Инструментарий анализа, классификации и интерпретации сцен

Евгений Черкашин
eugeneai@icc.ru

г. Иркутск, Институт динамики систем и теории управления им. В. М. Матросова СО РАН

Применение методов классификации с использованием технологий искусственного интеллекта для семантической интерпретации текстовых данных в системах автоматизации деятельности органов военного управления

Современное состояние

- В настоящий момент ИИ чаще всего – это машинное обучение: нейронные сети
  - алгоритмические структуры свертки
  - генерирующие нейронные сети
  - модели регрессии, таксономии, классификации на основе машинного обучения и т. п.
- известное ограничение – невозможность интерпретации получаемых моделей в виде процедуры трансформации данных. ИНС – это набор коэффициентов.
- другое ограничение – сложность построения моделей для меняющихся во времени объектов (*динамических объектов и систем*). Например, последовательность сообщений между участниками общения может формировать некоторый план саботажа промышленного объекта.

Построение моделей АКИД на основе логического программирования (СОЗ) позволяет продвинуться в решении задач распознавания на основе сценариев.

Цели исследования

Основная цель НИРОКР – разработка технологии анализа текстов, базирующихся на формализованных знаниях. Решаются следующие основные проблемы:

1. разработка методик анализа, классификации и интерпретации данных (АКИД), полученных на этапах применения машинного обучения, синтаксического анализа текста
2. создание моделей анализа сценариев (статических и динамических)
3. разработка приемов реализации алгоритмов АКИД средствами системы программирования Logtalk
4. формирование принципов использования хранилищ семантической информации в АКИД
5. реализация тестовых и практически значимых приложений

Основная идея АКИД (на примере динамической сцены)

1. Распознавание динамических свойств взаимодействующих объектов:
   - как объекты переходят из состояния в состояние
   - какие свойства меняются при каждом переходе
   - каков допустимый набор этих состояний
   - какие состояния обладают критическими признаками для принятия решения

   Необходимо принять решение – соответствует ли сцена необходимому набору признаков.

Анализ и классификация базируется на построении иерархической системы моделей:

1. нижний уровень – результаты АКИД объектов методами МО
2. выше – модели статического аспекта – свойства объектов и связей между ними
3. модели высокого уровня описывают поток сцен – сцена высокого уровня.

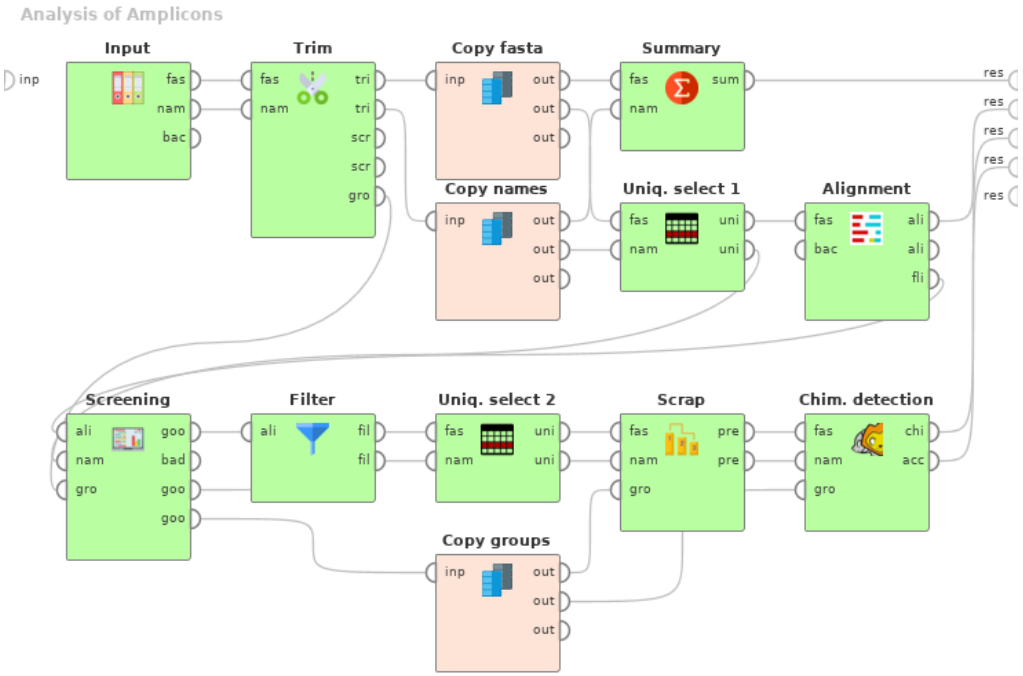
Сопоставление конкретной сцены одному из заданных сценариев – решение задачи классификации. При этом получаем и параметры компонентов сценария.

