL2genProduction*);
rename-rename(*LLL1Definition*, *LLL2genDefinition*);
rename-rename(*LLL1Symbol*, *LLL2genSymbol*);
rename-rename(*LLL1Nonterminal*, *LLL2genNonterminal*);
rename-rename(*LLL1Terminal*, *LLL2genTerminal*);
add-remove(**p**(**l**(*group*), *LLL2genSymbol*, *','*(**t**(*'('*),**slp**(*LLL2genDefinition*,'|'),**t**(*')'*))));
add-remove(**p**(**l**(*sepliststar*), *LLL2genSymbol*, *','*(**t**('{'),**n**(*LLL2genSymbol*),**n**(*LLL2genSymbol*),**t**('}$*$'))));
add-remove(**p**(**l**(*seplistplus*), *LLL2genSymbol*, *','*(**t**('{'),**n**(*LLL2genSymbol*),**n**(*LLL2genSymbol*),**t**('}+'))));

Propagation of nominal refactorings from δ to β to form γ is performed by an XSLT script `ξbgf`[2]. In general, propagating structural changes is hard and sometimes impossible (for some transformations, there is no easy way to express their permutation in XBGF), and in this particular scenario is even undesirable. We save space in the paper by reserving it for future work.

Applying coupled mutation **eliminate-metasymbol(group)** to Grammar Zoo

ada-kellogg	108	csharp-iso-23270-2003	0	java-1-jls-read	0
ada-kempe	89	csharp-iso-23270-2006	0	java-2-jls-impl	36
ada-laemmel-verhoef	79	csharp-msft-ls-1.0	0	java-2-jls-read	0
ada-lncs-2219	89	csharp-msft-ls-1.2	0	java-5-habelitz	65
ada-lncs-4348	109	csharp-msft-ls-3.0	0	java-5-jls-impl	60
c-iso-9899-1999	0	csharp-msft-ls-4.0	0	java-5-jls-read	1
c-iso-9899-tc2	0	csharp-zaytsev	23	java-5-parr	95
c-iso-9899-tc3	0	dart-google	58	java-5-stahl	92
cpp-iso-14882-1998	0	dart-spec-0.01	56	java-5-studman	91
cpp-iso-n2723	0	dart-spec-0.05	62	mediawiki-bnf	32
csharp-ecma-334-1	0	eiffel-bezault	45	mediawiki-ebnf	30
csharp-ecma-334-2	0	eiffel-iso-25436-2006	345	modula-sdf	50
csharp-ecma-334-3	0	fortran-derricks	101	modula-src-052	65
csharp-ecma-334-4	0	java-1-jls-impl	0	w3c-xpath1	3

Related and future work

Related work

- ★ Cicchetti et al: coevolution of models/metamodels (syntax/metasyntax) with language evolution and language coevolution happening simultaneously.
- ★ Wachsmuth: MDA/MOF solution, close to us.
- ★ Stevens: formulated properties like correctness and hippocraticness; need further investigation.

Future work

- Extract reference grammars from compiler sources
  - rare enabling precondition
  - known to be successful at least once [C500LP]
- Derive β from inline edits of the definition “in itself”
  - possible if edits are purely decorational
  - makes sense in context of IDE (structural editors?)
- Propagate all refactorings from δ to β to form γ

Conclusion

Conclusion

- ★ We extended XBGF to bidirectionality, resulting in ΞBGF.
- ★ We proposed EDD and XEDD for notation & its evolution.
- ★ We presented a case study of LLL evolution (GDK).
- ★ We generalised transformers and generators to transformations and mutations; also formalised them.
- ★ We implemented an XEDD processor for evolution, coevolution, change propagation and mutation.

Bibliography

Bibliography

[THIS]
Zaytsev,
Language Evolution, Metasyntactically
BX and hopefully ECEASST
[GRACE]
Czarnecki, Foster, Hu, Lämmel, Schürr, Terwilliger,
Bidirectional Transformations: A Cross-Discipline Perspective
GRACE 2008 meeting notes, state of the art, and outlook
[C500LP]
Lämmel, Verhoef,
Cracking the 500-Language Problem
IEEE Software, November/December 2001
[Z12a]
Zaytsev,
BNF WAS HERE: What Have We Done About the Unnecessary Diversity of Notation for Syntactic Definitions
SAC/PL 2012
[LDTA02]
Kort, Lämmel, Verhoef,
The Grammar Deployment Kit: System Demonstration
LDTA 2002 and ENTCS 65
[GDK]
Kort,
Grammar Deployment Kit Reference Manual
Universiteit van Amsterdam, May 2003

Discussion