Конструктивная логика как синтаксический формализм

Артем Андреев
Институт лингвистических исследований РАН
artem@iling.spb.ru

24 мая 2008 г.

Категориальные грамматики и конструктивная логика

Логика

λ-исчисление Чёрча (1936)
Комбинáторная логика Карри (1958)

Простое типизованное λ-исчисление Чёрча (1940)

Система F Жирара и Рейнолдса (1971)

Исчисление конструкций Кокана (1986)

Индуктивное исчисление конструкций (1990)

Линейная логика Жирара (1987)

Лингвистика

Семантические категории Лесневского (1929)

Синтаксические связки Айдукевича (1935)

Квазиарифметическая нотация Бар-Хиллеля (1953)

Исчисление Ламбека (1958)

Аппликативная универсальная грамматика Шаумяна (1965)

Комбинáторная категориальная грамматика Стидмена (1987)

Формальная семантика Монтегю (1974)

Исчисление Ламбека

$$\dfrac{\text{John} :: n \qquad \dfrac{\text{likes} :: (n\backslash s)/n \qquad \text{music} :: n}{\text{likes music} :: n\backslash s}}{\text{John likes music} :: s}$$

Исчисление Ламбека vs исчисление высказываний

Исчисление высказываний

Modus Ponens

$$\frac{P \to Q \qquad P}{Q}$$

Исчисление Ламбека

$$\frac{w :: p/q \qquad w' :: q}{ww' :: p}$$

$$\frac{w' :: q \qquad w :: p\backslash q}{w'w :: p}$$

Исчисление Ламбека vs λ-исчисление

λ-исчиление

$$\frac{\lambda x.\, A : p \to q \qquad B : p}{A[B\backslash x] : p}$$

Исчисление Ламбека

$$\frac{w :: p/q \qquad w' :: q}{ww' :: p}$$

$$\frac{w' :: q \qquad w :: p\backslash q}{w'w :: p}$$

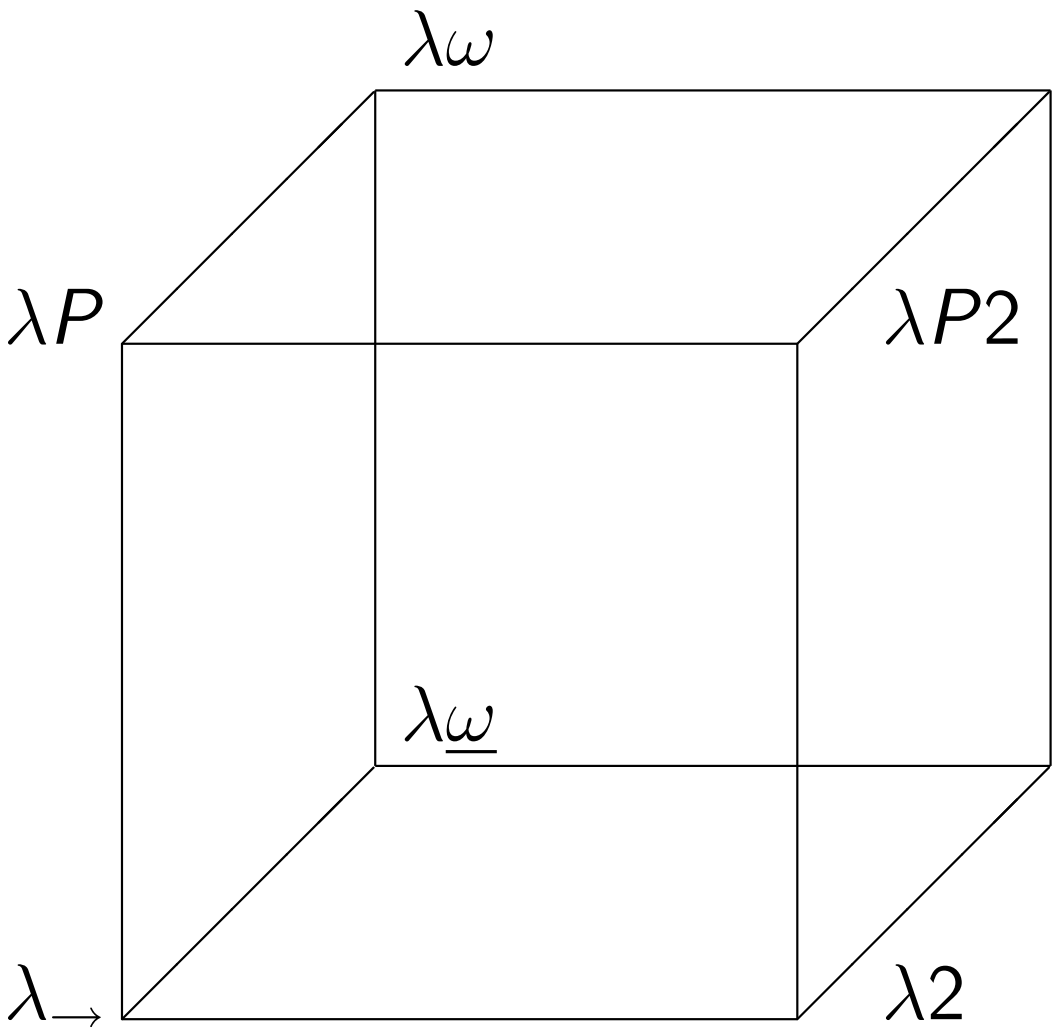
Категориальные грамматики

Синтаксический формализм строится по аналогии с логическим

Предлагаемый нами подход

Синтаксический формализм является подсистемой логического

λ-куб


$\lambda\omega$
$\lambda P\omega$
Исчисление конструкций
λP
$\lambda P2$
$\lambda\underline{\omega}$
$\lambda P\underline{\omega}$
$\lambda\rightarrow$
$\lambda 2$
Простое типизованное λ-исчисление

Логический взгяд на КС-грамматики

$$\forall x \forall y \ldots$$

S	$\rightarrow$	$NP\ VP$	$NP(x)\&VP(y)$	$\rightarrow$	$S(x \frown y)$
NP	$\rightarrow$	$Pron$	$Pron(x)$	$\rightarrow$	$NP(x)$
	$\rightarrow$	$Det\ NP'$	$Det(x)\&NP'(y)$	$\rightarrow$	$NP(x)$
