

# Управление программной на естественном языке

Е. А. Черкашин

ИДСТУ им. В. М. Матросова СО РАН

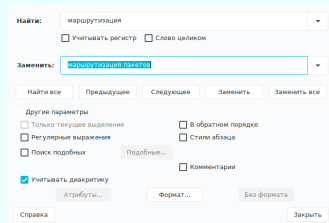
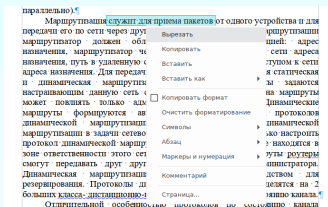
7 ноября 2019, Иркутск

**Направление ИИ** – Понимание естественного языка (ЕЯ), перевод из одного ЕЯ на другой. Решаются следующие задачи:

1. Анализ текстов, помещение изъятой информации в базу данных:
  - ▶ изготовление шаблонов документов, отчетов;
  - ▶ синтез структур данных для ИС;
  - ▶ заполнение баз данных ИС и т.п.;
2. Ведение диалога с пользователем:
  - ▶ идентификация моделей и планирование действий (интеллектуальные пользовательские интерфейсы);
  - ▶ приобретение знаний (оболочки экспертных систем);
3. Управление приложением:
  - ▶ запросы на естественном языке к базам данных;
  - ▶ внесение изменений в данные;
  - ▶ командное управление («Проветрить квартиру»).

# Графический пользовательский интерфейс

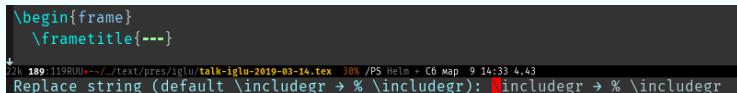
Специализирован на **унарных операциях над отдельными частями** информационного объекта.



В операциях аргументы вводятся в диалоговом окне (для операций с аргументами).

Более «умные» редакторы не используют контекстные операции (EMACS, VI, Visual Studio Code, Sublime, AutoCAD).

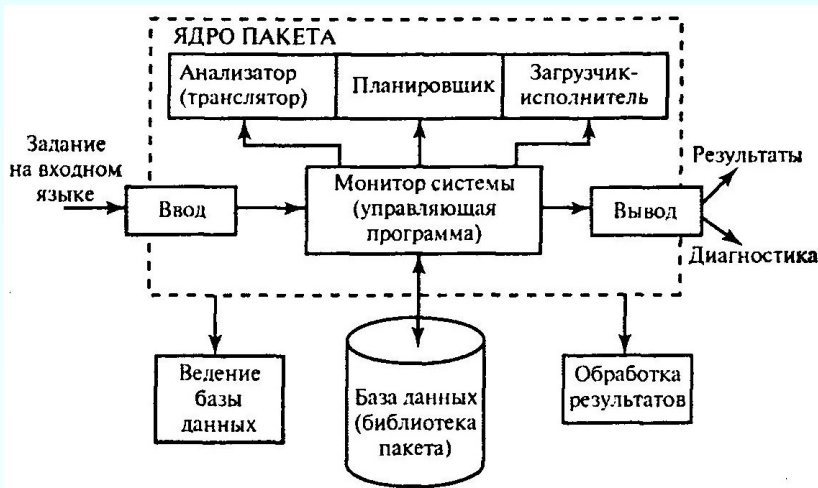
**Alt-X replace-string**, Набирается как «Alt-X repl str»



## Чего бы мы хотели от такой системы?

1. Представление запросов в терминологии предметной области ИС;
2. Выполнение двуаргументных операций над объектами целого класса (не только с экземплярами);
3. Система должна синтезировать ответ/действие на основе
  - ▶ введенного запроса/команды на ЕЯ,
  - ▶ данных и знаний предметной области;
4. Расширяемость/замена предметной области;
5. (в идеале) задавать утончающие вопросы;
6. (в идеале) обучаться (синонимы, новые декларативные конструкции).

# Пакеты прикладных программ



input a.txt multiply by b.txt output c.txt

Семиотика (наука о знаках) делится на три раздела (Моррис)

- ❑ **Семантика** — отношение знака к объекту (естествознание).
- ❑ **Синтаксис** — отношение знаков между собой (лингвистика, логика).
- ❑ **Прагматика** — отношение знака к субъекту (психология).

