1.

Complx API

Known bugs

Known issues

BDD
- BDDs are very innefficient to describes systems with high degree sites or a huge number of internal states.
  The BDU extension should solve this problem.

Quantitative compression
- The quantitative compression could be deeply optimize to avoid an exponential dependences with respect to the size of left hand sides.

Current Version

This API matches version 3.36 (July 9th 2008).

Requirement

Complx can be compiled into any platform.
Therefore, the Makefile is only provided for Unix like OS.

Furthermore, the installation requires:
- OCaml 3.09.2
- TK/lablTK (for the graphical interface)
- graphviz (to handle .dot documents)
- an X terminal
- Simplx (Version 3.272c).

Compilation

To install complx (and also simplx) go into the directory plectix/plx-engine:
  cd plectix/plx-engine

Then, you can either compile the light version (i.e. without the graphical interface (LabelTK is not required)):

```
make
```

Or the full version

```
make full.
```

Binaries are in the directory bin/.
To copy them in /usr/bin :

```
make install.
```

In case, you do not have root priviledge, please use the following instead:

```
make LOCAL_DIR="/home/bin" install_in_local .
```

Usage

In the light version, options are passed in the command line:

```
complx file [options]
```

In the full version, one can either use command line options:

```
complx file [options]
```

or launch the graphical interface.

```
complx
```

In both case, the (updated) list of options can be obtained with the following command:

```
complx --help
```

Graphical interface

The graphical interface is just a user-friendly way to set command line options.

1) Each command line option may be filled in the graphical interface;
   Some options are meta options that triggers a list of options;
2) Command line options are organized by Sections;
3) Some buttons allow:
   - to toggle between a normal and an expert mode (in normal mode some options are hidden);
   - to select another section;
   - to browse your file system to upload a file (from now the behavior is only defined whether only one file is donloaded);
   - to quit;
   - to reset options;
   - to load a config file;
   - to save a config file;
   - to launch the complx app.
4) When launched:
   - the current setting is stored,
   - the command line is computed,

- the command line is output on the standard output,
- the app is launched.

Command-line options

We describe here end-users options.
Some options that are restricted to the usage of developpers and experts are not described here.

General options

--help
Verbose help

-h
Short help

--gui
GUI to select

--(no-)expert
Expert mode (more options (not described in this API))

Actions

The following options set what is computed by complx.
Some computations require others, as a consequence unrequested computation may be performed.

meta-options

--do-all
set all following booleans to true

--reset-all
set all following booleans to false

regular options

--(no-)compute-local-views (default: enabled)
compute reachability analysis

--(no-)enumerate-complexes (default: enabled)
enumerate complexes

--(no-)build-influence-map (default: enabled)
construct influence maps

--(no-)compute-qualitative-compression (default: enabled)
simplify the rules

--(no-)compute-quantitative-compression (default: enabled)
simplify the rules

--(no-)do-low-res-contact-map (default: enabled)
compute the low resolution (aka syntactic) contact map

--(no-)do-high-res-contact-map (default: enabled)
compute the high resolution contact map

--(no-)do-ODE (default: disabled)
compute the ODE system

--(no-)do-refine-to-force-cycles (default: disabled)
refine the system to avoid the formation of polymers.

--(no-)do-compute-dag-refinement-relation (default: enabled)
compute the DAG of refinement relation in order to navigate between refinements of a same rule.

--(no-)do-compute-maximal-refinement-relation (default: enabled)
compute the relation between each rule r and the most abstract rule r' such that r is a refinement of r'.

--(no-)do-marshalling (default: enabled)
dump mashallization

--(no-)do-HTML (default: enabled)
launch HTML desktop

--(no-)do-XML (default: enabled)
dump XML session

Input file

--input-marshalling <name> (default: disabled)
Start computation from this marshallized state

--focus-on <name> (default: disabled)
This option takes a ASCII files that contains some Kappa rules.
The low resolution (syntactic) contact map will only provides information around the agents that are mentioned in these rules.
If the option is not used, the full contact map is computed.

--refine-only-these-rules <name> (default: disabled)
This option takes a ASCII files that contains some Kappa rules.
We only compute the refinement for these rules.

