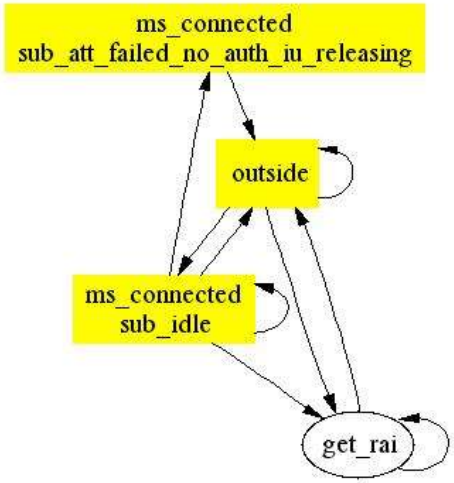
IT-universitetet i Göteborg
CHALMERS | GÖTEBORGS UNIVERSITET

Reverse engineering Erlang software

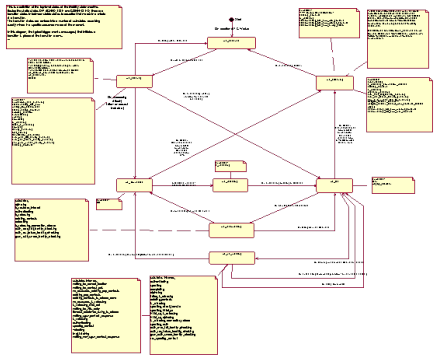


Thomas Arts

IT university in Göteborg

Cecilia Holmqvist

Ericsson AB



IT-universitetet i Göteborg
CHALMERS | GÖTEBORGS UNIVERSITET

Design out of date…

Ericsson has developed nodes for the UMTS and GPRS network in Erlang

Development in three countries over the last years. Now concentrated on Lindholmen (Göteborg).

Designed in UML, implemented in Erlang. ***The code has been changed, not the design.***

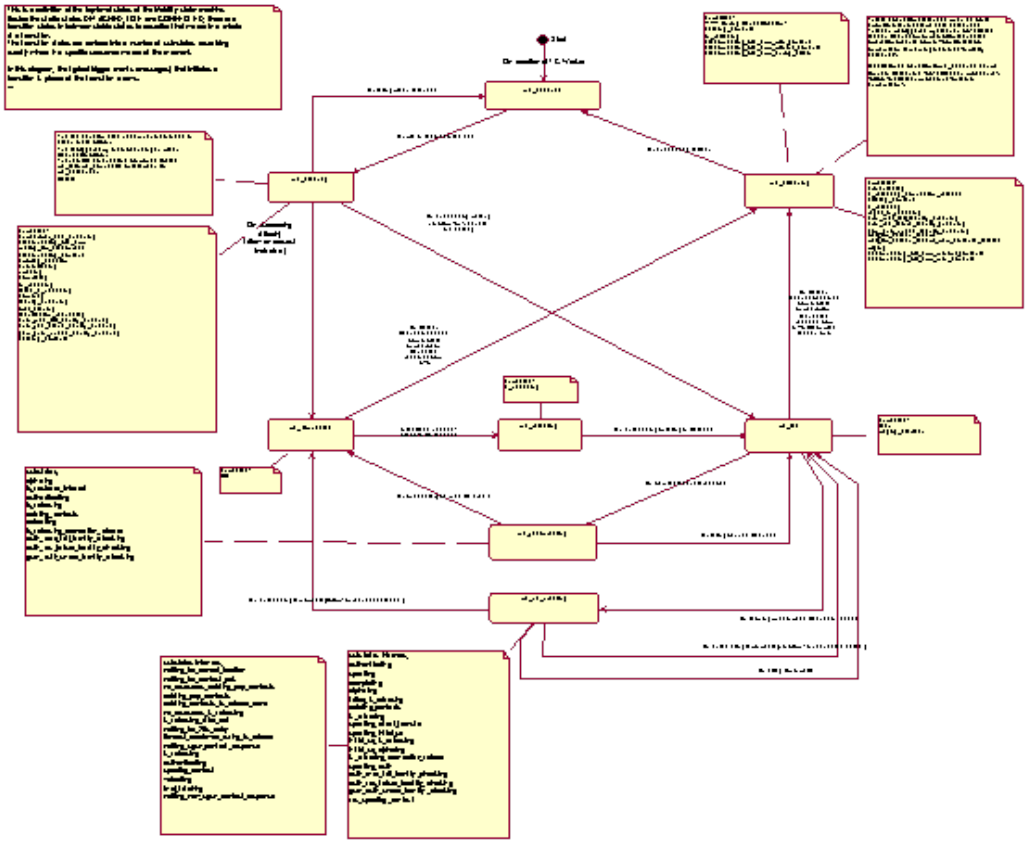
Design out of date…

Can we extract the design from the code?