# Язык программирования

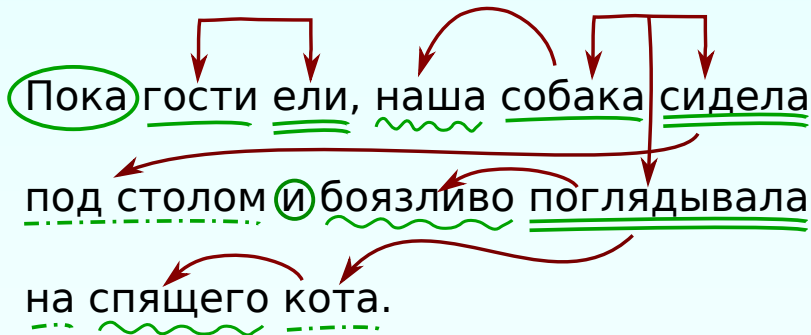
**Синтаксис** на уровне грамматики определяет корректные последовательности символов (лексемы, операторы, структуры). Но синтаксическая правильность не гарантирует даже осмысленности программы.

**Семантика** — это соответствие между синтаксически правильными программами и действиями абстрактного исполнителя, то есть это смысл синтаксических конструкций.

**Цель программиста** — получить нужный ему эффект в результате исполнения программы на конкретном оборудовании: трансляция и исполнение осуществляется на конкретных вычислителях.

**Прагматика** — задает конкретизацию абстрактного вычислителя для данной вычислительной системы. Стандарт языка программирования не полностью задаёт поведение исполнителя, которое производителями транслятора языка реализуется на конкретной программно-аппаратной платформе. Реализованный язык является прагматическим опосредованием абстрактной модели вычислений и возможностью ее реализации.

# Синтаксический разбор предложения





$$G = \langle T, N, \Sigma, R \rangle$$

$T$  – множество терминальных символов (слова, буквы, **IF**, **ELSE**),

$N$  – множество нетерминальных символов (обозначения,  $A, B, \langle \text{noun} \rangle, \langle \text{verb} \rangle$ ),  $T \cap N = \emptyset$ ,

$\Sigma$  – стартовый символ ( $\langle \text{программа} \rangle, \langle \text{предложение} \rangle$ ),  $\Sigma \in N$ ,

$R$  – множество правил грамматики

$A \rightarrow B$ ,

$R \subset ((T \cup N)^* N (T \cup N)^*) \times (T \cup N)^*$ .

Язык  $L(G) = \{ \Omega \in T^* \mid \Sigma \rightarrow^* \Omega \}$ .

Вывод  $\Sigma \rightarrow^* \Omega$

$\Sigma \rightarrow \Sigma A \quad \Sigma \rightarrow A$

$A \rightarrow b \Sigma e \quad A \rightarrow be$

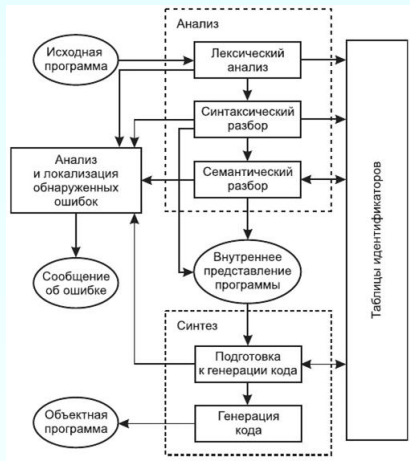
Пример:  $a = ' (', \quad b = ')'$ .

$\Sigma$	$\Sigma$
$A$	$A$
$b \Sigma e$	$(\Sigma)$
$b \Sigma A e$	$(\Sigma A)$
$b A A e$	$(A A)$
$b b e A e$	$(( ) A)$
$b b e b e e$	$(( ) ( ))$