Output

Ouput options are used to redirect the result of computations.
They do not launch computation by themselves (i.e. they are useless if thy require a computation that has not been done).

meta-options

--output-scheme <value>
This meta options generate all following options using the argument as common prefix.
More precisely:
complx egfr.ka --output-scheme egfr
is a short cut for:
complx egfr.ka
--output-ODE-contact egfr_plx_ODE_contact.dot
--output-ODE-mathematica egfr_plx_ODE_system.nb
--output-ODE-matlab egfr_plx_ODE_system.m
--output-ODE-alphabet egfr_plx_ODE_alphabet
--output-ODE-obs egfr_plx_ODE_obs
--output-marshalling egfr_plx.marshalling
--output-influence-map-txt egfr_plx_influence_map.txt
--output-influence-map-dot egfr_plx_influence_map.txt
--output-quantitative-compression egfr_plx_compressed_quantitative.ka
--output-qualitative-compression egfr_plx_compressed_qualitative.ka
--output-low-res-contact-map-dot egfr_plx_low_res_contact.dot
--output-low-res-contact-map-ps egfr_plx_low_res_contact.ps
--output-low-res-contact-map-jpg egfr_plx_low_res_contact.jpg
--output-low-res-contact-map-txt egfr_plx_low_res_contact.txt
--output-high-res-contact-map-dot egfr_plx_high_res_contact.dot
--output-high-res-contact-map-ps egfr_plx_high_res_contact.ps
--output-high-res-contact-map-jpg egfr_plx_high_res_contact.jpg
--output-high-res-contact-map-txt egfr_plx_high_res_contact.txt
--output-intermediate-encoding egfr_plx_ckappa.txt";
--output-gathered-intermediate-encoding egfr_plx_ckappa_gathered.txt";
--output-pretty-qualitative-compression egfr_plx_compressed_qualitative.txt
--output-pretty-quantitative-compression egfr_plx_compressed_quantitative.txt
--output-boolean-encoding egfr_plx_boolean_encoding.txt
--output-gathered-boolean-encoding egfr_plx_boolean_encoding_gathered.txt
--output-pack-constraints egfr_plx_site_constraints.txt
--output-reachable-complexes egfr_plx_reachables.txt
--output-specie-map egfr_plx_specie_map.txt
--output-dag-ref-dot egfr_plx_dag_refinement_relation.dot
--output-dag-ref-jpg egfr_plx_dag_refinement_relation.jpg
--output-maximal-ref-dot egfr_plx_maximal_refinement_relation.dot
--output-maximal-ref-jpg egfr_plx_maximal_refinement_relation.jpg
--output-xml egfr_plx.xml
--output-html egfr_plx.html
--output-refined-system egfr_plx_refinment.ka

regular options

--output-quantitative-compression <name> (default: disabled)
  write the compressed quantitative compression in a .ka format

--output-qualitative-compression <name> (default: disabled)
  write the compressed qualitative compression in a .ka format

--output-pretty-qualitative-compression <name> (default: disabled)
  write the pretty compressed qualitative compression

--output-pretty-quantitative-compression <name> (default: disabled)
  write the pretty compressed quantitative compression

--output-refined-system <name> (default: disabled)
  write the result of rule refinment

--output-influence-map-txt <name> (default: disabled)
  write the causality map with .txt format

--output-influence-map-dot <name> (default: disabled)
  write the causality map with .dot format

--output-low-res-contact-map-dot <name> (default: disabled)
  write the low resolution (aka syntactic) contact map with a .dot format

--output-low-res-contact-map-ps <name> (default: disabled)
  write the low resolution (aka syntactic) contact map with a .ps format

--output-low-res-contact-map-jpg <name> (default: disabled)
  write the low resolution (aka syntactic) contact map with a .jpg format

--output-high-res-contact-map-txt <name> (default: disabled)
  write the high resolution contact map with a .txt format

--output-high-res-contact-map-dot <name> (default: disabled)
  write the high resolution contact map with a .dot format

--output-high-res-contact-map-ps <name> (default: disabled)
  write the high resolution contact map with a .ps format

--output-high-res-contact-map-jpg <name> (default: disabled)
  write the high resolution contact map with a .jpg format

--output-high-res-contact-map-txt <name> (default: disabled)
  write the high resolution contact map with a .txt format

--output-ODE-mathematica <name> (default: disabled)
  write the ODE in mathematica format

--output-ODE-matlab <name> (default: disabled)
  write the ODE in matlab format
  also create subsidiary files of the form r<int>v<int>.m

--output-ODE-alphabet <name> (default: disabled)
  write the meaning of the ODE variables as a list of equation between numbered variable

and kappa observables

--output-ODE-obs <name> (default: disabled)
write the meaning of the ODE variables as a list of observables
this file can be appended to the kappa file in order to perform simulations

--output-pack-constraints <name> (default: disabled)
output the constraints used to define packs for the reachability analysis

--output-reachable-complexes <name> (default: disabled)
write the reachable species (or just their number) in a file