Useful for

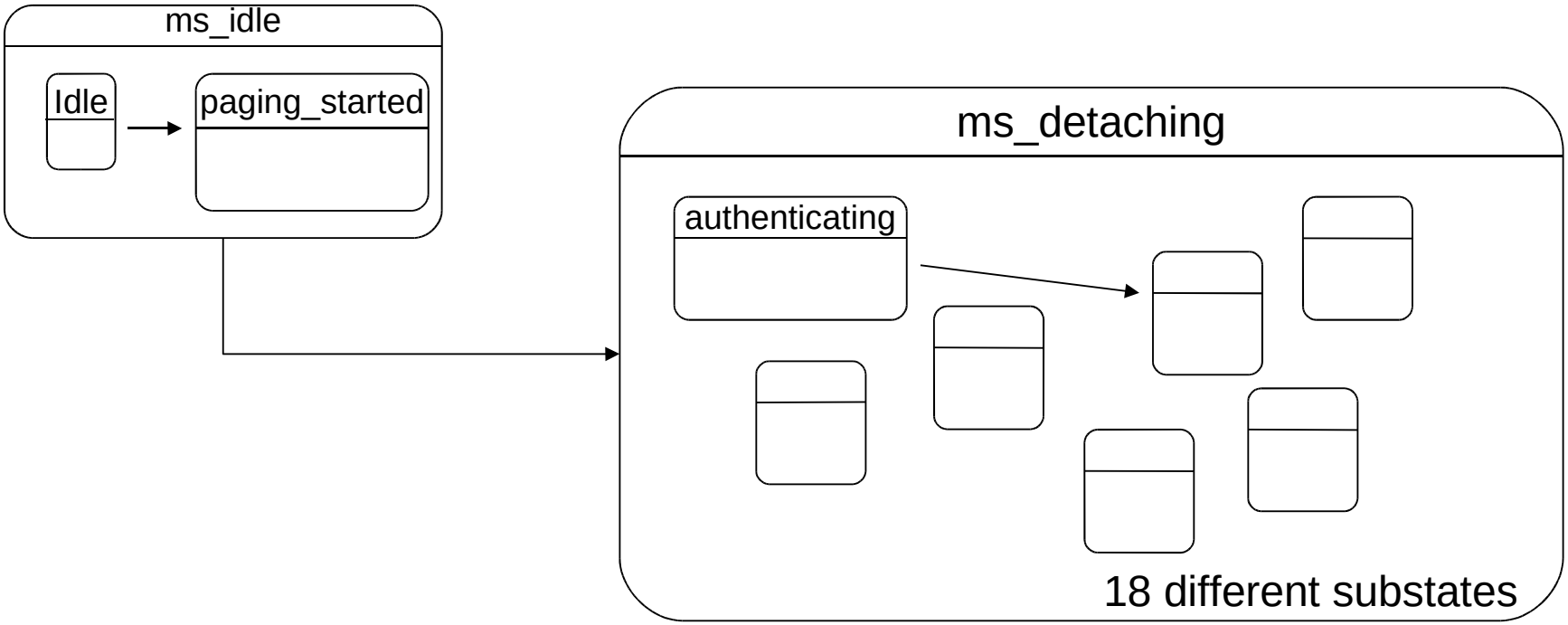
- *better understanding of the system,*
- *re-implementation of the system,*
- *documentation purposes,*
- *differences indicate possible problems*