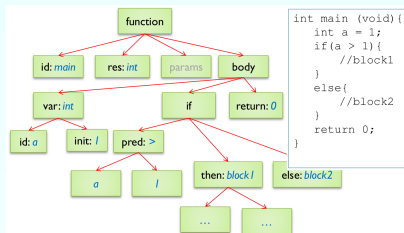
# Грамматика языка программирования С

```
<translation-unit> ::= {<external-declaration>}*
<external-declaration> ::= <function-definition>
                           | <declaration>
<function-definition> ::= {<declaration-specifier>}* <declarator> {<declaration>}* <compound-statement>
                           | union
<struct-declaration> ::= {<specifier-qualifier>}* <struct-declarator-list>
<specifier-qualifier> ::= <type-specifier>
                           | <type-qualifier>
<struct-declarator-list> ::= <struct-declarator>
                           | <struct-declarator-list> , <struct-declarator>
<struct-declarator> ::= <declarator>
                       | <declarator> : <constant-expression>
                       | : <constant-expression>
<selection-statement> ::= if ( <expression> ) <statement>
                           | if ( <expression> ) <statement> else <statement>
                           | switch ( <expression> ) <statement>
<iteration-statement> ::= while ( <expression> ) <statement>
                           | do <statement> while ( <expression> ) ;
                           | for ( {<expression>}? ; {<expression>}? ; {<expression>}? ) <statement>
<jump-statement> ::= goto <identifier> ;
                       | continue ;
                       | break ;
                       | return {<expression>}? ;
```

# Транслятор языка программирования



Внутреннее представление программы – абстрактное синтаксическое дерево.



По иерархии Ноама Хомского, грамматики делятся на **четыре** типа, каждый последующий является более ограниченным подмножеством предыдущего (но и легче поддающимся анализу):

- ❑ Тип 0. Неограниченные грамматики — возможны любые правила;
- ❑ Тип 1. Контекстно-зависимые грамматики — левая часть может содержать один нетерминал, окруженный «контекстом»; сам нетерминал заменяется непустой последовательностью символов в правой части;
- ❑ Тип 2. Контекстно-свободные грамматики — левая часть состоит из одного нетерминала;
- ❑ Тип 3. Регулярные грамматики — более простые, распознаются конечными автоматами.

# Пример трансляции

```
#include <stdio.h>

typedef unsigned long int ulint;

ulint fact (ulint n) {
    if (n==0) return 1;
    if (n==1) return 1;
    return n*fact(n-1);
}

int main() {
    ulint n = 10;
    printf("Factorial of %lu = %lu.\n",
        n, fact(n));
    return 0;
}
```

```
.file      "fact.c"
.text
.globl    fact
.type     fact, @function

fact:
.LFB11:
    movl   $1, %eax
    cmpq   $1, %rdi
    jbe    .L4

.L3:
    imulq   %rdi, %rax
    subq    $1, %rdi
    cmpq    $1, %rdi
    jne    .L3

.L4:
    ret

.LFE11:
.section   .rodata.str1.1,"aMS",@progbits,1
.LC0:
.string    "Factorial of %lu = %lu.\n"
.section   .text.startup,"ax",@progbits
.globl     main
.type      main, @function

main:
.LFB12:
;; . . . . .
    ret
.cfi_endproc

.LFE12:
.size      main, .-main
.ident     "GCC: (GNU) 8.2.1 20181127"
.section   .note.GNU-stack,"",@progbits
```

# Трансляция предложений естественного языка

```
grammar = nltk.parse_cfg("""
```

```
S → NP VP
```

```
NP → Det Nom | Det Nom PP | PropN
```

```
Nom → Adj Nom | N
```

```
VP → V | V NP | V NP PP | V S
```

```
PP → P NP
```

```
PropN → 'John' | 'Mary'
```

```
Det → 'the' | 'a'
```

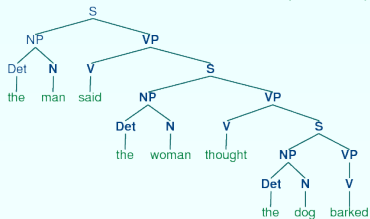
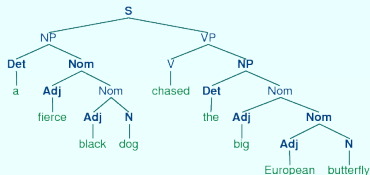
```
N → 'man' | 'woman' | 'park' | 'dog' |  
    'lead' | 'telescope' | 'butterfly'
```

```
Adj → 'fierce' | 'black' | 'big' |  
      'European'
```

```
V → 'saw' | 'chased' | 'barked' |  
    'disappeared' | 'said' | 'reported'
```

```
P → 'in' | 'with'
```

```
""")
```



# Linked grammar (грамматики связей)

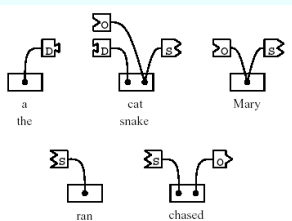
Последовательность слов находится в linked grammar, если существует способ нарисовать связи между словами, такие что

Связи не пересекаются (**планарный граф**);

Все слова последовательности соединены связями (**связность**);

Все связи удовлетворяют ограничениям (**непротиворечивость**)

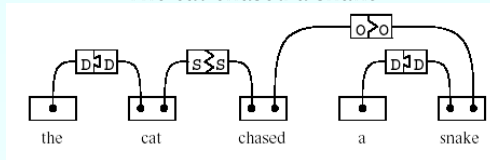
Два слова соединены одной и только одной связью (**исключительность**).