Интерпретация сценария

Интерпретация сценария – трансформация структуры и параметров компонент целевого сценария. Для этого создаются

- модель контекста интерпретации – как элементы сценария преобразуются в другие объекты

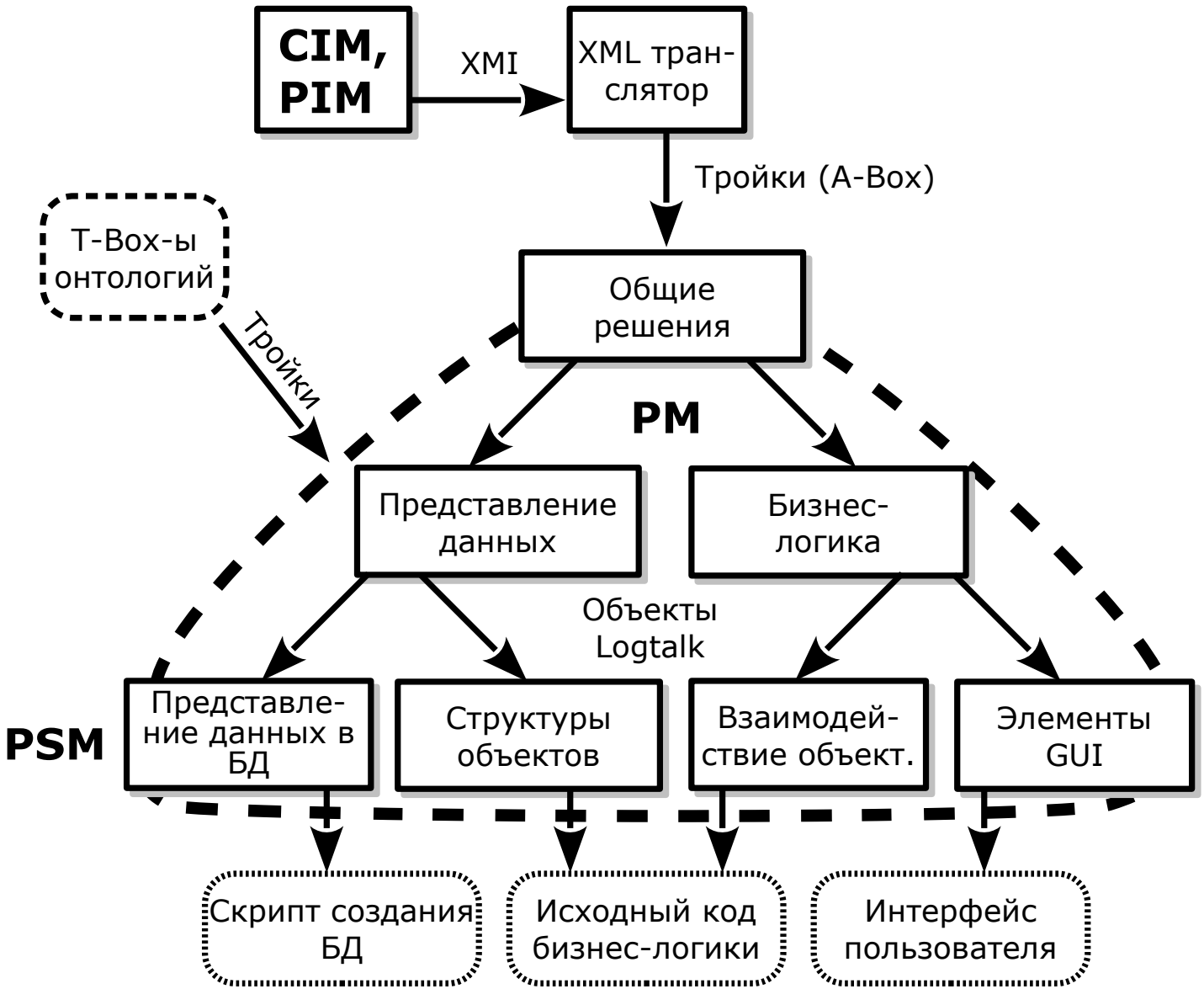
Разработана АКИД исходного кода прикладного пакета Mothur и порождения модулей Rapidminer studio (144 модуля).