Mobility Management



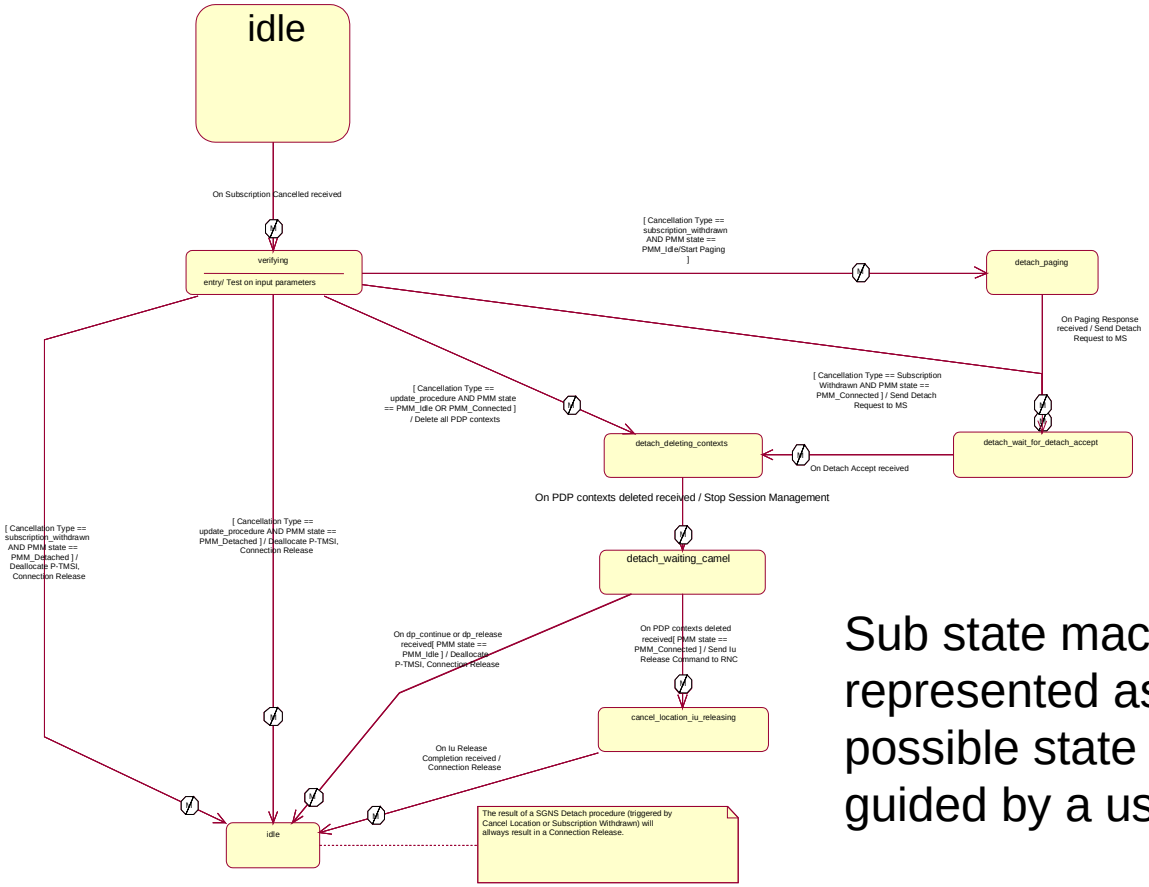
Mobility Management

Hierarchical State Machines



Mobility Management

ms_detach

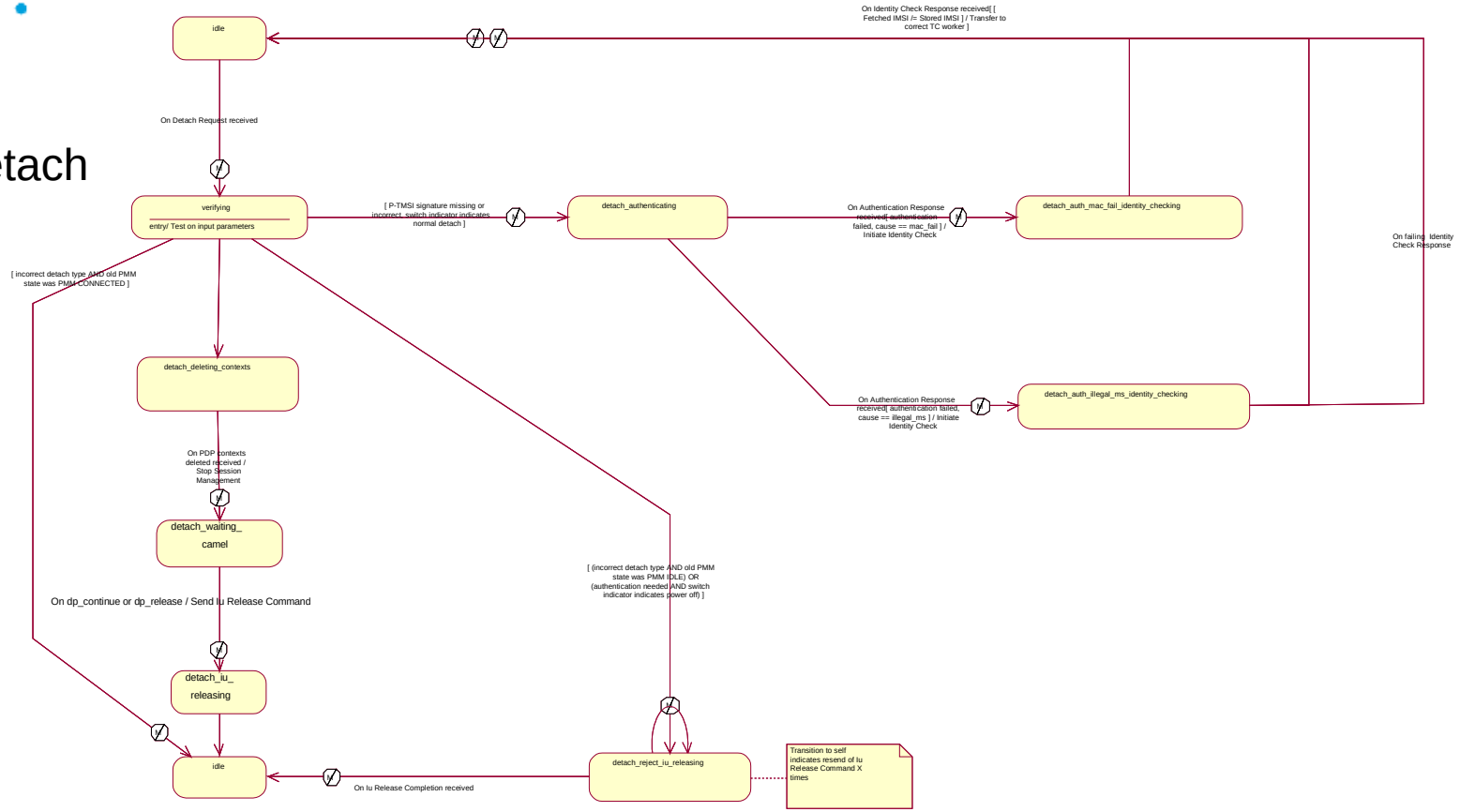


Sub state machines represented as a set of possible state transitions guided by a use-case

Mobility Management

IT-universitetet i Göteborg
CHALMERS | GÖTEBORGS UNIVERSITET

ms_detach



Extract the design

Can we generate similar pictures, i.e., generate a state model for a certain use-case ?

Source code analysis has been studied before, e.g. Nyström 2001 and Mohagheghi et al 2003

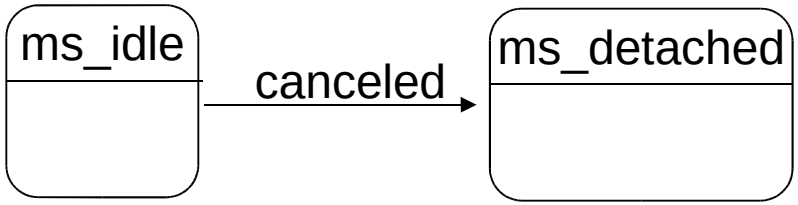
Runtime analysis has been studied in earlier work Arts and Fredlund 2002

Source code analysis

Example:

Given the source code of a generic finite state machine, one can extract a picture of the finite state machine

```
ms_idle({canceled,…},Data) ->
    ……
    {next_state,ms_detached,…};
```



Source code analysis

Difficult to use here, because state machine implemented by many different modules, in a very non-standard way.

Events are implemented as function calls, e.g. `detach_request/6`

Cascade of function calls in several modules follow such an event

One of the function calls in the cascade can be `mmumoc:set_state/2` *which registers the state*

Flow analysis almost impossible

Source code analysis

The diagrams one wants to obtain reflect a flow through the state machine when dealing with a certain use-case

With pure symbolic analysis one obtains the complete state machine instead of one specific for a scenario

Runtime analysis

Idea: run the software on a special test case obtained from the use-case

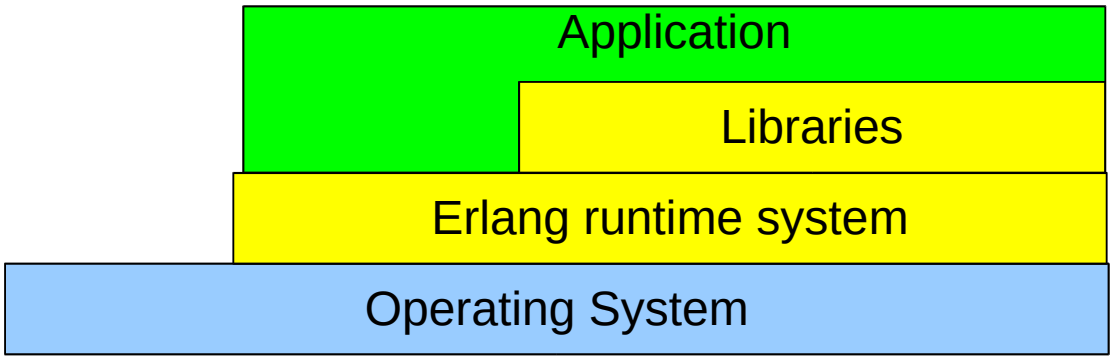
Register all events and state changes that occur

Test cases are already developed

Erlang has an advanced tracing possibility

Runtime analysis

The funtions `erlang_trace/2` and `erlang_trace_pattern/2` can be used to send a message to a collection process every time a certain function is called.