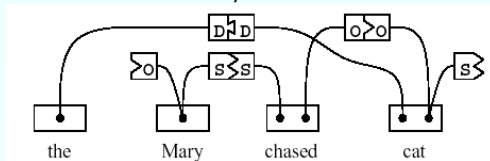
words	formula
a the	D+
snake cat	D- & (O- or S+)
Mary	O- or S+
ran	S-
chased	S- & O+

# Примеры разбора Linked grammar

“The cat chased a snake”



“The Mary chased cat”

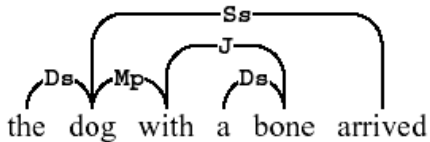
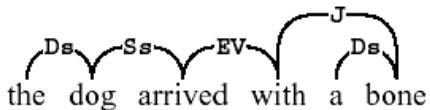




# Примеры разбора Linked grammar

“A dog arrived with a bone”

“A dog with a bone arrived”



## Примеры разбора Linked grammar

pinkparser> GABA mediates the inhibitory effect of NO on the AVP and OXT responses to insulin-induced hypoglycemia.

```
++++Time
```

0.47 seconds (0.47 total)

Found 129 linkages (75 had no P.P. violations)

```
Linkage 1, cost vector = (UNUSED=0 DIS=0 AND=0 LEN=39)
```

LEFT-WALL GABA mediates the inhibitory effect of NO on the AVP and OXT

responses.n to insulin-induced hypoglycemia[?].n

LEFT-WALL GABA mediates the inhibitory effect of NO on the AVP and OXT

responses.n to insulin-induced hypoglycemia[?].n

# Информационная система GeoBase. База данных

GeoBase – программа, позволяющая делать запросы на ЕЯ к базе данных по географии США, Borland, 1988.

```
state('alabama','al','montgomery',3894e3,51.7e3,22,'birmingham','mobile','montgomery','huntsville').
state('alaska','ak','juneau',401.8e3,591e3,49,'anchorage','fairbanks','juneau','sitka').

city('alabama','al','birmingham',284413).
city('alabama','al','mobile',200452).

border('florida','fl',['georgia','alabama']).

highlow('alabama','al','cheaha mountain',734,'gulf of mexico',0).

mountain('alaska','ak','mckinley',6194).
mountain('alaska','ak','st. elias',5489).

road('66',['district of columbia','virginia']).

lake('huron',59570,['michigan']).
```

The database contains the following information:

Information about states:

- Area of the state in square kilometers
- Population of the state in citizens
- Capital of the state
- Which states border a given state
- Rivers in the state
- Cities in the state
- Highest and lowest point in the state in meters

Information about rivers:

- Length of river in kilometers

Information about cities:

- Population of the city in citizens

Some sample queries:

- states
- give me the cities in california.
- what is the biggest city in california ?
- what is the longest river in the usa?
- which rivers are longer than 1 thousand kilometers?
- what is the name of the state with the lowest point?
- which states border alabama?
- which rivers do not run through texas?
- which rivers run through states that border the state with the capital austin?

# Схемы (спецификации) интерпретации

```
schema('abbreviation','of','state').
schema('state','with','abbreviation').
schema('capital','of','state').
schema('state','with','capital').
schema('population','of','state').

schema('area','of','state').
schema('city','in','state').

schema('length','of','river').
schema('state','with','river').
schema('river','in','state').

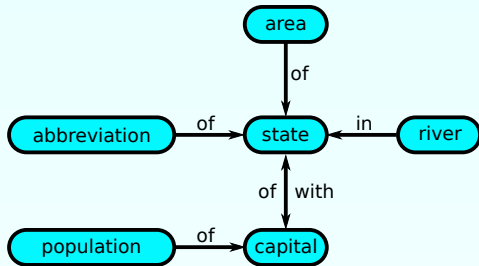
schema('capital','with','population').
schema('point','in','state').

schema('height','of','point').
schema('mountain','in','state').

schema('height','of','mountain').
schema('lake','in','state').

schema('name','of','river').
schema('name','of','capital').

schema('road','in','continent').
```



