Erlang и практика использования трансформируемого кода

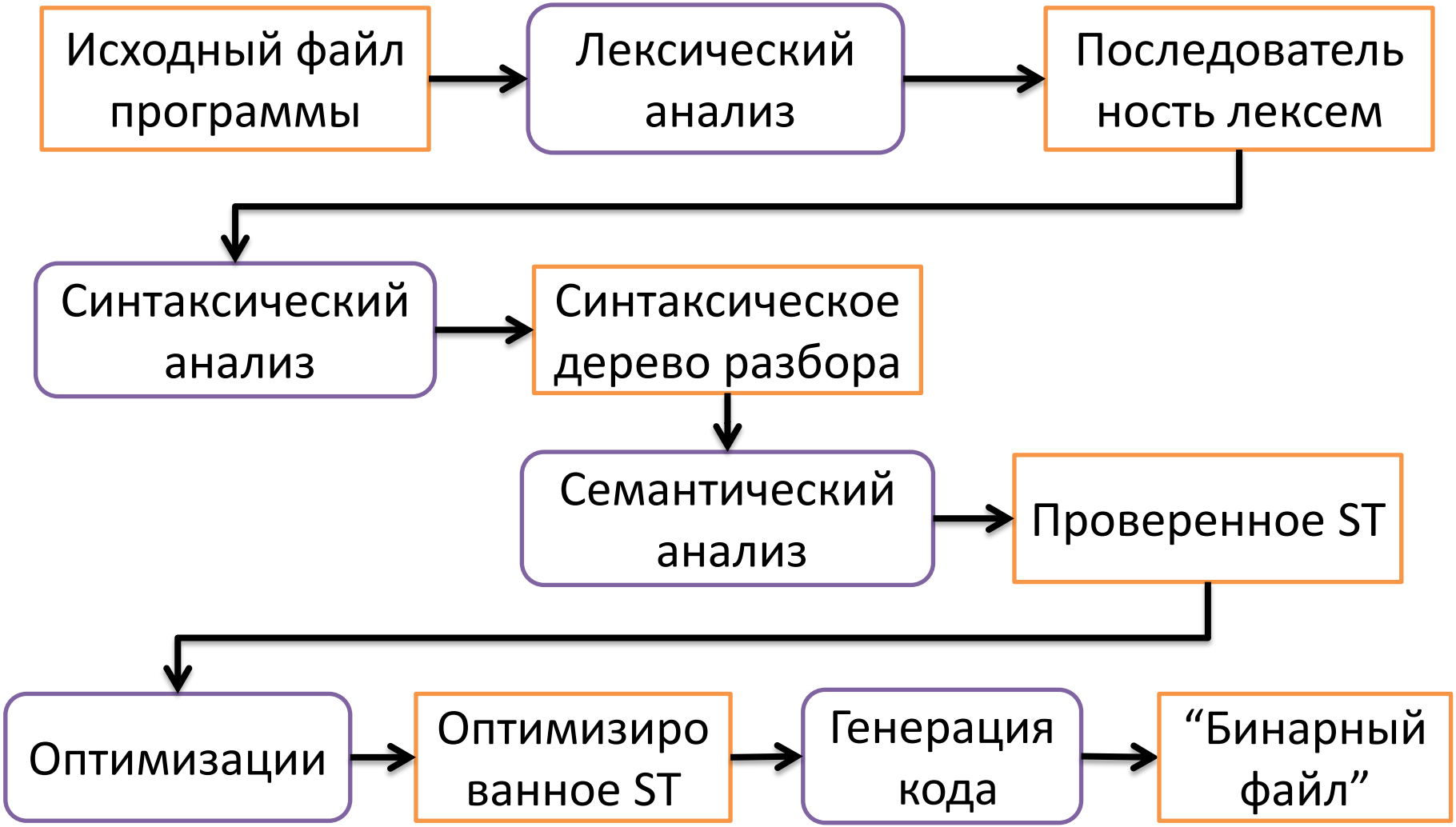
Рябков Антон
инженер-программист
anton.ryabkov@gmail.com