--output-specie-map <name> (default: disabled)
write the specie map in a file

--output-dag-ref-dot (default: disabled)
contains the DAG that relates rules and their refinement in a dot file

--output-dag-ref-jpg (default: disabled)
contains the DAG that relates rules and their refinement in a jpg file

--output-maximal-ref-dot (default: disabled)
contains the relation between rules and their most abstract instances in a dot file

--output-maximal-ref-jpg (default: disabled)
contains the relation between rules and their most abstract instances in a jpg file

--output-xml <name> (default: disabled)
write any computed (in this session or in previous sessions) into an xml file

--output-html <name> (default: disabled)
write any ouput results in a html file

--output-marshalling <name> (default: disabled)
marshallize the computation state

Debug

--(no-)trace_iteration_number (default: enabled)
Dump rule Id before interpreting it

--(no-)version (default: disabled)
to dump version number

Compression

--(no-)compute-qualitative-compression (default: enabled)
simplify the rules

--(no-)compute-quantitative-compression (default: enabled)
simplify the rules

--output-quantitative-compression <name> (default: disabled)
write the compressed quantitative compression in a .ka format

--output-qualitative-compression <name> (default: disabled)
write the compressed qualitative compression in a .ka format

--output-pretty-qualitative-compression <name> (default: disabled)
write the pretty compressed qualitative compression

--max-lens-size <int> (default: 5)
not compress rules with more connected components in their left hand side

--(no-)comment-in-compression (default: enabled)
put(or remove) the initial kappa file in comments in the .ka output

--(no-)dump-concrete-rules (default: enabled)
dump rules before compression in the pretty printed output

--(no-)dump-abstract-rules (default: enabled)
dump rules after compression in the pretty printed output

Concretization

--(no-)enumerate-complexes (default: disabled)
enumerate complexes even if there are too numerous

--(no-)dump-all-complexes (default: enabled)
display complexes, otherwise just show their number

--output-reachable-complexes <name> (default: disabled)
write reachable species (or just their number) in a file

--(no-)sort-complexes (default: enabled)
to sort complexes before dumping them

--complex-limit <int> (default: 1000000)
only enumerate complexes when there are less complexes or the option --enumerate-complexes is triggered

Contact map

--(no-)do-low-res-contact-map (default: enabled)

compute the low resolution (aka syntactic) contact map

--(no-)do-high-res-contact-map (default: enabled)
compute the high resolution contact map

--focus-on (default: disabled)
narrow the syntactic contact map around the rules in these file

--output-low-res-contact-map-dot <name> (default: disabled)
write the low resolution contact map with a .dot format

--output-low-res-contact-map-ps <name> (default: disabled)
write the low resolution contact map with a .ps format

--output-low-res-contact-map-jpg <name> (default: disabled)
write the low resolution contact map with a .jpg format

--output-low-res-contact-map-txt <name> (default: disabled)
write the low resolution contact map with a .txt format

--output-high-res-contact-map-dot <name> (default: disabled)
write the high resolution contact map with a .dot format

--output-high-res-contact-map-ps <name> (default: disabled)
write the high resolution contact map with a .ps format

--output-high-res-contact-map-jpg <name> (default: disabled)
write the high resolution contact map with a .jpg format

--output-high-res-contact-map-txt <name> (default: disabled)
write the high resolution contact map with a .txt format

--(no-)find-cycles (default: disabled)
computes the cycles in the contact map

--(no-)find-connected-components (default: disabled)
computes the connetected components in the contact map

--boolean-site-color <name> (default: yellow)
color of sites that cannot be bound

--boundable-site-color <name> (default: cyan)
color of sites that cannot be marked

--both-site-color <name> (default: green)
color of sites that can be both marked and bound

--agent0 <name> (default: floralwhite)
color of agent that have no site

--agent1 <name> (default: skyblue1)
color of agent that have one site

```
--agent2 <name> (default: deeppink)
  color of agent that have two sites

--agent3 <name> (default: darkorchid)
  color of agent that have three sites

--agentn <name> (default: red)
  color of agent that have many sites
```

External applications

--html-browser <name> (default: firefox)
comand line for launching an html browser

HTML

--(no-)do-HTML (default: enabled)
Launch HTML desktop

--output-html <name> (default: disabled)
write an html file

--html-browser <name> (default: firefox)
comand line for launching an html browser

Influence map

--(no-)build-influence-map (default: enabled)
construct influence maps

--output-influence-map-txt <name> (default: disabled)
write the causality map with .txt format

--output-influence-map-dot <name> (default: disabled)
write the causality map with .dot format