## Основной цикл программы

```
geobase(STR, X, E):-
    STR \= "",
    atom_string(ATOM,STR),
    tokenize_atom(ATOM,LIST),      /* Returns a list of words(symbols)      */
    filter(LIST,LIST1),            /* Removes punctuation and words to be ignored*/
    pars(LIST1,E,Q),               /* Parses queries                      */
    findall(A,eval_interp(Q,A),L),
    unik(L,L1),
    % unit(E,U),
    member(X,L1).
```

## Синтаксический анализатор (часть)

```
    pars(LIST,E,Q):-s_attr(LIST,OL,E,Q),check(OL),!.
    pars(LIST,_,_):-error(LIST),fail.
%
/* ... How big is the biggest city - BIG QUERY */
s_attr([BIG|S1],S2,E1,q_eaq(E1,A,E2,Q)):-
    size(_,BIG),s_minmax(S1,S2,E2,Q),
    size(E2,BIG),entitysize(E2,E1),
    schema(E1,A,E2),!.

s_attr(S1,S2,E,Q):-s_minmax(S1,S2,E,Q).
%
/* ... the shortest river in texas - MIN QUERY */
s_assoc1([MIN|S1],S2,E1,A,q_eaq(E1,A,E2,q_min(E2,Q))):-minn(MIN),!,
    s_nest(S1,S2,E2,Q),schema(E1,A,E2).

/* ... the longest river in texas - MAX QUERY */
s_assoc1([MAX|S1],S2,E1,A,q_eaq(E1,A,E2,q_max(E2,Q))):-maxx(MAX),!,
    s_nest(S1,S2,E2,Q),schema(E1,A,E2).
```

# Корпус и источник данных

population of Washington

Население штата или города?

1? schema('population', 'of', 'city').

2? schema('population', 'of', 'state').

Корпус реализован при помощи реструктуризации базы данных.

```
/* Relationships about states */
db(abbreviation,of,state,ABBREVIATION,STATE):- state(STATE,ABBREVIATION,_,_,_,_,_,_,_).
db(state,with,abbreviation,STATE,ABBREVIATION):-state(STATE,ABBREVIATION,_,_,_,_,_,_,_).
db(area,of,state,AREA,STATE):- state(STATE,_,_,_,AREA1,_,_,_,_,_),str_real(AREA,AREA1).
db(capital,of,state,CAPITAL,STATE):- state(STATE,_,CAPITAL,_,_,_,_,_,_,_).
db(state,with,capital,STATE,CAPITAL):-state(STATE,_,CAPITAL,_,_,_,_,_,_,_).
db(population,of,state,POPULATION,STATE):-state(STATE,_,_,POPUL,_,_,_,_,_,_,_),str_real(POPULATION,POPUL).
db(state,border,state,STATE1,STATE2):-border(STATE2,_,LIST),member(STATE1,LIST).

/* Relationships about rivers */
db(length,of,river,LENGTH,RIVER):- river(RIVER,LENGTH1,_,_),str_real(LENGTH,LENGTH1).
db(state,with,river,STATE,RIVER):- river(RIVER,_,LIST),member(STATE,LIST).
db(river,in,state,RIVER,STATE):- river(RIVER,_,LIST),member(STATE,LIST).

/* Relationships about points */
db(point,in,state,POINT,STATE):- highlow(STATE,_,POINT,_,_,_,_).
db(point,in,state,POINT,STATE):- highlow(STATE,_,_,_,POINT,_,_).
db(state,with,point,STATE,POINT):- highlow(STATE,_,POINT,_,_,_,_).
db(state,with,point,STATE,POINT):- highlow(STATE,_,_,_,POINT,_,_).
db(height,of,point,HEIGHT,POINT):- highlow(_,_,_,_,POINT,H),str_real(HEIGHT,H),!.
db(height,of,point,HEIGHT,POINT):- highlow(_,_,POINT,H,_,_,_),str_real(HEIGHT,H),!.
% . . . . .
```