Runtime analysis

Standard trace tool used to trace calls to MMU modules while executing test cases for a certain use-case

```
TRACE 2004-02-16 14:41:01,915166 <3730.1335.0>,call,mmumoc_c:modify_node_prop
Caller.........:noncos_c_nodeprop:modify_node_prop/3
Node...........:'e_Erlang__Global_pm1_1_2_1@octavo151.0'
Argument List..:[void,"SelectiveAuthenticationFrequency","10"]

TRACE 2004-02-16 14:41:01,915215 <3730.1335.0>,call,mmumoc_c:check_values
Caller.........:mmumoc_c:modify_node_prop/3
Node...........:'e_Erlang__Global_pm1_1_2_1@octavo151.0'
Argument List..:["SelectiveAuthenticationFrequency",10]

TRACE 2004-02-16 14:41:01,915252 <3730.1335.0>,return_from,mmumoc_c:check_values/2
Node...........:'e_Erlang__Global_pm1_1_2_1@octavo151.0'
Return Value...:{valid,10}

TRACE 2004-02-16 14:41:01,917425 <3730.1335.0>,return_from,mmumoc_c:modify_node_prop/3
Node...........:'e_Erlang__Global_pm1_1_2_1@octavo151.0'
Return Value...:{ok,"void"}
```

IT-universitetet i Göteborg
CHALMERS | GÖTEBORGS UNIVERSITET

Trace data

Average 15,000 entires, file size 9MB

Erlang log-file format (binary)

Analyzing with emacs rather unpleasant

We are interested in the calls to `mmumoc:set_state/2`

IT-universitetet i Göteborg
CHALMERS | GÖTEBORGS UNIVERSITET

Filter Trace data

Prepared for 1GB or larger log files, filter out the functions in which state is set.

```
read(FileName,Predicate) ->
    {ok,FileDescr} =
        file:open(FileName,[read, raw, binary]),
    Terms =
        unpack(FileDescr,Predicate),
    file:close(FileDescr),
    Terms.
```

Filter Trace data

```
unpack(FileDescr,Predicate) ->
    case file:read(FileDescr,5) of
        {ok,<<B1,B2,B3,Size:16>>} ->
          {ok,BTerm} =
               file:read(FileDescr,Size),
          Term =
               binary_to_term(BTerm),
          case Predicate(Term) of
              true ->
                 [Term|unpack(FileDescr,Filter)];
              false ->
                 unpack(FileDescr,Filter)
          end;
        eof ->
          []
    end.
```

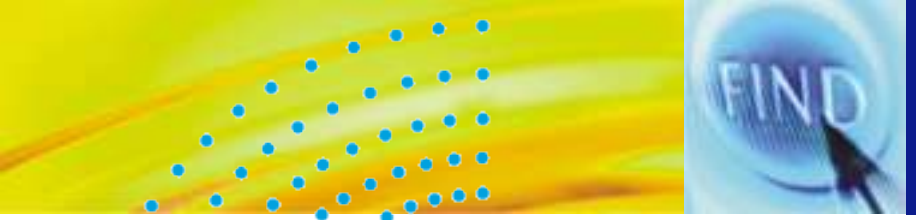
Filter Trace data

Predicate example:

```
state_mmu() ->
    fun({trace_ts,Pid,call,{mmumoc,set_state,[S,SS]},Caller,TS}) ->
        true;
       (_) ->
        false
    end.
```

Combining several predicates (or, and, not):

```
pred_or(F1,F2) ->
    fun(T) ->
        F1(T) or F2(T)
    end.
```



IT-universitetet i Göteborg
CHALMERS | GÖTEBORGS UNIVERSITET

Abstract trace data

Similar to filtering, we define abstraction functions that are applied to all entries in the trace.

For example:

```
{trace_ts,Pid,call,{mmumoc,set_state,
   [S,SS]},Caller,TS}
```

can be abstracted to

```
{state,[S,SS]}
```

Abstract trace data

Abstracting trace entries allows to map different entries to the same constant.

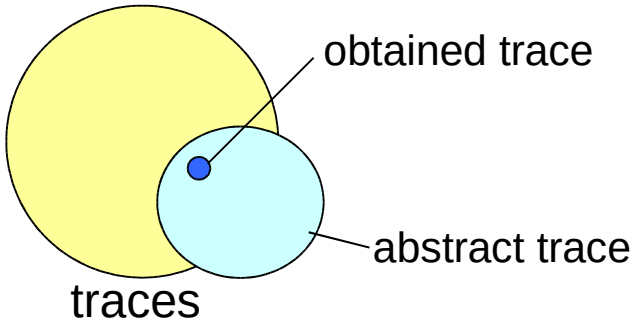
For example:

All states but `ms_connected` *are abstracted to* `outside`

This selects one state with all substates from a trace

Abstract trace data

More than one trace



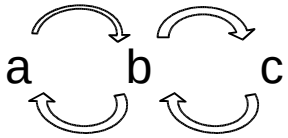
Abstract trace data

```
a(event,X) ->
 {next_state,b,X}.

b(event,X) ->
  case X rem 2 of
       0 -> {next_state,a,X+1};
       1 -> {next_state,c,X+1}
  end.

c(event,X) ->
 {next_state,b,X}.
```

=

Trace:

a(event,1)
b(event,1)
c(event,2)
b(event,2)
a(event,3)
b(event,3)
c(event,4)
b(event,4)
a(event,5)
b(event,5)
....