--(no-)wake-up-map (default: enabled)
build wake up relations

--(no-)inhibition-map (default: enabled)
build inhibition map

Marshalling

--(no-)do-marshalling (default: enabled)
Dump mashallization

--input-marshalling <name> (default: disabled)
  Start computation from this marshallized state

--output-marshalling <name> (default: disabled)
  marshallize the computation state

Memory usage

--memory-limit <int> (default: 0)
  Limit the memory usage in Mb, if 0 then there is no limitation
  Any value greater than 4Gb will ba considered as 4Gb

ODE

--(no-)do-ODE (default: disabled)
 compute the ODE system

--output-ODE-mathematica <name> (default: disabled)
  write the ODE in mathematica format

--output-ODE-matlab <name> (default: disabled)
  write the ODE in matlab format
  also create subsidiary files of the form r<int>v<int>.m

--output-ODE-alphabet <name> (default: disabled)
  write the meaning of the ODE variables as a list of equation between numbered variable and kappa observables

--output-ODE-obs <name> (default: disabled)
  write the meaning of the ODE variables as a list of observables
  this file can be appended to the kappa file in order to perform simulations

--initial-time <float> (default: 0.000000)
  initial time for ODE integration

--final-time <float> (default: 1.000000)
  final time for ODE integration

--epsilon-value <float> (default: 0.000010)
  smallest float greater than 0.

--(no-)flat-mode (default: disabled)
  complute the flat ODE system instead of the compressed one

Polymers prevention

--(no-)do-refine-to-force-cycles (default: disabled)
refine the system to avoid the formation of polymers.

--output-refined-system <name> (default: disabled)
write the refined system in the given file

--cycles-depth <int> (default: 10)
only prevents polymeres for cycles bigger than the argument

--(no-)cycle-detection-mode (default: disabled)
show warning, but not refine rule

--(no-)use-constraints-to-refine (default: disabled)
use constraints to avoid combinatorial blow up

--refine-only-these-rules (default: disabled)
refine only these rules

--kynetic-amplifier <float> (default: 1.000000)
multiplifying factor for the kynetics of rules that close cycles

Reachability analysis

--(no-)compute-local-views (default: enabled)
compute reachability analysis

--output-pack-constraints <name> (default: disabled)
dump contraints among sites in a file

--output-reachable-complexes <name> (default: disabled)
write the reachable states in a file

--output-specie-map <name> (default: disabled)
write the specie map in a file

--(no-)auto-packs (default: enabled)
use automatic packing

--(no-)abstract-away-relations-between-sites (default: disabled)
Abstract away any relation between sites

--(no-)abstract-away-relations-between-phosphorilation-and-binding (default: disabled)
to abstract away any relation between phosphorilation and binding

--(no-)abstract-away-information-about-phosphorilation (default: disabled)
to abstract away information about binding

--(no-)abstract-away-information-about-binding (default: disabled)

to abstract away information about phosphorilation

Semantics

--(no-)forward (default: disabled)
to ignore reciproque reactions

Standard output

--(no-)dump-rule-iteration (default: enabled)
to dump the number of rules when iterating

--(no-)dump-iteration-number (default: enabled)
to dump whole iteration number

XML

--(no-)do-XML (default: enabled)
dump XML session

--output-xml <name> (default: disabled)
write an xml file

Input formats

Complx can accept a kappa file:
complx models/egfr/egfr.ka
The grammar of kappa file is given in the Simplx API.

When the option --input-marshalling is used, complx restore the state of a previous session, and perform further computation according to command line options.

The options --focus-on <name> and --refine-only-these-rules <name> input some ASCII files containing Kappa rules.
These rules are used to narrow the contact map or the refinement algorithm to a subset of rules

Output formats

The following line:
complx models/egfr/egfr.ka --output-scheme models/egfr/egfr
produces many output files:

models/egfr/egfr_plx_boolean_encoding_gathered.txt
this file is a boolean representation of the transition system
in this file, rules are gathered when they perform the same sequence of actions

models/egfr/egfr_plx_boolean_encoding.txt
this file is a boolean representation of the transition system.

- models/egfr/egfr_plx_ckappa_gathered.txt
this file is a description of the transition system in a precompiled form
in this file, rules are gathered when they perform the same sequence of actions

- models/egfr/egfr_plx_ckappa.txt
this file is a description of the transition system in a precompiled form.

- models/egfr/egfr_plx_compressed_qualitative.ka
this file is a kappa file;
some rules have been commented and replaced with simple ones;
some rules have been commented because they are covered by other;
some rules are commented because they have been detected to be unappliable;
the new transition system if equivalent with respect to qualitative properties
(it may break kynetics).