ELTEX
Комплексные решения для построения сетей

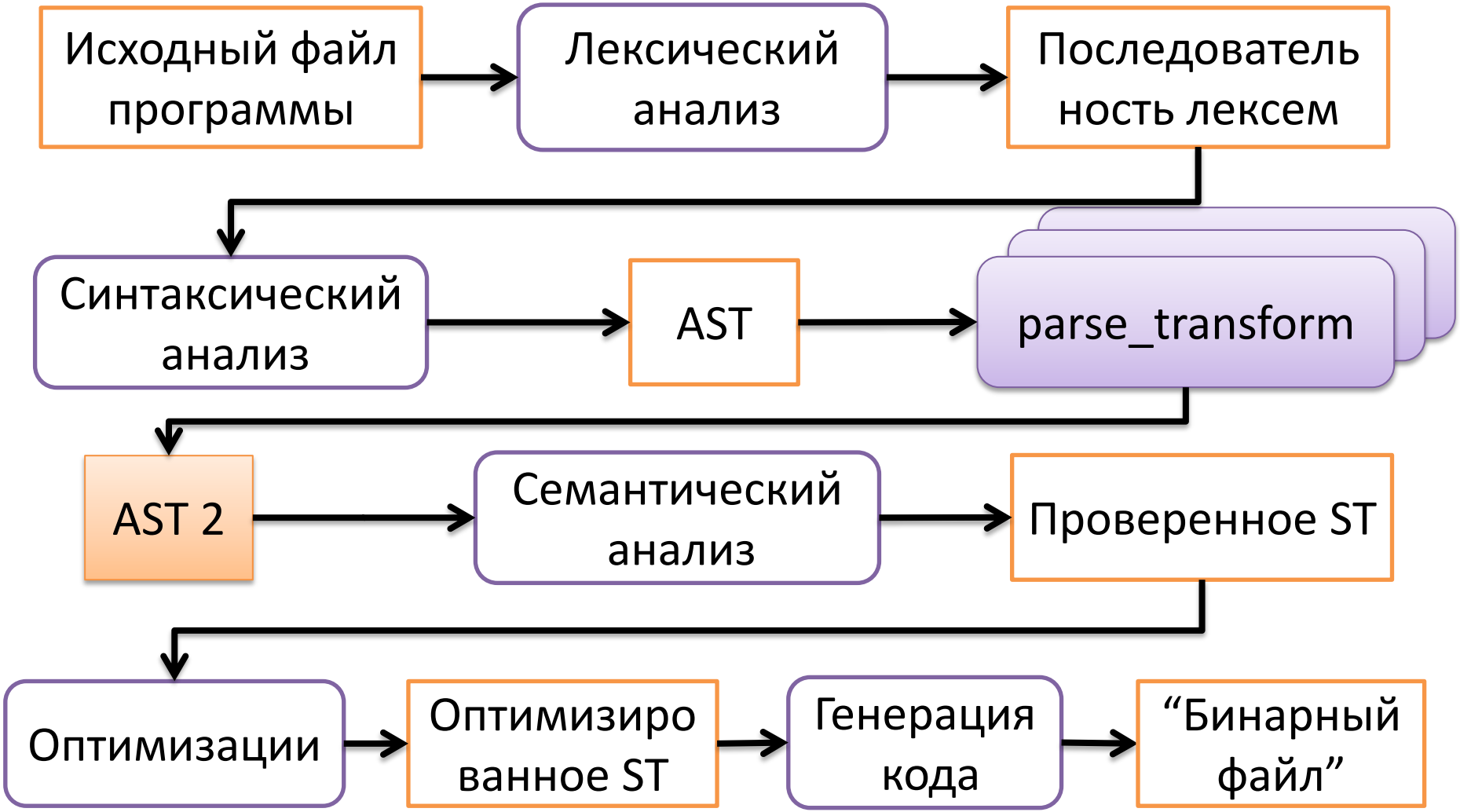
Сентябрь 2013

www.eltex.nsk.ru

Компиляция программы



Компиляция в Erlang



Code & AST

```
1. -module(hello_world).
2. -export([print/1]).

3. print(Text) -> io:format("Hello ~p~n", [Text]).
```

```
1. [{attribute,1,file,{"hello_world.erl",1}},
2.  {attribute,1,module,hello_world},
3.  {attribute,3,export,[{print,1}]},
4.  {function,5,print,1,
5.   [{clause,5, [{var,5,'Text'}], [],
6.    [{call,5, {remote,5,{atom,5,io},{atom,5,format}},
7.              [{string,5,"Hello ~p~n"},
8.               {cons,5,{var,5,'Text'},{nil,5}}]}]}]},
9.  {eof,6}]
```