Интерпретация запроса ...findall(A,eval\_interp(Q,A),L),

...

```
eval_interp(Q,IAns):-
    eval(Q,A),
    e_i(A,IAns).

eval(q_min(ENT,TREE),ANS):-
    forall(X,eval(TREE,X),L),
    entitysize(ENT,ATTR),
    sel_min(ENT,ATTR,99e99,'',ANS,L).

eval(q_max(ENT,TREE),ANS):-
    forall(X,eval(TREE,X),L),
    entitysize(ENT,ATTR),
    sel_max(ENT,ATTR,-1,'',ANS,L).

eval(q_sel(E,gt,ATTR,VAL),ANS):-
    schema(ATTR,ASSOC,E),
    db(ATTR,ASSOC,E,SVAL2,ANS),
    str_real(SVAL2,VAL2),
    VAL2>VAL.

% eval(q_eaq(E1,A,E2,TREE),ANS):-
%     eval(TREE,VAL),db(E1,A,E2,ANS,VAL).

eval(q_eaec(E1,A,E2,C),ANS):-db(E1,A,E2,ANS,C).

eval(q_e(E),ANS):-      ent(E,ANS). % EVAL "ATOM"

eval(q_or(TREE,_),ANS):- eval(TREE,ANS).

eval(q_or(_,TREE),ANS):- eval(TREE,ANS).

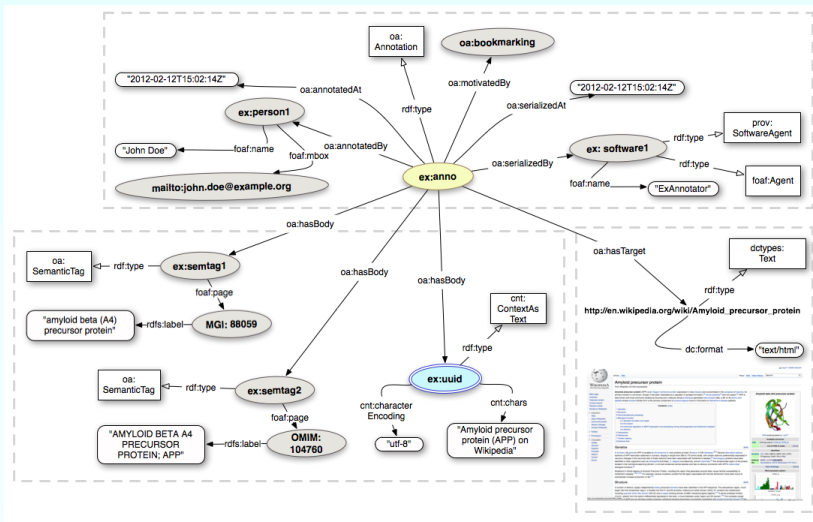
eval(q_and(T1,T2),ANS):- eval(T1,ANS1),eval(T2,ANS),ANS=ANS1.
```



В настоящее время появились новые варианты представления предметной области, при этом

- ❑ сущности идентифицируются, как правило, глобально;
- ❑ разработаны стандарты представления информации в конкретных предметных областях (oa, prov, foaf, ...);
- ❑ стандартизированы базовые онтологии (RDF, RDFs, dc, dcterms), при помощи которых задаются предметные онтологии;
- ❑ исследуются свойства логик описаний (DL, SHOIND, ...);
- ❑ предложены способы;
- ❑ созданы форматы представления данных (RDF, RDFa, OWL, JSON-LD, ...);
- ❑ существуют сервисы предоставления онтологической информации (dbpedia.org, lov.linkeddata.es/dataset/lov/), программное обеспечение создания таких сервисов (ClioPatria);
- ❑ язык запросов к онтологическим сервисам (SPARQL), технологии распределенных запросов к сервисам;
- ❑ инструментальные средства моделирования предметной области с выдачей в стандартные форматы (Protegé).

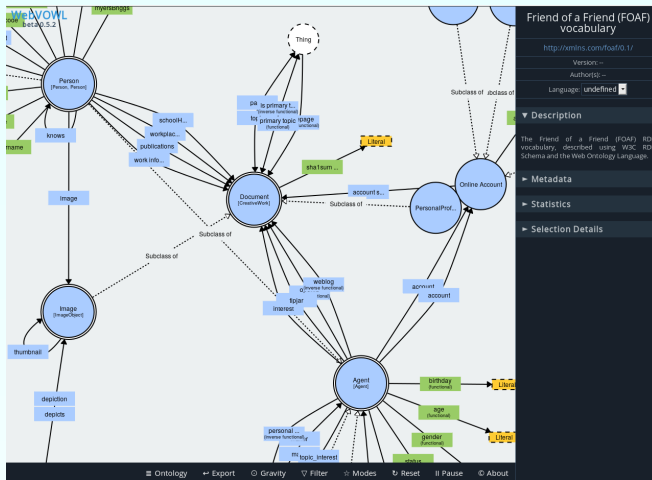
# Онтология Open Annotation (oa)