Термин	Определение
NGS	Секвенирование нового поколения
Amplicon	Часть ДНК или РНК, скопированная много раз
Mothur	Пакет для исследований в NGS
Rapidminer studio	Визуальный редактор Dataflow-диаграмм

Зеленые блоки – модули Mothur, другие – модули Rapidminer studio.

Архитектура модулей трансформации



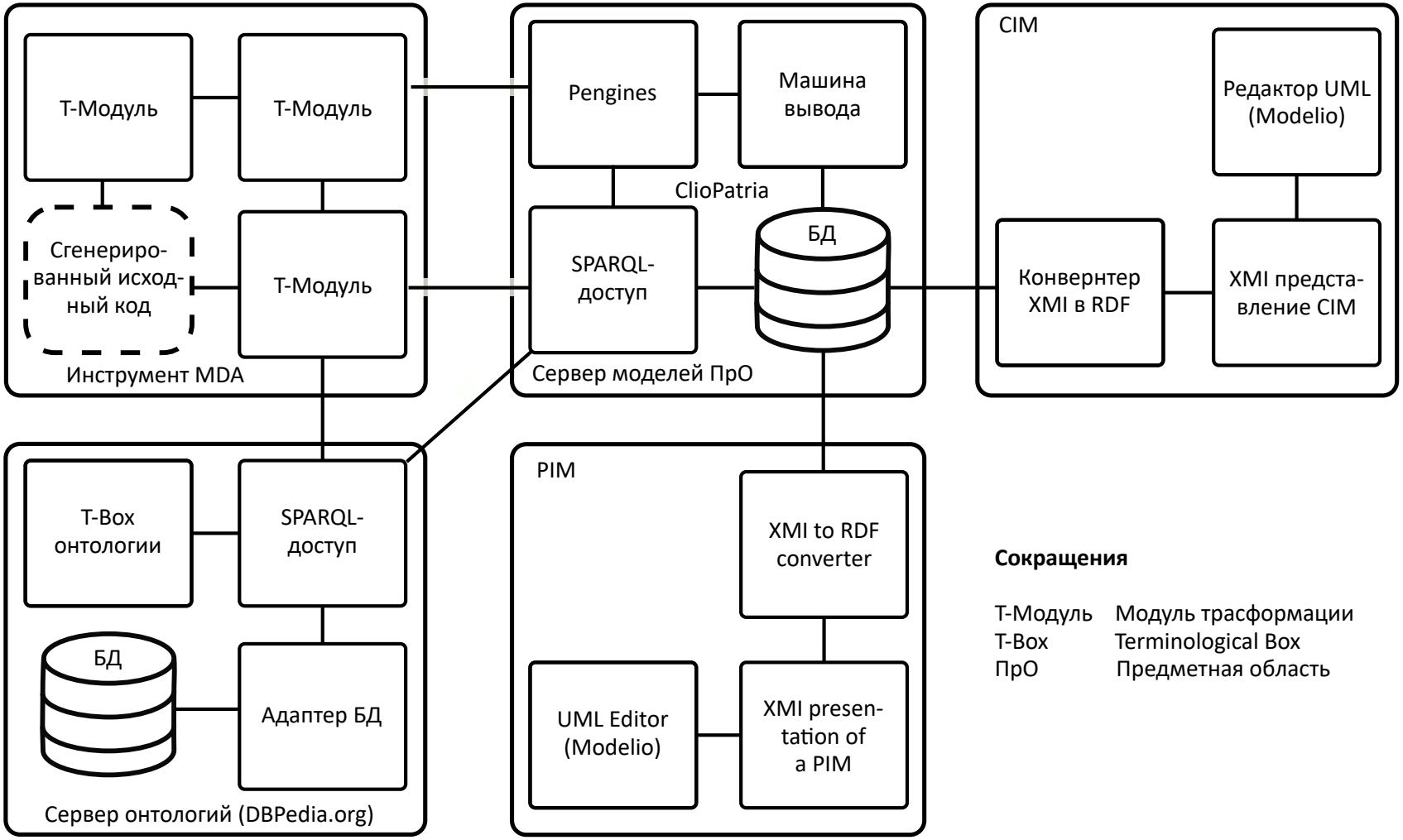
Сгенерированный модуль Rapidminer studio

```
vector<string> AlignCommand::setParameters(){ // PART OF MODULE SOURCE
try {
  CommandParameter ptemplate("reference", "InputTypes", "", "", "none", "none", "none","",false,true,true); para
  CommandParameter pcandidate("fasta", "InputTypes", "", "", "none", "none", "none","fasta-alignreport-accnos",f
  CommandParameter psearch("search", "Multiple", "kmer-blast-suffix", "kmer", "", "", "","",false,false,true); p
  CommandParameter pksize("ksize", "Number", "", "8", "", "", "","",false,false); parameters.push_back(pksize);
  CommandParameter pmatch("match", "Number", "", "1.0", "", "", "","",false,false); parameters.push_back(pmatch)
// . . . . . .
package com.rapidminer.ngs.operator; // GENERATED JAVA MODULE
// imports

class MothurChimeraCcodeOperator extends MothurGeneratedOperator {
  private InputPort fastaInPort = getInputPorts().createPort("fasta");
  private InputPort referenceInPort = getInputPorts().createPort("reference");
  private OutputPort chimeraOutPort = getOutputPorts().createPort("chimera");
  private OutputPort mapinfoOutPort = getOutputPorts().createPort("mapinfo");
  private OutputPort accnosOutPort = getOutputPorts().createPort("accnos");

  public MothurChimeraCcodeOperator (OperatorDescription description) {
    super(description);
  }
  @Override
  public void doWork() throws OperatorException {
    super();
    // . . . . . .
  }
  @Override
  public List<ParameterType> getParameterTypes() {
    super();
        // . . . . . .
  }
  @Override
  public String getOutputPattern(String type) {
    if (type=="chimera") return "[filename],[tag],ccode.chimeras-[filename],ccode.chimeras";
    if (type=="mapinfo") return "[filename],mapinfo";
    if (type=="accnos") return "[filename],[tag],ccode.accnos-[filename],ccode.accnos";
    return super.getOutputPattern(type);
  }
}
```