Пример использования

```
1. -module(hello).
2. -compile({parse_transform, first_pt}).
3. -export([print/1]).

4. print(Text) ->
5.     Attr = ?MODULE:module_info(attributes),
6.     case proplists:get_value(compile_date, Attr) of
7.         [{{Y,M,D}, {DD,MM,SS}}] ->
8.             io:format("~p/~p/~p ~p:~p:~p Hello ~p~n",
9.                       [Y, M, D, DD, MM, SS, Text]);
10.         _ ->
11.             io:format("Hello ~p~n", [Text])
12.     end.
```

```
> hello:print(everybody).
 2013/10/4 22:17:32 Hello everybody
```

Пример parse_transform

```
1. -module(first_pt).

2. -export([parse_transform/2]).

3. parse_transform(AST, _Options) ->
4.     add_attribute(AST, []).

5. add_attribute([], Res) ->
6.     erlang:error(no_module_attribute);
7. add_attribute([{attribute, Line, module, _} = M|AST],
8.                Res) ->
9.     lists:reverse([{attribute, Line, compile_date,
10.     calendar:local_time()}, M|Res]) ++ AST;
11.add_attribute([M|AST], Res) ->
12.    add_attribute(AST, [M|Res]).
```

Logger

```
1. -module(test_logger).
2. -include_lib("chronica/include/chronica.hrl").

3. print(Text) ->
4.     log:debug("Start ~p", [Test]),
5.     io:format("~p~n", [Test]),
6.     log:debug("Stop ~p", [Test]).

7. failed_print(Text) ->
8.     log:debug("Start ~p ~p", [Test, 1 / 0]),
9.     io:format("~p~n", [Test]),
10.    log:debug("Stop ~p ~p", [Test , 1 / 0]).
```

string:match

```
1. -module(test_string_match).
2. -include_lib("pt_scripts/include/pt_str_parser.hrl").

3. parse(Input) ->
4.     string:match(Input, [Type=string], [D1=delim],
5.                  [CCName=string],[D2=delim],
6.                  [Number=string],[D3=delim],'end'),
7.     {Type, CCName, Number}.

8. raw_parse(Input) ->
       match,[Type,D1,CCName,D2,Number,D3]}= re:run(Input,
1. "^\\s*((?<1>((\"([^\"]*)\")|('([^']*)')|(\\S+))))?(("
2. "?<2>\\s+))?((?<3>((\"([^\"]*)\")|('([^']*)')|(\\S+)"
3. ")))?((?<4>\\s+))?((?<5>((\"([^\"]*)\")|('([^']*)')|"
4. "(\\S+))))?((?<6>\\s+))?$",
   [{capture,['1','2','3','4','5','6'],list}]),
5.    {Type, CCName, Number}.
```

CP-options

```
1. -module(test_options).
2. -include_lib("cp_options/include/pt_cp_options.hrl").

3. host() -> "localhost".

4. port() -> 8080.
```

```
1. -module(test).

2. do_test() ->
3.   io:format("~s:~p~n", [test_options:host(),
4.                         test_options:port()]),
5.   ecss_cm:update_property(test_options, host,"127.0.0.1"),
6.   ecss_cm:update_property(test_options, port,8888),
7.   io:format("~s:~p~n", [test_options:host(),
8.                         test_options:port()]).
```

Сторонние использования

- **Literal XML** – трансформирует текст xml в объект xml;
- **mad-cocktail** – «синтаксических сахар» для Erlang;
- **SeqBind** – имитация переменных в Erlang
- **Atomizer** – валидация списка используемых атомов

Итог

Использование parse_transform позволит сделать ваш код более читабельным, компактным, динамичным, повысить его производительность. Платой за это является повышенные требования к разработчикам parse_transform функций.

Спасибо за внимание!

ELTEX
Комплексные решения для построения сетей