Structuring trace data

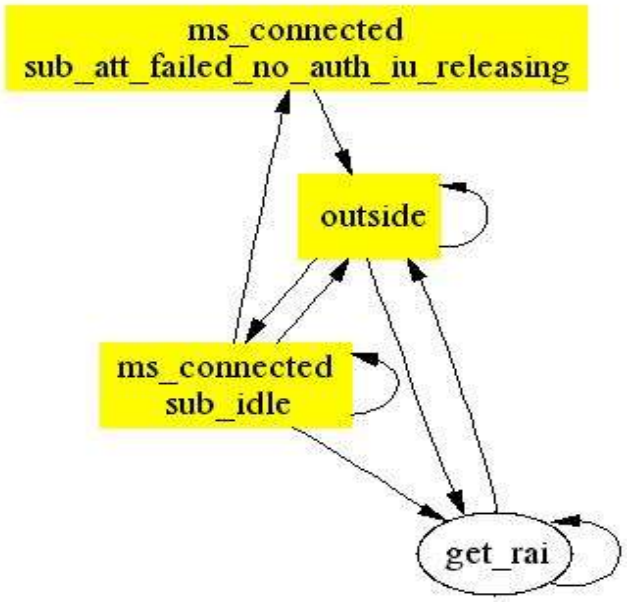
After filtering and abstraction we can obtain a trace like:

[{state,outside},...,{state,outside},
{state,[ms_connected,sub_idle]},
{event,get_rai},
{state,outside},...,{state,outside},
{state,[ms_connected,sub_idle]},
{event,get_rai},
{state,outside},...,{state,outside},
{state,[ms_connected,sub_idle]},
{state,[ms_connected,sub_idle]},
{state,outside},...,{state,outside},
{event,get_rai},
...]

Structuring trace data

Represented as a graph, this gives a clear picture of possible scenarios

```
[{state,outside},...,{state,outside},
 {state,[ms_connected,sub_idle]},
 {event,get_rai},
 {state,outside},...,{state,outside},
 {state,[ms_connected,sub_idle]},
 {event,get_rai},
 {state,outside},...,{state,outside},
 {state,[ms_connected,sub_idle]},
 {state,[ms_connected,sub_idle]},
 {state,outside},...,{state,outside},
 {event,get_rai},
 ...]
```