Представление структуры модулей Mothur в виде графа знаний

```
@prefix xml: <http://www.w3.org/XML/1998/namespace> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
ngsp:spec a ngsp:Specification ;
    ngsp:module mothur:NoCommand,
        mothur:align-check,
        mothur:align-seqs,
# . . . . .
mothur:align-check a ngsp:Module ;
    ngsp:outputPattern [ a cnt:Chars ;
            ngsp:parameterName "type" ;
            ngsp:pattern [ ngsp:patternString
                    "[filename],align.check" ;
                    dc:identifier "aligncheck" ] ;
            cnt:chars # . . . .
# . . . . .
mothur:align-check-idir-parameter a ngsp:Parameter ;
    ngsp:important false ;
    ngsp:multipleSelectionAllowed false ;
    ngsp:optionsDefault "" ;
    ngsp:required false ;
    ngsp:type mothur:String ;
    dc:title "inputdir" .

mothur:align-check-map-parameter a ngsp:Parameter ;
    ngsp:important true ;
    ngsp:multipleSelectionAllowed false ;
    ngsp:optionsDefault "" ;
    ngsp:required true ;
    ngsp:type mothur:InputTypes ;
    dc:title "map" .

mothur:align-check-name-parameter a ngsp:Parameter ;
    ngsp:chooseOnlyOneGroup "namecount" ;
    ngsp:important false ;
    ngsp:multipleSelectionAllowed false ;
# . . . . .
```

Инфраструктура сервисов



Сокращения

Т-Модуль	Модуль трасформации
Т-Box	Terminological Box
ПрО	Предметная область

Язык Logtalk

Язык Logtalk выбран в качестве языка представления трансформаций по следующим причинам:

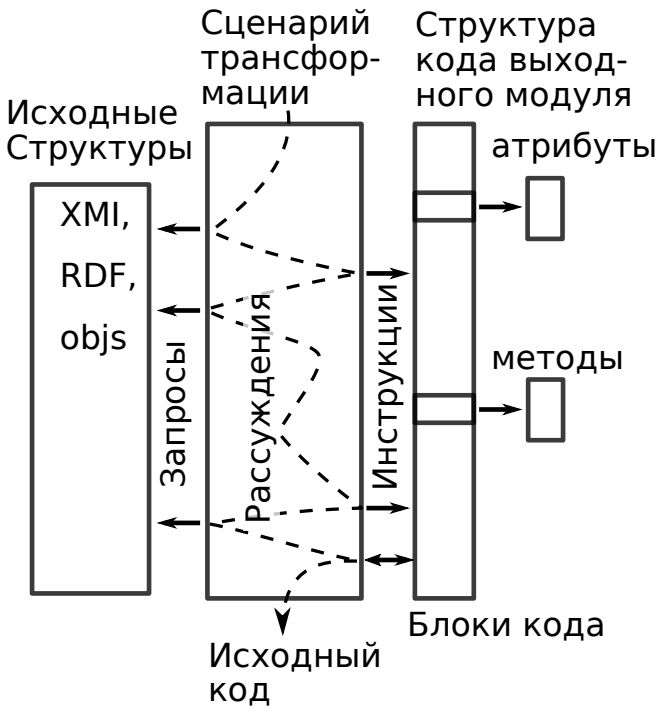
- наследует все свойства сред программирования **ISO-Prolog**;
- реализован в виде **макропакета**; накладные расходы в случае использования только статических объектов – 1.5%;
- гибкий подход к представлению трансформаций: синтаксически одинаково задаются как трансформации, так и ограничения;
- параметрические объекты (`circle(0,0,10)::square(S)`).
- реализация ОО-представления базы знаний (правил): структуризация, инкапсуляция, замена части и т.п.;
- представление объектов-трансформаций как композиций при помощи категорий;
- механизмы фильтрации при передаче сообщений;
- представлены реализации для разных ISO-Prolog-систем.