- models/egfr/egfr_plx_compressed_qualitative.txt
dump the compression in a pretty way.
the new transition system if equivalent with respect to qualitative properties
(it may break kynetics).

- models/egfr/egfr_plx_compressed_quantitative.ka
this file is a kappa file;
some rules have been commented and replaced with simple ones;
some rules have been commented because they are covered by other;
some rules are commented because they have been detected to be unappliable;
the new transition system if equivalent whatever the kynetic parameters are.

- models/egfr/egfr_plx_compressed_quantitative.txt
dump the compression in a pretty way.
the new transition system if equivalent whatever the kynetic parameters are.

- models/egfr/egfr_plx_high_res_contact.dot

this file contains a description of the high resolution contact map in a dot file;
the following instruction:
dot -Tps models/egfr/egfr_plx_high_res_contact.dot -o models/egfr/egfr_plx_high_res_contact.ps
generates a postscript file.

- models/egfr/egfr_plx_high_res_contact.jpg
this file is a jpg image of the high resolution contact map.

- models/egfr/egfr_plx_high_res_contact.ps
this file is a ps image of the high resolution contact map.

- models/egfr/egfr_plx_high_res_contact.txt
this file contains a text description of the low resolution contact map.

- models/egfr/egfr_plx.html
this file contains a html desktop.

- models/egfr/egfr_plx_influence_map.txt
this file is a text description of the influence map;
it may be an overapproximation (i.e. some edges are false positive).

- models/egfr/egfr_plx_low_res_contact.dot
this file contains a description of the low resolution contact map in a dot file;
the following instruction:
dot -Tps models/egfr/egfr_plx_low_res_contact.dot -o models/egfr/egfr_plx_low_res_contact.ps
generates a postscript file.

- models/egfr/egfr_plx_low_res_contact.jpg
this file is a jpg image of the low resolution contact map.

- models/egfr/egfr_plx_low_res_contact.ps
this file is a ps image of the low resolution contact map.

- models/egfr/egfr_plx_low_res_contact.txt
this file contains a text description of the low resolution contact map.

- models/egfr/egfr_plx.marshalling
this file contains the marshallization of the current session.

- models/egfr/egfr_plx_ODE_alphabet
this file contains the meaning of the ODE variables.

- models/egfr/egfr_plx_ODE_contact.dot
  this file contains the contact map with dotted edges
  dotted edges are link that are ignored when computing the ODE

- models/egfr/egfr_plx_ODE_obs
  this file contains the meaning of the ODE variables as kappa observable,
  it can be appended to a kappa file for futher simulation

- models/egfr/egfr_plx_ODE_system.m
  this file contains the ODE in matlab format
  it uses subsidiary files of the form models/egfr/<int>v<int.mr

- models/egfr/egfr_plx_ODE_system.nb
  this file contains the ODE in mathematica format

- models/egfr/egfr_plx_reachables.txt
  this file contains a description of all reachable complexes;
  first some acyclic complexes are given;
  then some partial complexes to generate others are given;
  they all satisfy Kappa-syntax, but semi-links may contain more information:
    A(x!B.y) means that the site x is bound to the site y of an agent B

  An upper bound to the number of complexes is given when it is possible.

- models/egfr/egfr_plx_refinment.ka
  this file contains a refinment of rules that are given in arguments, so that the transformed system cannot build polymers
  rules are given in a kappa-friendly grammar

- models/egfr/egfr_plx_site_constraints.txt
  this file contains a description of constraints among sites.
  they all satisfy Kappa-syntax, but semi-links may contain more information:
    A(x!B.y) means that the site x is bound to the site y of an agent B

- models/egfr/egfr_plx_specie_map.txt
  this file contains a description of all local views;
  they all satisfy Kappa-syntax, but semi-links may contain more information:
    A(x!B.y) means that the site x is bound to the site y of an agent B

- models/egfr/egfr_plx.xml
  this file contains a XML description of the session.

- models/egfr/egfr_plx_dag_refinement_relation.dot
  this file contains a dot description of the refinement relation between rules

- models/egfr/egfr_plx_dag_refinement_relation.jpg
  this file contains a jpg description of the refinement relation between rules

- models/egfr/egfr_plx_maximal_refinement_relation.dot
  this file contains a dot description of the refinement relation between each rule and their most abstract instance

- models/egfr/egfr_plx_maximal_refinement_relation.jpg
  this file contains a jpg description of the refinement relation between each rule and their most abstract instance

Internal representation

The intenal representation signature is described in the file frontend/pb_sig.ml.
Some external applications may use this data-structures and update it by using the pipeline methods.