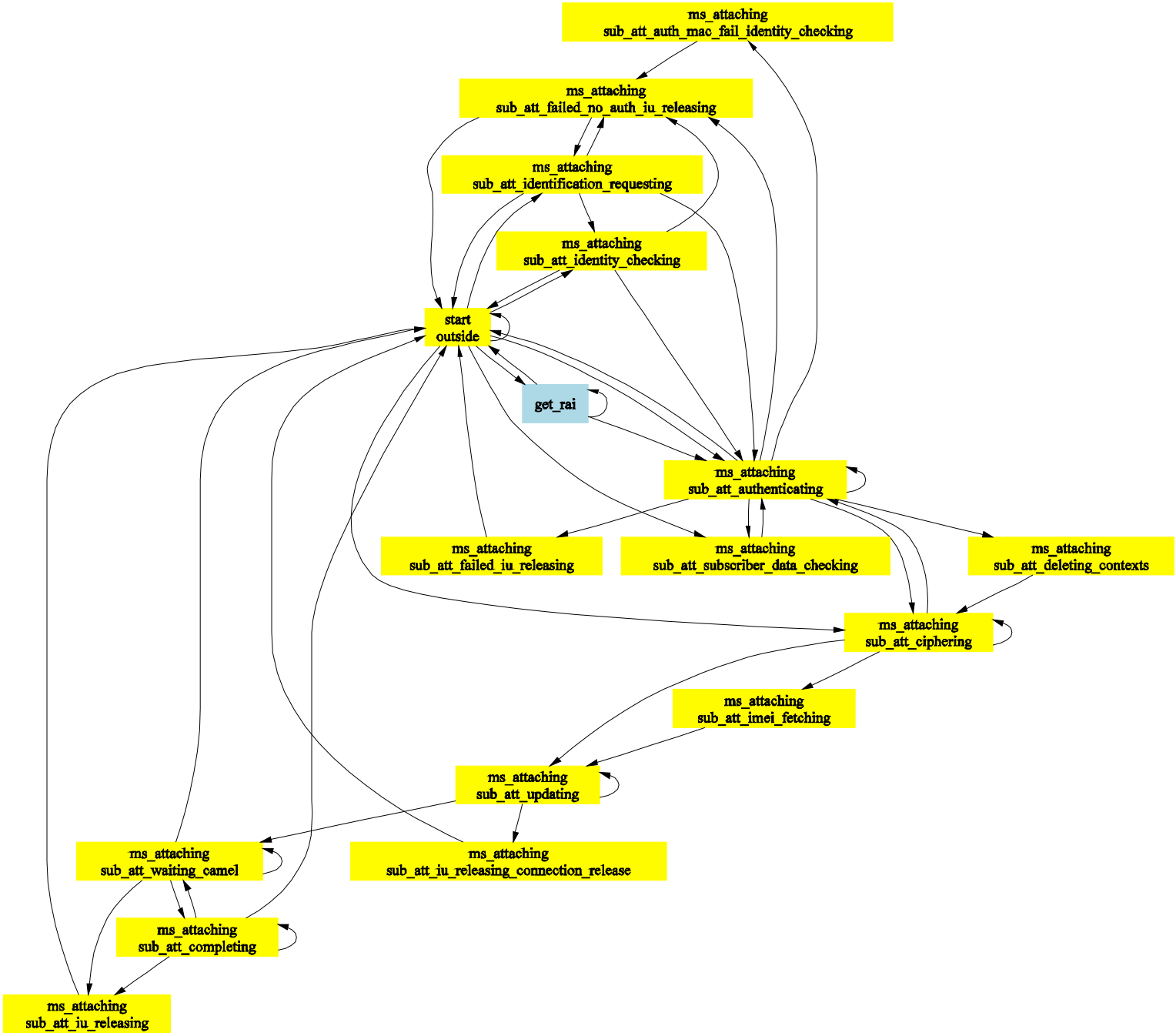
Comparison

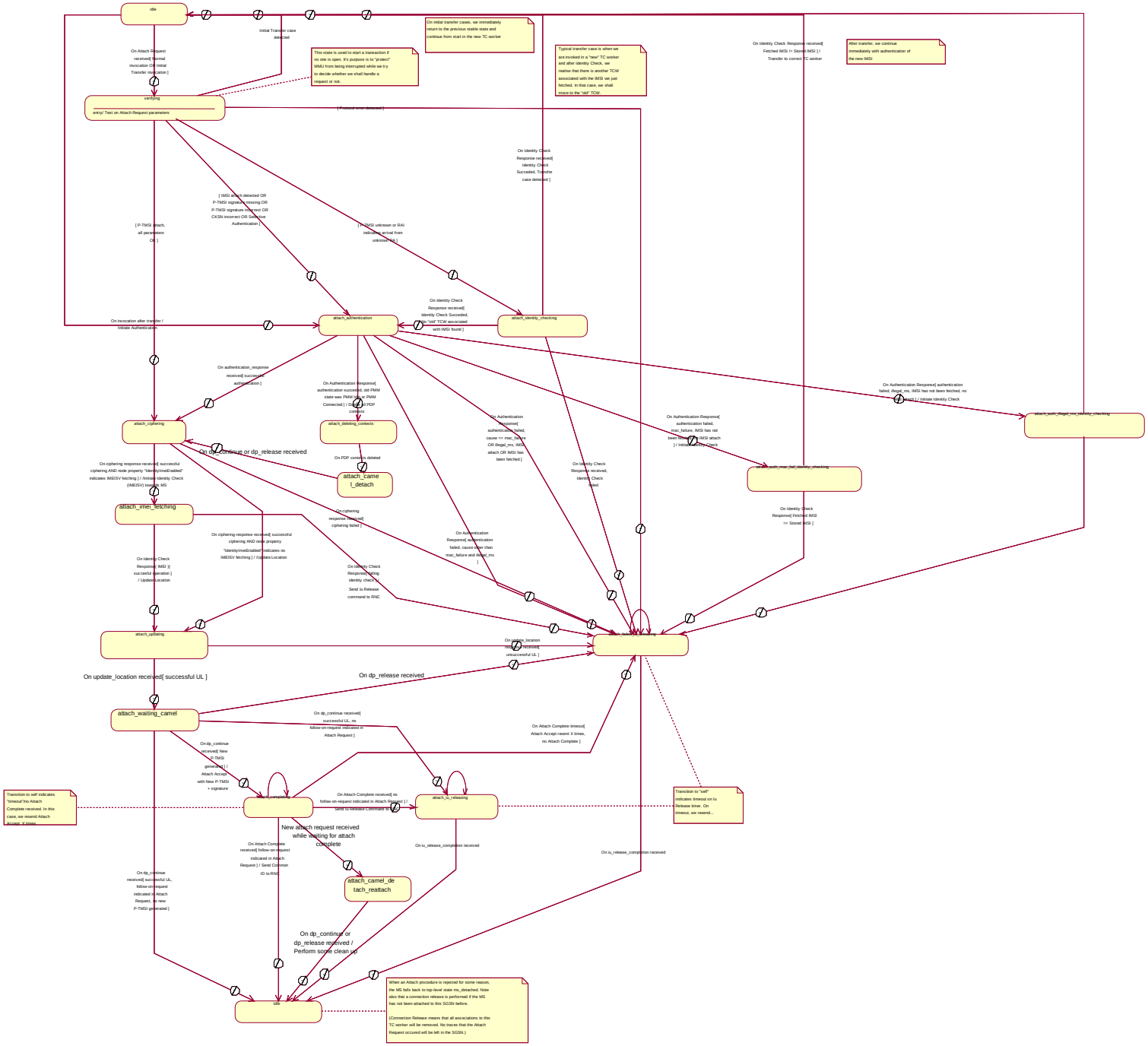
We found (after manual mapping of names):