NP'	$\rightarrow$	$Adj\ NP'$	$Adj(x)\&NP'(x)$	$\rightarrow$	$NP'(x \frown y)$
	$\rightarrow$	$Noun$	$Noun(x)$	$\rightarrow$	$NP'(x)$

Исчисление конструкций: базовые правила

$\forall$-введение

$$\frac{\Gamma, x : P \vdash A : Q}{\Gamma \vdash (\lambda x : P.A) : (\forall x : P.Q)}$$

$\forall$-удаление

$$\frac{\Gamma \vdash A : \forall x : P.Q \qquad \Gamma \vdash B : P}{\Gamma \vdash AB : Q[P\backslash x]}$$

β-редукция

$$\frac{(\lambda x : P.y)z}{x[z\backslash y]}$$

Исчисление конструкций: производные правила

&-ВВЕДЕНИЕ

$$\frac{\Gamma \vdash A : P \qquad \Gamma \vdash B : Q}{\Gamma \vdash \langle A, B \rangle : P \& Q}$$

&-УДАЛЕНИЕ

$$\frac{\Gamma \vdash A : P \& Q}{\Gamma \vdash \pi_1 A : P \qquad \Gamma \vdash \pi_2 A : P}$$

∨-ВВЕДЕНИЕ

$$\frac{\Gamma \vdash A : P \qquad \Gamma \vdash Q : *}{\Gamma \vdash \iota_1 A : P \vee Q} \qquad \frac{\Gamma \vdash P : * \qquad \Gamma \vdash B : Q}{\Gamma \vdash \iota_2 B : P \vee Q}$$

∨-УДАЛЕНИЕ

$$\frac{\Gamma \vdash A : P \vee Q \qquad \Gamma \vdash B : P \to R \qquad \Gamma \vdash C : Q \to R}{\Gamma \vdash (B \mid C) A : R}$$

Σ-типы (зависимые пары)

$\exists$-введение

$$\frac{\Gamma \vdash A : P \qquad \Gamma \vdash Q : P \to * \qquad \Gamma \vdash B : QA}{\Gamma \vdash \langle A, B \rangle_Q : \exists x : P.Qx}$$

$\exists$-удаление

$$\frac{\Gamma \vdash A : \exists x : P.Qx}{\Gamma \vdash \mathbf{W}A : P \qquad \Gamma \vdash \Pi A : Qx}$$

Деревья составляющих

ЛИСТ

$$\frac{\Gamma, P : * \vdash A : P}{\Gamma, P : * \vdash \lfloor A \rfloor : \mathit{Tree}\ P}$$

УЗЕЛ

$$\frac{\Gamma, P : * \vdash A : \mathit{Tree}\ P \qquad \Gamma, P : * \vdash B : \mathit{Tree}\ P}{\Gamma, P : * \vdash A \frown B : \mathit{Tree}\ P}$$

ПРИНЦИП ИНДУКЦИИ

$$\frac{\Gamma, P : * \vdash Q : \mathit{Tree}\ P \to * \qquad \Gamma, P : * \vdash A : \forall x : P.Q\lfloor x \rfloor \qquad \Gamma, P : * \vdash B : \forall t_1, t_2 : \mathit{Tree}\ P.Qt_1 \to Qt_2 \to Q(t_1 \frown t_2)}{\Gamma, P : * \vdash \mathbf{T}QAB : \forall x : \mathit{Tree}\ P.Qx}$$

Индуктивные определения в Coq

Inductive $Tree$: Set :=
| $Leaf$: $S \rightarrow Tree$
| $Joint$: $Tree \rightarrow Tree \rightarrow Tree$.