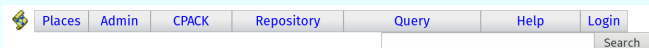
[illegible]

security - Security Ontology  
2016-08-12

# Инструменты: браузер онтологий



# Инструменты: сервер онтологий ClioPatria



## Local view for "http://cliopatria.swi-prolog.org/schema/cpack#gitURL"

Predicate	Value (sorted: <a href="#">default</a> )
<a href="#">rdfs:label</a>	"GIT URL"
<a href="#">rdfs:type</a>	<a href="#">owl:ObjectProperty</a>
<a href="#">rdfs:domain</a>	<a href="#">cpack:Software repository</a>
<a href="#">rdfs:range</a>	<a href="#">rdfs:Resource</a>
<a href="#">rdfs:comment</a>	"URL to clone the repository using git"
<a href="#">rdfs:subPropertyOf</a>	<a href="#">cpack:SCM URL</a>

All properties reside in the graph <http://cliopatria.swi-prolog.org/schema/cpack>

The resource does not appear as an object

## Predicate statistics

Predicate	#Triples	#Distinct subjects	#Distinct objects	Domain(s)	Range(s)
<a href="#">cpack:gitURL</a>	<a href="#">102</a>	<a href="#">102</a>	<a href="#">99</a>	<a href="#">2</a>	<a href="#">rdfs:Resource</a>

## Context graph



[cliopatria.swi-prolog.org/browse/list\\_graph?graph=http://cliopatria.swi-prolog.org/schema/cpack](http://cliopatria.swi-prolog.org/browse/list_graph?graph=http://cliopatria.swi-prolog.org/schema/cpack)

# Инструменты: редактор онтологий Protegé

Active Ontology x Entities x Classes x Object Properties x D

Class hierarchy (Inferred)  
Class hierarchy  
Class hierarchy: Feature

owl:Thing  
├── Fault  
│ ├── Associated\_CSS  
│ ├── Associated\_IGGSS  
│ └── Info  
│ ├── Compiler  
│ ├── Date  
│ ├── Location  
│ ├── Name  
│ ├── Reliability\_class  
│ ├── Last\_historical\_earthquake  
│ └── Seismic\_activity\_of\_fault  
├── Quality  
│ ├── ActivityDegree  
│ ├── EventParameter  
│ ├── FeatureQuality  
│ └── Unit  
├── Value  
│ ├── Angular  
│ ├── Event  
│ ├── Linear  
│ │ ├── Depth  
│ │ ├── Length  
│ │ ├── Rate  
│ │ └── Width  
│ └── Notional  
│ ├── Feature  
│ └── Slip  
│ └── Cenosoic\_Slip

Datatypes  
Annotation property hierarchy  
Individuals by type  
Data property hierarchy  
Object property hierarchy  
Object property hierarchy: featur

owl:topObjectProperty  
├── property  
│ ├── cenosoic  
│ ├── qualityProperty  
│ ├── quantityProperty  
│ └── value  
│ ├── activity  
│ └── feature

Class Annotations Class  
Annotations: Feature

Annotations +

Description: Feature

Equivalent To +

SubClass Of +  
Notional

General class axioms +

SubClass Of (Anonymous Ancestor)