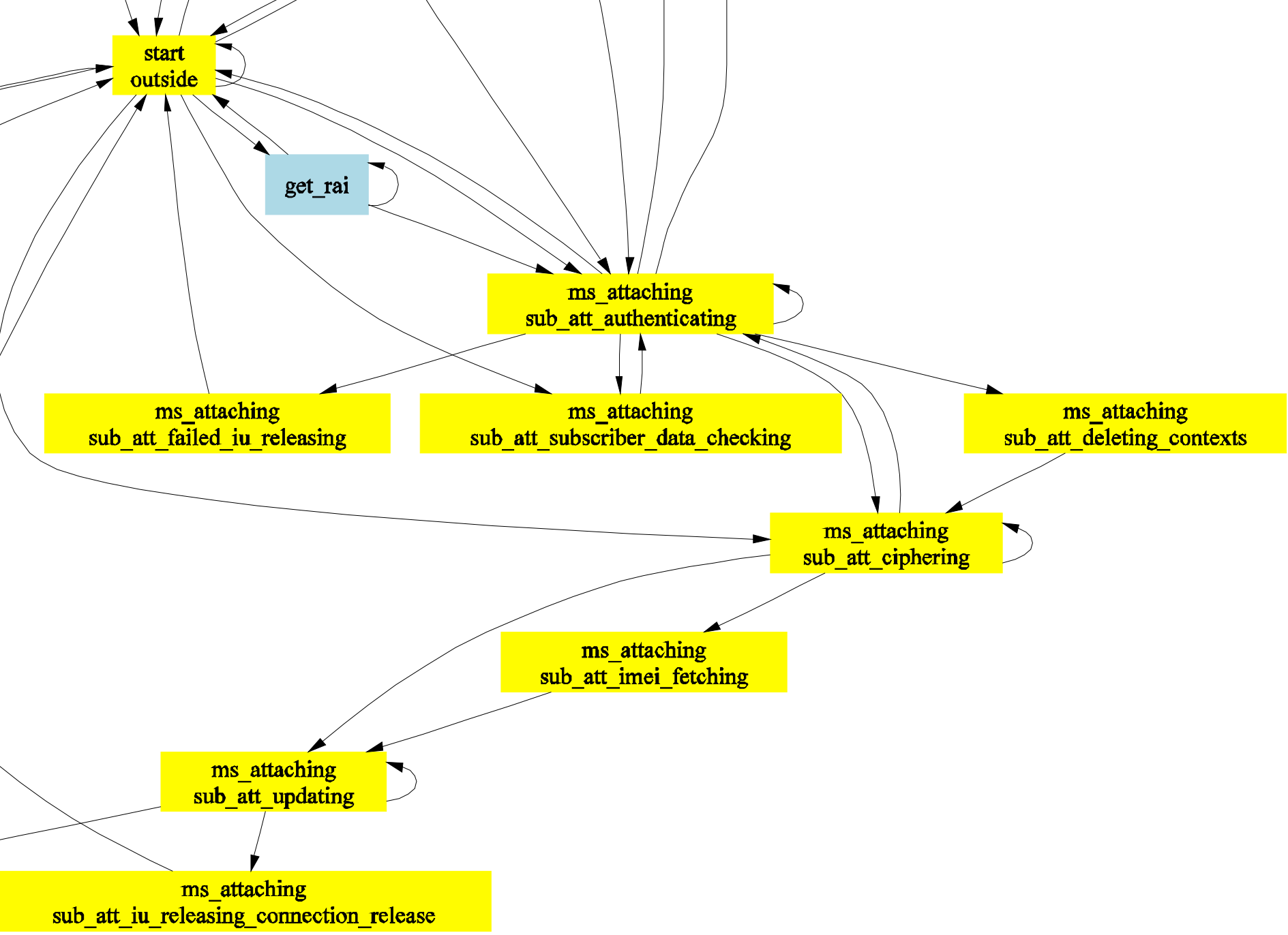
States in obtained diagram that are not in original

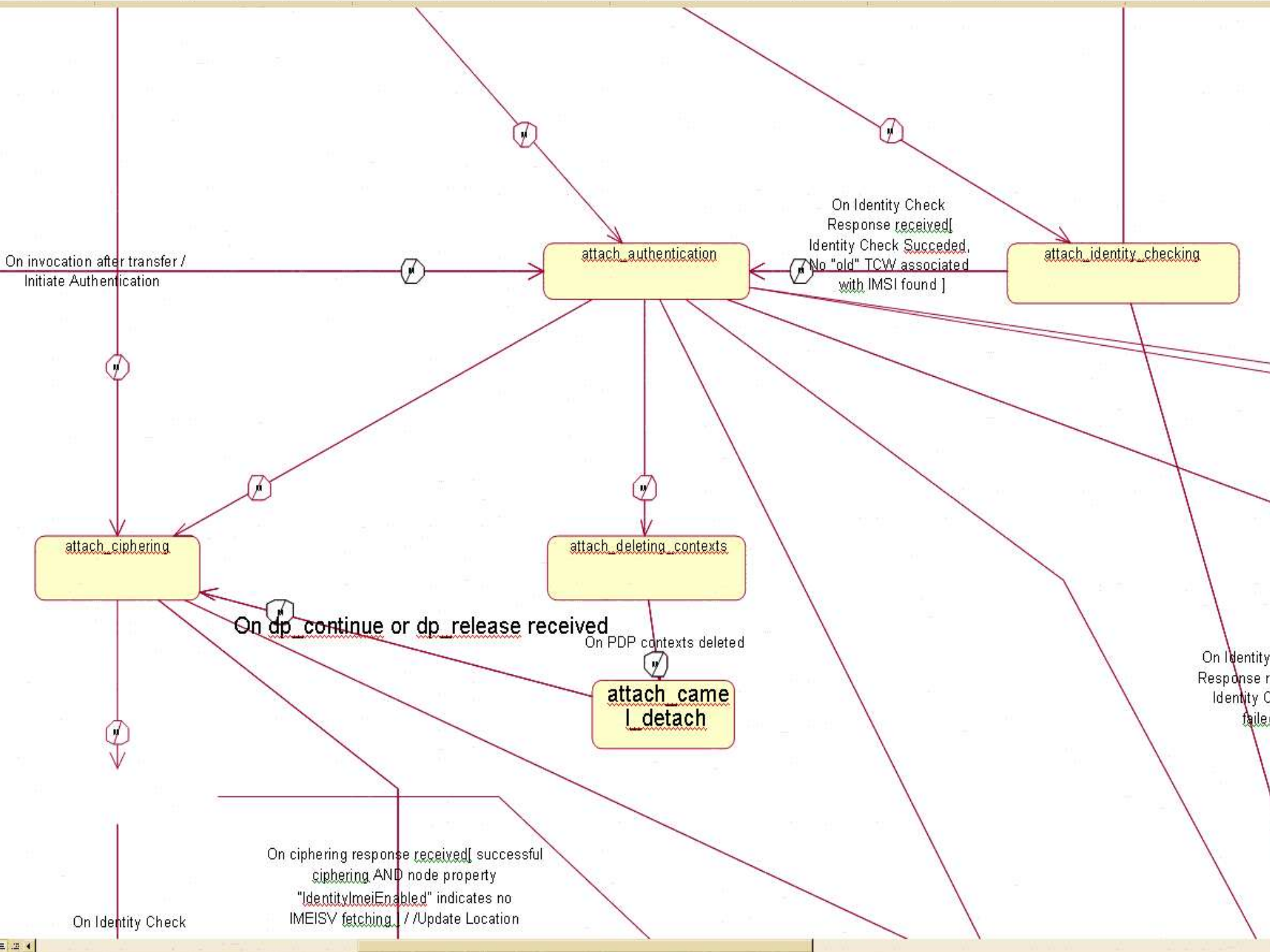
State transitions in obtained diagram that are not in original