Применение SW для представления моделей

- Использование накопленного опыта и стандартов формализации предметных областей;
- Представление T-Box и A-Box при помощи множества троек (графа) `<субъект, отношение, объект>`;
- Элементы стандартных онтологий формально описаны (`rdfs:domain`, `rdfs:range`);
- Поддерживаются большинством программных систем (библиотеки, системы логического вывода, SPARQL);
- Предлагает способ глобальной идентификации объектов в различных независимых системах;
- SWI-Prolog включает библиотеку, позволяющую осуществлять запросы к графу, интерпретацию семантики некоторых отношений (`rdfs:label`, `dc:title`), инкапсуляцию BNode в многоаргументные предикаты; сервер онтологий ClioPatria;
- Предоставляется простой способ разделения уровней доступа к информации: (`rdfs:seeAlso`);
- Разметка RDF/LOD позволяет интегрировать гетерогенные системы.

Сценарий синтеза класса по сценарию

```
:- object(script(_Package_,_LocalProf_,_CodeProf_)). % Трансформационный профиль
  :- public([tr/4,tr/3]).                           % Публичный интерес сценария
  % . . . . . .
  tr(class, Class, ClassID):-   % Синтез класса
    % Запрос к структурам пакета
    query(_Package_)::class(Name, ClassID),
    create_object(Class, . . . . .% Создание объекта «Класс»
    create_object(Attributes,. . .% Создание атрибута
    create_object(Methods, . . . .% ... метода
    Class::name(Name),            % Поименование класса
    % Порождение атрибутов класса,
    % Представление их в виде локальной базы данных.
    % ... то же с методами ...
    Class::attributes(Attributes),  % Ассоциация атрибутов с классом
    Class::methods(Methods).        % ... и методов тоже..

    % Трансформация атрибутов
  tr(attribute, Attribute, ClassID, AttributeID):-
    query(_Package_)::attribute(Name,ClassID,AttrID),
    create_object(Attribute,  % . . . . .
    Attribute::name(Name).    % Поименование атрибута

    % Трансформация метода
  tr(method, Method, ClassID, MethodID):-
    query(_Package_)::method(Name,ClassID,MethodID),
    create_object(Method,     % . . . . .
    Method::name(Name).       % Поименование метода
  . . . . . .
  tr :-                       % Запуск АКИД
    forall( tr(class, Class, ClassID),
         ( forall(tr(attribute, A, ClassID, AID), true),
           forall(tr(method, M, ClassID, MID), true)).
:- end_object.
```