Here the signature of the data-structures (subsidiairy types are declared in frontend/pb_sig.ml).

```
type 'a pb =
{
options:options option,
  this field describes analysis options
quarks:bool;
  this field contains quark information  to compute the influence map
txt_lines: Comment_sig.commented_line list option;
  this field contains the stucture of the kappa file (with comments)
simplx_encoding: (Rule.t list * Solution.t) option;
  this field contains the output of simplx parser
first_encoding: 'a cpb option;
intermediate_encoding: 'a cpb option;
  ckappa encoding (rules are isolated)
  it also contains the low resolution contact map
gathered_intermediate_encoding: 'a cpb option;
  ckappa encoding (rules are gathered)
boolean_encoding: 'a boolean_encoding option;
  boolean encoding (rules are isolated)
gathered_boolean_encoding: 'a boolean_encoding option;
  boolean encoding (rules are gathered)
packs: string list list StringMap.t option;
  packing definition
reachability_analysis: ((string*'a) list StringMap.t) option;
```

sub local views (abstract representation)
contact_map:contact_map option ;
high resolution contact map
bdd_sub_view: 'a internalsubviews option ;
sub local views (bdd abstract representation)
bdd_false:'a internalsubviews option ;
internal representation for missing agents (bdd abstract representation)
concretization:(string * (bool * pretty StringMap.t) * pretty_fun) list list option;
list of reachable complexes (abstract encoding)
reachable_complexes: (string intinf * (string list list * int) list * (string list list * int) list) option;
list of reachable complexes (numbers and pretty printed complexes)
wake_up_map:influence_map option;
low resolution influence map (positive)
inhibition_map:influence_map option;
low resolution influence map (negative)
pretty_map: pretty StringMap.t StringMap.t;
data to pretty print local views and complexes
qualitative_compression: compression option;
compression result
quantitative_compression: compression option;
compression result
unreachable_rules: RuleIdSet.t option;
dead rules
rule_warning: string list RuleIdMap.t;
a list of warning associated to rules
specie_map:(string * string list list) list option;
local views
pack_value:(string * (string list * string list list) list) list option;
sub local views (pretty printed)
n_complex:string intinf option;
number of complexes
n_rules:int option;
number of rules
n_classes:int option;
number of classes of rules (gathered by sequence of actions)
potential_cycles:(string*string) list list option;
list of any potential cycle in the contact map
connected_components:string list list option;
list of any connected component in the contact map
refined_system:'a boolean_encoding IntMap.t;
maps each rule to the internal (boolean) representation of its refinment
drawers:drawers option;
contains the relations between contact map items and rules
refinement_relation_closure:IntSet.t IntMap.t list option;
contains the closure of the relation between rules and their refinements
first partitionned into a list of action sequences, and for each action sequence the relation
refinement_relation_dag:IntSet.t IntMap.t list option;
contains the relation between rules and their most concrete refinements
first partitionned into a list of action sequences, and for each action sequence the relation
refinement_relation_maximale:IntSet.t IntMap.t list option;
contains the relation between rules and their most abstract refinements
first partitionned into a list of action sequences, and for each action sequence the relation

```
}
```

Pipeline methods

The pipeline provides some methods to progress in the computation.
Each method fill fields in the data structure.
Built-in dependences help methods in computing required fields before making its own computation.

```
type prefix  = (string*(string list))
```
(the first component should be printed before progress line status in the standard output, the second one is a calling stack for handling exceptions)
```
type file_name = string
type simplx_encoding = (Rule.t list * Solution.t) option
type 'a intermediate_encoding = 'a Pb_sig.cpb option
type 'a boolean_encoding = 'a Pb_sig.boolean_encoding option
type 'a internal_encoding = 'a Pb_sig.pb option
type log = (string*float) list
type message = string list
type output_channel = log*message
type influence_map = IntSet.t IntMap.t * IntSet.t IntMap.t
```

```
type 'a step = 'a internal_encoding -> output_channel -> 'a internal_encoding *
output_channel
```
A standard method takes two arguments:
  - the current step of computation
  - a message channel
and returns two values:
  - the updated step of computation
  - the updtaed channel.

```
type ('a,'b) step_with_output = 'a internal_encoding -> output_channel -> 'b * 'a
internal_encoding * output_channel
```
A standard method takes two arguments:
  - the current step of computation
  - a message channel
and returns three values:
  - a result
  - the updated step of computation
  - the updtaed channel.

```
type 'a pipeline =
    {

    reset:'a step;
```
    to remove any obselete information (due to analysis option modification)