Theorem $treeDecEq$: $\forall\ (x\ y : Tree), \{x = y\} + \{x \neq y\}$.

Theorem $leafNotJoint$: $\forall\ x : S, \forall\ (y\ z : Tree), Leaf\ x \neq Joint\ y\ z$.

Definition $LangSet := Tree \rightarrow$ Set.

Inductive $SubTree$: $Tree \rightarrow Tree \rightarrow$ Prop :=
| $subTreeRefl$: $\forall\ x : Tree, SubTree\ x\ x$
| $subTreeLeft$: $\forall\ (x\ y\ z : Tree), SubTree\ x\ y \rightarrow$
$SubTree\ x\ (Joint\ y\ z)$
| $subTreeRight$: $\forall\ (x\ y\ z : Tree), SubTree\ x\ y \rightarrow$
$SubTree\ x\ (Joint\ z\ y)$.

Формы

Definition *existG* (x : *Tree S*) := { l : *L* & *G* l x}.

Definition *decideG* (x : *Tree S*) := *sum* (*existG* x) (*failedG* x).

Definition *Form* := { x : *Tree S* & *decideG* x }.

Definition *decideNode* (x y : *Tree S*)
(dx : *decideG* x) (dy : *decideG* y) : *decideG* (*Joint* x y).

Definition *join* (x y : *Form*) :=
existS decideG (*Joint* (*projS1* x) (*projS1* y))
(*decideNode* (*projS2* x) (*projS2* y)).

В Coq нотация { x : P & Q x} используется для обозначения Σ-типа $\exists x : P.Qx$

Формы: неграмматичные конструкции

Record *Failure* (x : *Tree* S) : Set := *mkFailure* {
  failedTree : *Tree* S;
  failedSubtree : *SubTree* *failedTree* x;
  failedLabel : L;
  failedCat : G *failedLabel* *failedTree*
}.

Record *failedG* (x : *Tree* S) : Set := *mkFailedG* {
  failedLeft : *Failure* x;
  failedRight : *Failure* x;
  failureProof : $\neg$*Compatible* (*failedLabel* *failedLeft*)
                                (*failedLabel* *failedRight*)
}.

Унифицированный интерфейс грамматики

множество меток

```
Parameter L : Set.
```

предикат грамматической корректности

```
Parameter G : L → LangSet.
```

предикат совместимости меток

```
Parameter Compatible : L → L → Prop.
```

разрешающая процедура совместимости меток

```
Parameter compat_dec : ∀ (l1 l2 : L),
  {Compatible l1 l2} + {~Compatible l1 l2}.
```

процедура вычисления результирующей метки

```
Parameter compat_label : ∀ (l1 l2 : L) (x y : Tree),
                         ∀ (gx : G l1 x), ∀ (gy : G l2 y),
  Compatible l1 l2 → { t : L & G t (Joint x y) }.
```

Базовые категории в стиле исчисления Ламбека

Module *BasicCategory* <: *UnifiedGrammar*.
  Parameter *simpleLabel* : Set.
  Parameter *simpleLabelEqDec* : $\forall$ (*x y* : *simpleLabel*), {*x* = *y*} + {*x* ≠ *y*}.

  Inductive *CatLabel* : Set :=
  | *CatSimple* : $\forall$ *x* : *simpleLabel*, *CatLabel*
  | *CatFun* : *CatLabel* → *CatLabel* → *CatLabel*.

  Definition *L* := *CatLabel*.

  Parameter *primCatType* : *CatLabel* → *S* → Set.
  Inductive *CatType* : *CatLabel* → *LangSet* :=
  | *LeafCategory* : $\forall$ *l* : *CatLabel*, $\forall$ *s* : *S*,
    primCatType l s → *CatType l* (*Leaf s*)
  | *JoinCategory* : $\forall$ (*l1 l2* : *CatLabel*), $\forall$ (*x y* : *Tree*),
    CatType (*CatFun l1 l2*) *x* → *CatType l1 y* → *CatType l2* (*Joint x y*).
...

End *BasicCategory*.

Объединение грамматик

Module *ConjCategory* (*G1 G2* : *UnifiedGrammar*) <: *UnifiedGrammar*.