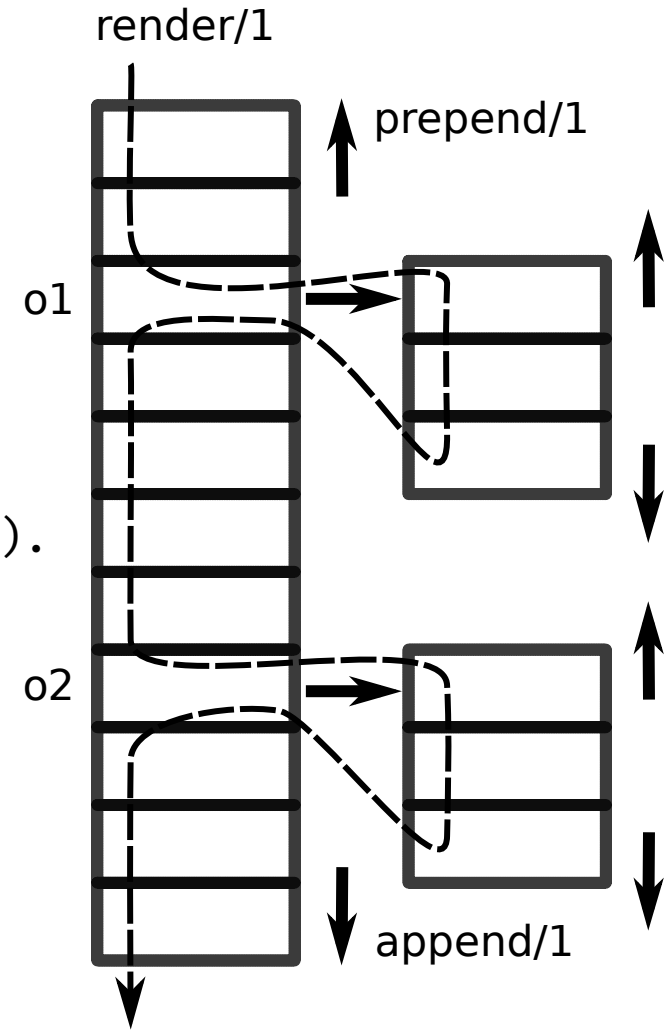
Реализация объекта-фасада query

```
:- object(query(_XMI_)).

  :- public([class/2, attribute/3, method/3]).
  class(Name, ID):-                                % Распознавание
    _XMI_::rdf(ID,rdf:type,uml:'Class'),           % класса
    _XMI_::rdf(ID,rdfs:label, literal(Name)).

  attribute(Name, ClassID, ID):-                   % ...атрибута...
    _XMI_::rdf(ClassID, xmi:ownedAttribute, ID),
    _XMI_::rdf(ID, rdfs:label, literal(Name)).

  method(Name, ClassID, ID):-                      % ...метода...
    _XMI_::rdf(ClassID, xmi:ownedOperation, ID),
    _XMI_::rdf(ID, rdfs:label, literal(Name)).
  % . . . . . .
:- end_object.
```

Блок кода (интерпретация объекта)

Идея реализации взята из llvmlite*)

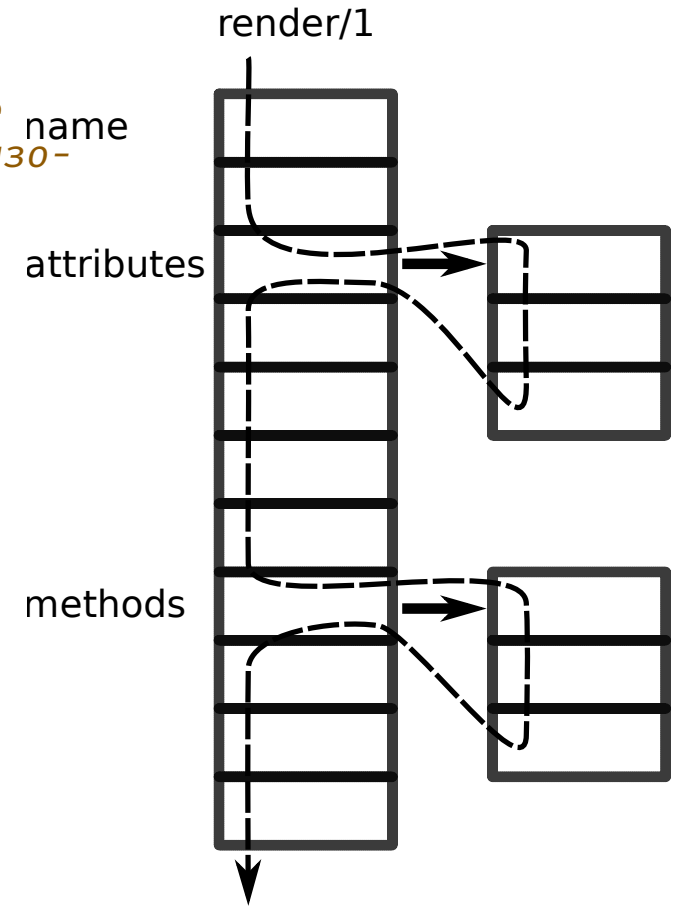
```
:- object(code_block, specializes(root)).
  % Публичный интерфейс объекта
  :- public([append/1, prepend/1, clear/0,
    render/1, render_to/1, remove/1,
    item/1, items/1]).
  % Элементы блока
  :- dynamic([item_/1]).
  :- private([item_/1]).
  % Специализации методов при наследовании
  :- protected([renderitem/2, render_to/2]).
  % Позволить объекту сгенерировать
  % свое представление самостоятельно
  renderitem(Object, String):-
    current_object(Object), !,
    Object::render(String).
  % Преобразовать литерал в строку
  renderitem(literal(Item), String):-!,
    atom_string(Item, String).
  % Отобразить как есть (для отладки).
  renderitem(Item, String):-
    root::iswritef(String, '%q', [Item]).
:- end_object.
```