message handlers

dump_version: prefix -> output_channel -> output_channel;
to add the version number in the output channel

print_footpage: prefix -> output_channel -> output_channel;
to dump the footpage

print_headpage: prefix ->output_channel -> output_channel;
to dump the headpage

empty_channel: output_channel;
to build an empty channel

log_time: prefix -> string -> output_channel -> output_channel;
to log a computation step in a channel

add_message: prefix -> string -> output_channel -> output_channel;
to add a message in a channel

print_channel: prefix -> output_channel -> unit;
to dump the content of a channel

other methods

parse_file: file_name -> prefix -> output_channel -> 'a internal_encoding*output_channel;
to use parse a file thanks to simplx

parse_line_by_line: file_name -> 'a step ;
to parse a file for complx backend

unmarshallize: file_name -> output_channel -> 'a internal_encoding*output_channel;
to restore a marshallized computation step

marshallize: file_name -> 'a step;
to save a computation step

build_pb: simplx_encoding -> 'a internal_encoding;
to translate simplx compilation output in an internal representation

translate: 'a step;
to translate simplx compitation encoding into ckappa encoding

dump_ckappa: file_name -> compile -> 'a step;
to dump the ckappa encoding; when the second argument is "Smashed", rules are gathered in equivalence classes

compile: compile -> 'a step;
to convert ckappa encoding into boolean encoding; when the second argument is "Smashed", rules are gathered in equivalence classes

dump_boolean_encoding: file_name -> compile -> 'a step;
to dump the boolean encoding; when the second argument is "Smashed", rules are gathered in equivalence classes

build_contact: precision -> 'a step;
to build the contact map

find_potential_cycles: precision -> 'a step;
detect the set of cycles in the contact map,
the first argument can be either Low, or High
in low precision, we smash all sites

find_connected_components: precision -> 'a step;
detect the set of connected components in the contact map
the first argument can be either Low, or High
in low precision, we smash all sites

reachability_analysis: 'a step;
to compute local views

refine_subviews: 'a step;
take into account the computation of subviews

refine_views: 'a step;
gather subviews into views

dump_local_views: file_name -> 'a step;
to dump local views

dump_packs_constraints: file_name -> 'a step;
to dump sub-local views

dump_contact_map_txt: precision -> ile_name -> 'a step;
to dump the contact map in txt formal

dump_contact_map_dot: precision -> file_name -> 'a step;
to dump the contact map in dot format

dump_contact_map_ps: precision -> file_name -> 'a step;
to dump the contact map in ps format

dump_contact_map_jpg: precision -> file_name -> 'a step;
to dump the contact map in jpg format

quarkification: 'a step;
to compute the quark encoding

count_complexes: 'a step;
to count reachable complexes without enumerating them

build_influence_map: file_name -> file_name -> 'a step;
to build the influence map

```
build_compression: compression_mode -> file_name -> file_name -> 'a step;
```
to compute the compression

```
build_enumeration: file_name -> 'a step;
```
to enumerate reachable complexes

```
dump_session: file_name -> 'a step;
```
to dump the XML output

```
dump_html_output: file_name -> 'a step;
```
to compute the HTML desktop

```
good_vertice: file_name -> prefix -> output_channel -> StringSet.t option *
output_channel;
```
compute the set of agents that occurs in a list of rules

```
template: file_name -> file_name -> file_name -> file_name -> file_name -> file_name
-> file_name -> file_name -> 'a step;
```
compute the ODE

```
dump_potential_cycles: precision -> 'a step;
```
dump the potential cycles on the standard output
if precision = Low, sites are smashed.

```
refine_system_to_avoid_polymers: file_name -> simplx_encoding option ->
Avoid_polymere.mode -> int option -> float -> ('a,('a rule_class list)) step_with_output;
```
refine some rules to prevent polymerization

```
build_drawers:  'a step;
```
build the relations between items in the contact map and the rules

```
compute_refinement_relation_maximal:  'a step;
```
build the relations between rules and their most abstract instance

```
export_refinement_relation_maximal:  ('a,Rule.t list option) step_with_output;
```
export the result of the relation between rules and their most abstract instance in the list of rules of simplx

```
compute_refinement_relation_dag:  'a step;
```
build the DAG between rules and their refinements

```
compute_refinement_relation_closure:  'a step;
```
build all the relations between rules and their refinements

```
dump_maximal_refinement_relation:  file_name -> file_name -> 'a step;
```
dump the relations between rules and their most abstract instance, the fist file is the dot file, the second one is the jpg file

```
dump_refinement_relation_dag:  file_name -> file_name -> a step;
```
build the DAG between rules and their refinements, the fist file is the dot file, the second one is the jpg file

```
save_options: 'a step
```
to remember which options have been used