States and transitions in original that do not show in obtained diagram


ms_attaching
sub_att_auth_mac_fail_identity_checking
ms_attaching
sub_att_failed_no_auth_iu_releasing
ms_attaching
sub_att_identification_requesting
ms_attaching
sub_att_identity_checking
start
outside
get_rai
ms_attaching
sub_att_authenticating
ms_attaching
sub_att_failed_iu_releasing
ms_attaching
sub_att_subscriber_data_checking
ms_attaching
sub_att_deleting_contexts
ms_attaching
sub_att_ciphering
ms_attaching
sub_att_imei_fetching
ms_attaching
sub_att_updating
ms_attaching
sub_att_waiting_camel
ms_attaching
sub_att_iu_releasing_connection_release
ms_attaching
sub_att_completing
ms_attaching
sub_att_iu_releasing


idle
Initial Transfer case detected
On initial transfer cases, we immediately return to the previous stable state and continue from start in the new TC worker
This state is used to start a transaction if no one is open. It's purpose is to "protect" MMU from being interrupted while we try to decide whether we shall handle a request or not.
Typical transfer case is when we are involved in a "new" TC worker and after Identity Check, we realize that there is another TCW associated with the IMSI we just fetched. In that case, we shall move to the "old" TCW.
On Identity Check Response received[Fetched IMSI != Stored IMSI] / Transfer to correct TC worker
After transfer, we continue immediately with authentication of the new IMSI
On Attach Request received[Normal invocation OR Initial Transfer invocation]
verifying
entry/ Test on Attach Request parameters
[Protocol error detected]
On Identity Check Response received[Identity Check Succeded, Transfer case detected]
[IMSI attach detected OR P-TMSI signature missing OR P-TMSI signature incorrect OR CKSN incorrect OR Selective Authentication]
[P-TMSI unknown or RAI indicates MS arrival from unknown RA]
[P-TMSI attach, all parameters OK]
On Identity Check Response received[Identity Check Succeded. Found "old" TCW associated with IMSI found]
attach_identity_checking
On invocation after transfer / Initiate Authentication
attach_authentication
On authentication_response received[successful authentication]
On Authentication Response[authentication succeeded, old PMM state was PMM-IDLE or PMM Connected] / Detach all PDP contexts
On Authentication Response[authentication failed, cause == mac_failure OR illegal_ms, IMSI attach OR IMSI has been fetched]
On Authentication Response[authentication failed, illegal_ms, IMSI has not been fetched, no attach] / Initiate Identity Check
attach_auth_illegal_ms_identity_checking
On Authentication Response[authentication failed, mac_failure, IMSI has not been fetched, no IMSI attach] / Initiate Identity Check
attach_auth_mac_fail_identity_checking
attach_ciphering
attach_deleting_contexts
On dp_continue or dp_release received
On PDP contexts deleted
attach_camel_detach
On ciphering response received[successful ciphering AND node property "IdentityImeiEnabled" indicates IMEISV fetching] / Initiate Identity Check (IMEISV) to MS
attach_imei_fetching
On Identity Check Response[Fetched IMSI == Stored IMSI]
On ciphering response received[successful ciphering AND node property "IdentityImeiEnabled" indicates no IMEISV fetching] / Update Location
On ciphering response received[ciphering failed]
On Authentication Response[authentication failed, cause other than mac_failure and illegal_ms]
On Identity Check Response received, Identity Check failed
On Identity Check Response[IMEI][successful operation] / Update Location
On Identity Check Response[Nothing identity check] / Send Iu-Release command to RNC
attach_updating
attach_releasing_camel
On update_location received[successful UL]
On update_location received[unsuccessful UL]
On dp_release received
attach_waiting_camel
On dp_continue received[successful UL, no follow-on-request indicated in Attach Request]
On Attach Complete timeout[Attach Accept resent X times, no Attach Complete]
On dp_continue received[New P-TMSI generated] / Attach Accept with New P-TMSI + signature
attach_waiting_complete
On Attach Complete received[no follow-on-request indicated in Attach Request] / Send Iu-release Command to RNC
attach_iu_releasing
Transition to self indicates "timeout"ing Attach Complete received. In this case, we resend Attach Accept, X times
Transition to "self" indicates timeout on Iu Release timer. On timeout, we resend...
New attach request received while waiting for attach complete
On Attach Complete received[follow-on-request indicated in Attach Request] / Send Common ID to RNC
On Iu_release_completion received
On dp_continue received[successful UL, follow-on-request indicated in Attach Request, no new P-TMSI generated]
attach_camel_detach_reattach
On dp_continue or dp_release received / Perform some clean up
idle
When an Attach procedure is rejected for some reason, the MS falls back to top-level state ms_detached. Note also that a connection release is performed if the MS has not been attached to this SGSN before.
(Connection Release means that all associations to this TC worker will be removed. No traces that the Attach Request occured will be left in the SGSN.)


start
outside
get_rai
ms_attaching
sub_att_authenticating
ms_attaching
sub_att_failed_iu_releasing
ms_attaching
sub_att_subscriber_data_checking
ms_attaching
sub_att_deleting_contexts
ms_attaching
sub_att_ciphering
ms_attaching
sub_att_imei_fetching
ms_attaching
sub_att_updating
ms_attaching
sub_att_iu_releasing_connection_release


On invocation after transfer /
Initiate Authentication
attach_authentication
On Identity Check
Response received[
Identity Check Succeded,
No "old" TCW associated
with IMSI found]
attach_identity_checking
attach_ciphering
attach_deleting_contexts
On dp_continue or dp_release received
On PDP contexts deleted
attach_came
l_detach
On Identity
Response r
Identity C
faile
On ciphering response received[successful
ciphering AND node property
"IdentityImeiEnabled" indicates no
IMEISV fetching] / /Update Location
On Identity Check

Future work

- *Analyze more blocks*
- *Connect Rational visualization tools to our analysis (show differences in one picture)*
- *Automatically generate test cases such that most states and events in the original are covered in the trace*
- *Generate Hierarchical State Machine model*