*) https://github.com/numba/llvmlite

Модель класса Python (пример)

```
:- object(class, specializes(code_block),
   imports([named])). % Категория поименованных сущностей
  :- public([classlist/1, methods/1, attributes/1]).
  % . . . . . .
  renderitem(Item, Result):-    % Стандартное
    ^^renderitem(Item, Result). % преобразование
  render(Result):-       % Генератор кода, реализо-
    ^^render(Name),      % ванный в категории
    ( ::item(classlist(List)) ->
     % . . . . . .
        [Name]) ),
    ( ::item(attributes(Attributes))->
     % . . . . . .
        [DefAttrList]),
      Attributes::items(InstanceAttrs),
      findall(S, ( % Инициализация атрибутов
         % . . . . . .
         ), AttrAssigns),
        root::unindent,
        AttrList=[ConstructorDef|AttrAssigns];
         % . . . . . .
        AttrList=[ConstructorDef, Pass] ),
    ( ::item(methods(Methods))-> % Если есть ...
      Methods::render(MethodList);
      MethodList=[] ),
    lists::append(AttrList,MethodList,StringList),
    root::unindent, Result=[Signature|StringList].
:- end_object.
```



Категории Logtalk

Категория поименованных сущностей

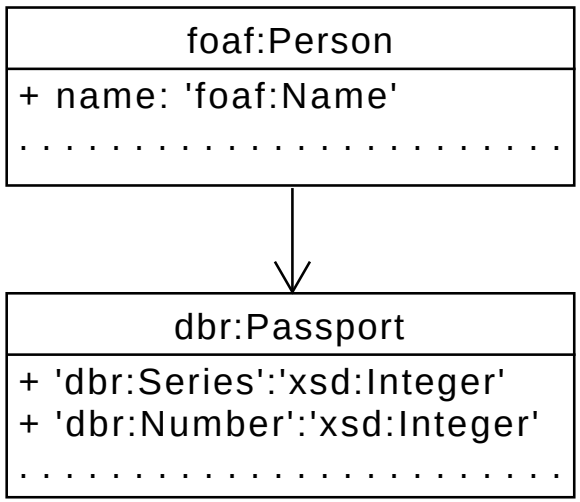
```
:- category(named).
  :- public([name/1, render/1]).
  :- protected([renderitem/2]).
  name(Name):- ::prepend(name(Name)).
  renderitem(name(Name), String):-!, atom_string(Name, String).
  render(String):-  % Порождение кода
    ::item(name(Name)), ::renderitem(name(Name), String).
:-end_category.
```

Категория поименованных типизированных сущностей

```
:- category(namedtyped, extends(named)).
  :- public([type/1,render/2, separator_option/2,list_separator/1]).
  :- protected([renderitem/2]).
  type(Type):- ::append(type(Type)).
  renderitem(Item, String):- ^^renderitem(Item, String),!.
  renderitem(type(Type),String):-!, ::list_separator(Separator),
    writef::swritef(String, '%w%w', [Separator, Type]).
  render(Middle, String):- ^^render(SName),
    ( ::item(type(Type)) ->
      ::renderitem(type(Type), SType),
      string_concat(SName, Middle, _1),
      string_concat(_1, SType, String) ;
      SName = String ).
  render(String):-  ::render("", String).
  list_separator(Separator):-
      ::separator_option(Name, Default),!, % Глобальные настройки
      root::option(Name, Separator, Default).
:- end_category.
```

Доступ к внешним графам знаний, LOD

```
:- category(sparql).
  :- public(query/2).
  query(Pattern,Parameters,Row):-
    prepare(Pattern,Parameters,Query),
    server(Host,Port,Path),
    sparql_query(Query, Row,
      [host(Host),port(Port),path(Path)]).
   :- protected(server/3).  % реализовать
                            % при наследовании.
  :- protected(prepare/3). % подготовка запроса
  % . . . . . .           % в виде строки.
  :- end_category.

  :- object(dbpedia, extends(sparql)).
  :- protected(server/3).
  server('dbpedia.org',80,'/sparql').
  :- public(entity_name/2).
  entity_name(Entity,Language,Name):-
    query('select ?name where { '
      ' %w rdfs:label ?name. '
      'FILTER langMatches( lang(?label),'
      ' "%w" )}', [Entity, Language],
      row(Name)).
:- end_object.

% ?- dbpedia::entity_name(dbr:'Passport', 'ru', Name).
```