Definition *ConjLabel* := *prod G1.L G2.L*.

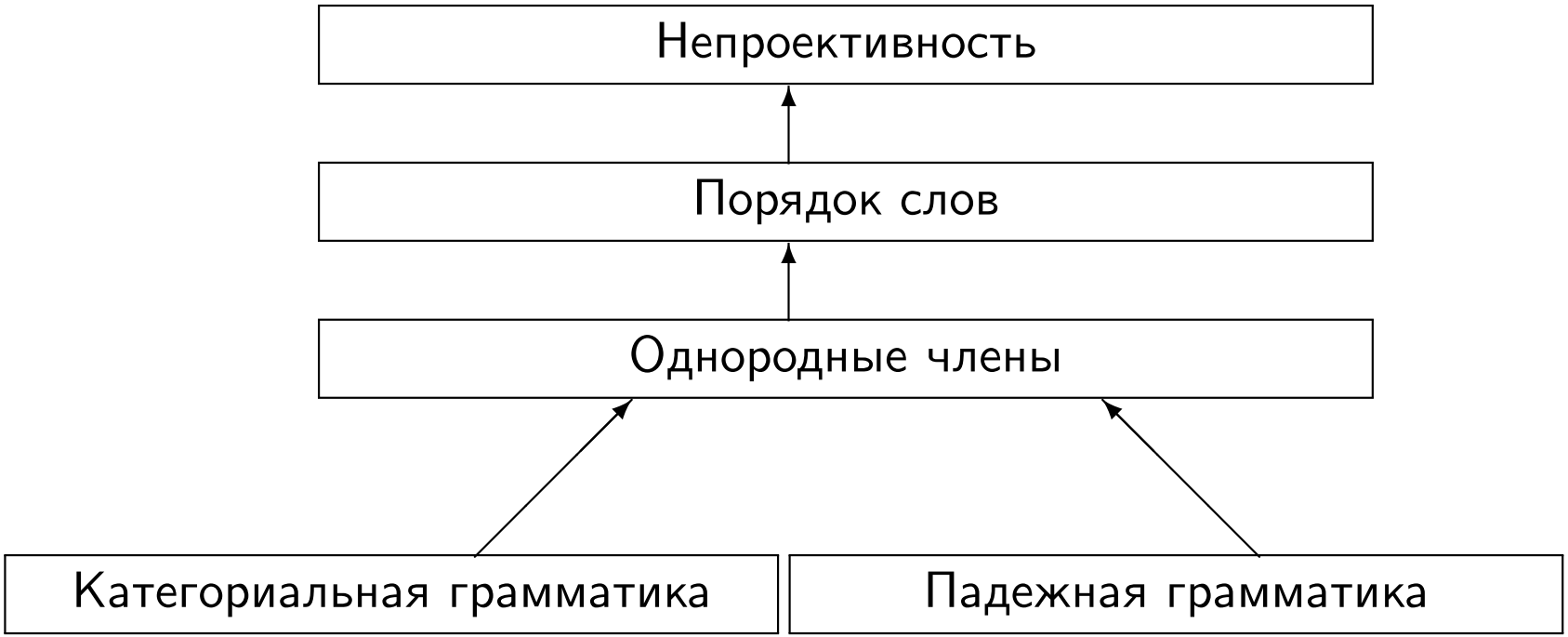
Definition *ConjG* (*l* : *ConjLabel*) (*x* : *Tree*) := *prod* (*G1.G* (*fst l*) *x*) (*G2.G* (*snd l*) *x*).

Definition *ConjCompatible* (*l1 l2* : *ConjLabel*) :=
G1.Compatible (*fst l1*) (*fst l2*) ∧ *G2.Compatible* (*snd l1*) (*snd l2*).

Definition *conjCompatibleDec* (*l1 l2* : *ConjLabel*) :
{*ConjCompatible l1 l2*} + {~*ConjCompatible l1 l2*}.

Definition *conjLabelCompatLabel* (*l1 l2* : *ConjLabel*) (*x y* : *Tree*) (*gx* : *ConjG l1 x*) (*gy* : *ConjG l2 y*) :
ConjCompatible l1 l2 → { *t* : *ConjLabel* & *ConjG t* (*Joint x y*) }.
End *ConjCategory*.

Модуляризация грамматики



Ссылки

Coq	http://coq.inria.fr
Agda	http://appserv.cs.chalmers.se/users/ulfn/wiki/agda.php
Epigram	http://www.e-pig.org
Isabelle/HOL	http://isabelle.in.tum.de
Maude	http://maude.cs.uiuc.edu

Исчисление конструкций http://www.inria.fr/rrrt/rr-0530.html

Модульная грамматика

http://www.swiss.ai.mit.edu/~dae/related-papers/steele.ps.Z