}

Module tree

complx_rep/:

abstract_expr — abstract domain for expressions description
backend
compressor — compression modules
config — config files
cyclical_complexes — detection of potential cycles and automatic refinments to avoid them
data_structures
expr — expression description
frontend
influence_map — influence map construction
lib — external libraries
main.ml — usual pipeline
Makefile
ODE — ODE synthesis
pipeline — pipeline modules
reachability — reachability analsis modules

refinements — detection of refinement relation between rules
share
tools — tools modules
vars — variable description

complx_rep/abstract_expr:

abstract_expr_sig.ml — signature for abstract expression modules
bdd.ml — binary decision diagram implementation module
partition.ml — partitioning functor
rough.ml — propositional logic implementation module

complx_rep/backend:

contact_map — dump the contact map
HTML
parse_comment — (re)parse the kappa file to insert comments
XML

complx_rep/backend/contact_map:

acyclicity.ml — check whether they might be polymeres according to the contact map

connected_components.ml compute the connected componnents in the contact map
fin_cycles.ml enumerate the cycles according to a contact map
neighborhood.ml restrict a contact map around some agents
output_contact_map.ml output the contact map in various format

complx_rep/backend/HTML:

html.ml generate the HTML desktop

complx_rep/backend/parse_comment:

comment_sig.ml result signature
lexeur.mll alphabet
yacc.mli2 grammar signature
yacc.mly grammar

complx_rep/backend/XML:

xml.ml generate the XML data-structure

complx_rep/compressor:

compressor.ml library to compress rule system

complx_rep/config:

config_complx.ml default values, option list and parsing command line option

complx_rep/data_structures:

big_array.ml to use array with size < max_int
data_structures.ml data structures (maps, sets)
hash.ml hashtables implementations

complx_rep/expr:

expr.ml module for boolean expressions
kleenean_expr.ml module for three value-logic expressions

complx_rep/frontend:

cbng.ml	translation from ckappa to boolean encoding
cbng_sig.ml	ckappa signatures
pb_sig.ml	boolean encoding signature
translate.ml	translation from simplx encoding to ckappa.

complx_rep/influence_map:

influence_map.ml	functions to build the flow map
quarkification.ml	build the quark encoding

complx_rep/lib:

full	version for Labltk
light	version without Labltk
with_key	version with key-protection
without_key	version without key-protection
ml_wordexp.c	Antoine's library
superarg.ml	Antoine's library (for command line options)
wordexp.ml	Antoine's library

complx_rep/lib/full:

superargTk.ml	Antoine's library

complx_rep/lib/light:

superargTk.ml	Antoine's library (patched to compile without LablTk)

complx_rep/lib/with_key:

key.ml	Deal with key protection

complx_rep/lib/without_key:

key.ml	Avoid key protection

complx_rep/ODE:

annotated_contact_map.ml	Primitives to compute the annotated contact map
fragments.ml	Primitives to handle with fragments
ode_computation.ml	Compute the projection of ODE over fragements
ode_print.ml	Pretty-printing primitives
ode_print_sig.ml	Type definition for pretty printing primitives
views.ml	Primitives to handle with views

complx_rep/pipeline:

pipeline.ml	pipeline definition

complx_rep/reachability:

concretization.ml	complex enumeration
contact_map.ml	contact map signature
count_complexes.ml	count complexes without enumerating them
packing.ml	automatic packing algorithm
reachability.ml	reachability analysis iterator

complx_rep/refinements:

refinements.ml	primitives to deal with refinement relation between rules

complx_rep/share:

share.ml

complx_rep/tools:

array_ext.ml	potentially infinite array (for simplx)
error_handler.ml	primitives to catch exceptions and dump XML output
exceptions.ml	exception declaration
map2.ml	patched map module (for binary operators)
map_random.ml	patched map module (for random choice)
map_random.ml	patched map module (with optimized sharing)
memory_usage.ml	track memory usage and launch exception
tools2.ml	various functions
tools.ml	various functions

unbounded_array.ml potentially infinite array (for complx)

complx_rep/vars:

var.ml variable description

Bug report

The errors are stacked LOG file of the XML output.
For instance:

```
<Log>
<Entry Type="ERROR" Application="Complx" Method="reachability_analysis"
Exception="Exit" Stack="reachability_analysis,convert_contact,get_intermediate_encoding"
Message="MEMORY OVERFLOW"/>
</Log>
```

Please warn me by sending me both the input files and the XML output.