Обсуждение

Интересные замечания по поводу использования Logtalk:

- Logtalk и RDF гибки и достаточно универсальны для удобной реализации задач АКИД, выразимые в терминах анализа свойств и преобразования элементов
- Наиболее полезным средством Prolog и Logtalk – предикатная и объектная инкапсуляция: сложные рассуждения можно скрыть за интерфейсами
- Необходимо дальнейшее изучение свойств языка (2009).
- В настоящий момент ведется НИРОКР в области распознавания структуры текста в PDF-документе, в частности в учебных программ вузов и их RDFa-разметки.
- Для размеченного инструмента разрабатывается веб-сервис редактирования компонент текста с учетом их семантики (текст, перечисление, иерархический список, ссылка на литературу и т.п.)

Заключение

Результаты, полученные к настоящему времени:

- Разработаны методики представления алгоритмов распознавания текстовых структур.
- Разработана методика представления сценариев распознавания и трансформации в виде иерархии логических объектов.
- Накоплен некоторый опыт реализации конкретных задач в области порождения кода информационных систем и анализа слабоструктурированных документов.
- Существенных технических проблем пока не выявлено.

Дальнейшее развитие проекта:

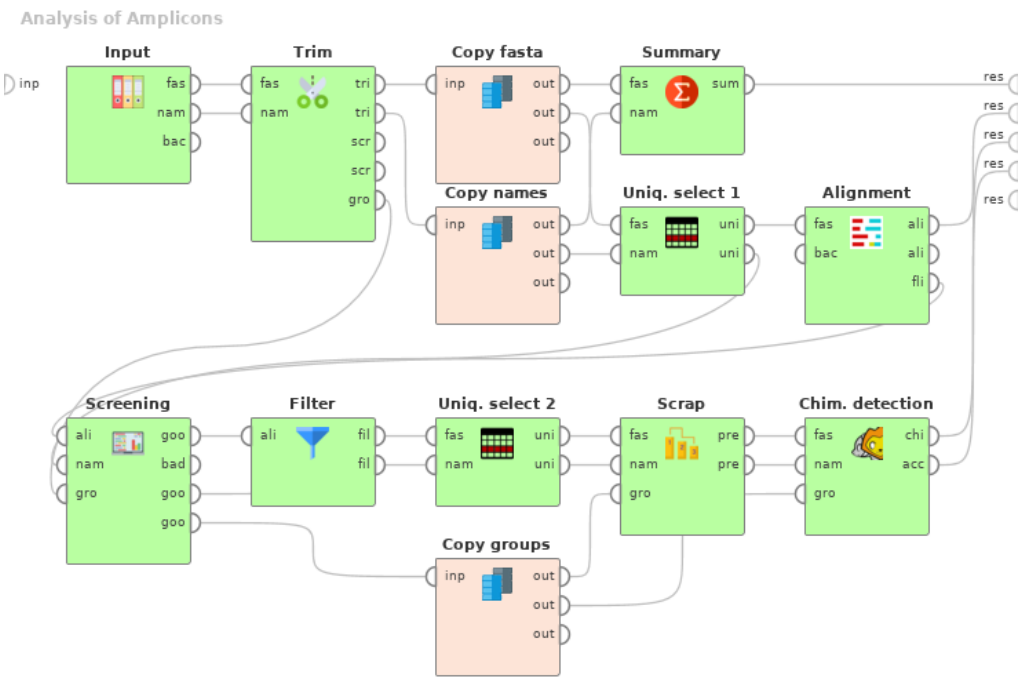
- Разработка методик программирования с преимущественным использованием статических объектов.
- Реализация библиотек и для конкретных классов задач.

Исходные коды находятся на Github в разделах:

- https://github.com/isu-enterprise,
- https://github.com/eugeneai.

Спасибо за уделенное внимание!

Приложение: Представление NGS в виде диаграммы DataFlow



Термин	Определение
NGS	Секвенирование нового поколения
Amplicon	Часть ДНК или РНК, скопированная много раз
Mothur	Пакет для исследований в NGS
Rapidminer studio	Визуальный редактор Dataflow-диаграмм

Зеленые блоки – модули Mothur, другие – модули Rapidminer studio.

Использование MDA позволяет актуализировать структуру ПП Mothur (в н.в. 144 модуля).