Instances +

Target Ter Key +

Disjoint With +  
Value, Fault, Quality

Disjoint Union Of +

Asserted

```
[eugeneai@center data]$ head -n 50 activity_fall_data.ttl
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
@prefix geob: <http://www.semanticweb.org/bernard_black/ontologies/2016/3/
@prefix nie: <http://www.semanticdesktop.org/ontologies/2007/01/19/nie#> .
@prefix owl: <http://www.w3.org/2002/07/owl#> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix xml: <http://www.w3.org/XML/1998/namespace> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
```

```
[ ] a geob:Fault ;
    nie:identifier "RUAF_235-1" ;
    nie:title "Северобайкальский" ;
    geob:activity [ a geob:Activity ;
        geob:degree "повышенная" ;
        geob:eventage [ a geob:EventAge ;
            geob:index 2e+00 ;
            geob:type geob:LastActivationAge ;
            geob:value "Голоценовое" ] ;
        geob:value 1.4e+01 ] ;
    geob:angle [ a geob:Angle ;
        geob:quality "ЛС" ;
        geob:value "50-80" ] ;
    geob:azimuth [ a geob:Azimuth ;
        geob:quality "ЛС" ;
        geob:value 1.02e+02 ] ;
    geob:cenosoicSlip [ a geob:CenosoicSlip ;
        geob:quality "ЛС" ;
        geob:reliabilityClass 1e+00 ;
        geob:type geob:vertical ;
        geob:value 4e+03 ] ;
    geob:compiler [ a geob:Compiler ;
        nie:created "15.11.2014" ;
        foaf:name "Лунина О.В." ] ;
    geob:event [ a geob:Event ;
        geob:associatedCSS "Северобайкальский" ;
        geob:averaged_slip_rate_mm_year 3e+00 ;
        geob:isActiveFault "Да" ;
        geob:potential_ms_max 7.7e+00 ;
        geob:potential_mm_max 0e+00 ;
        geob:quality "ЛС" ;
        geob:slip_rate_mm_year "1-4.99" ;
        geob:type geob:TotalMaxSlip ;
        geob:value 9e+00 ] ,
    [ a geob:Event ;
        geob:type geob:LateralMaxSlip ;
        geob:value 0e+00 ] ,
    [ a geob:Event ;
        geob:type geob:VerticalMaxSlip ;
        geob:value 9e+00 ] ;
    geob:feature [ a geob:Feature ;
```

# Модификация GeoBase на основе Semantic WEB

```
schema('fault','in','continent'). % Связь с терминами GeoBase
schema('fault','with','feature').
schema('name','of','fault'). %
% schema('feature','of','fault'). % Это отношение находится в T-Box Faults''

% «Загрузка» онтологий
schema(Prop, 'of', SubjName):- % используется на этапе трансляции
    var(SubjName),
    geob_prop(Prop,_).

schema(Prop, 'of', SubjName):- % используется на этапе интерпретации
    nonvar(SubjName), % задан конкретный класс
    geob_prop(Prop, GProp),
    geob_ent_class(SubjName, Subj),
    rdf_reachable(Subj, rdfs:subClassOf, Parent),
    rdf(GProp, rdfs:domain, Parent),!.

geob_prop(Prop, GProp):- % Проверка свойства
    rdf_global_id(geob:Prop, GProp),
    rdf(GProp,rdf:type,owl:'ObjectProperty'),!.

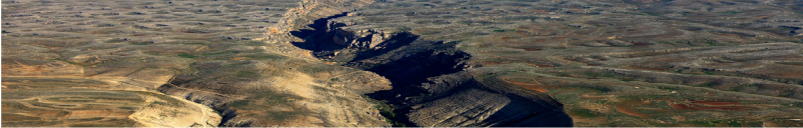
geob_class(Class, GClass):- % Проверка класса
    rdf_global_id(geob:Class, GClass),
    rdf(GClass,rdf:type,owl:'Class'),!.

geob_ent_class(Ent, Class):-
    sub_atom(Ent,0,1,R,H),
    sub_atom(Ent,1,R,0,T),
    string_upper(H,U),
    atom_concat(U,T,GEnt),
    geob_class(GEnt, Class).

geob_ent(E, A):-
    nonvar(E),
    geob_ent_class(E,Class),
    rdf(A, rdf:type, Class, geodata). % Сущность онтологии geodata
```

# Интерфейс пользователя GeoBase + ActiveFaults

← | ubuntu:3020/apps/geobase/index.html | Search



**Введите запрос**

show any names of faults

Go Clear

**Результат**

Entity	Type
Аблатуканский	name
Абчадский	name
Агайский	name
Аганайский	name
Агардагский	name
Агардагский (Эрзино-Агардагский)	name
Агардагский (Эрзинско-Агардагский)	name



В разработке программного обеспечения используются модели, представляющие объекты, бизнес-процессы и организационную систему предприятия (UML-2.4, SysML, BPMN-2.0, CMMN).

1. Формирование модели при помощи редакторов;
2. Экспорт модели в формат XMI;
3. Преобразование XMI в формат онтологии  
`<subject, predicate, object>`;
4. Преобразование базы данных в формат онтологии, или реализация адаптера;
5. Настройка связей с основной схемой данных;
6. Настройка корпуса, который также можно организовать при помощи адаптера базы данных.

Спасибо за внимание!