Adam: The User Guide

Manuel Gieseking

Carl von Ossietzky Universität Oldenburg
manuel.gieseking(at)informatik.uni-oldenburg.de

May 27, 2015

Abstract. Adam (**A**nalyzer of **D**istributed **A**synchronous **M**odels) is a synthesis tool for reactive systems with multiple independent processes. The systems are modelled as Petri games, games with one environment player and an arbitrary but bounded number of system players described as Petri nets. Adam synthesis winning strategies of the Petri games by reducing the Petri game to a finite-graph game, solving the graph game and reconstructing the Petri game strategy from the one of the finite-graph game.
Adam is written in Java and uses JavaBDD as library for manipulating BDDs. The APT format is used as input / output format and APT itself is used for parsing / rendering the Petri games and for providing a data structure for Petri nets. For visualizing the finite-graph games, the Petri games, and their strategies, Adam uses the DOT format of Graphviz.

Table of Contents

1 Dependencies

The dependencies for using Adam in the most comfortable way are:

- Java in a version greater or equal to 7 is needed.
- For saving the games and strategies as a pdf file, dot (Graphviz) has to be installed in a version $\geq$ 2.36.0.

– Please stick to the documentation of JavaBDD, if you would like to use another library (like BuDDy, CUDD, CAL, etc.) for the BDD manipulation than the Java implementation of JavaBDD. The compiled file of such a library has then to be placed on the same level as the jar file of Adam. The parameter for choosing a different library is applicable from the user interface. By default, Adam uses BuDDy, if it's accessible, otherwise Adam falls back to the Java implementation of JavaBDD.

2 Installation

In order to install Adam, just extract the tarball, which can be found here. This should create a folder named 'adam' containing the program 'adam.jar', a compiled version of the BDD library BuDDy 'libbuddy.so', a README file on how to use Adam and a folder 'examples' containing some Petri games and their strategies.

To run Adam on Linux systems, you can execute the bash script named 'adam.sh', also placed in this folder, or directly use Java. More details on starting and using Adam, see Sec. 3.2.

3 Usage

Adam has three categories of modules. There are converters from Petri games, defined in the APT file format, to the .dot format or to a pdf document visualized by Graphviz. Then, there are generators for some example Petri games, which are also used within the benchmark suite. Finally, there are modules creating finite graph or Petri game strategies.

3.1 List of Available Modules

This section lists all programs Adam provides, which can also be printed by executing 'sh adam.sh' or 'java -jar adam.jar'.

```
Usage: sh adam.sh <module> or java -jar adam.jar <module>
Available modules:
  pg2dot             Converts a Petri game in APT format to a dot file.
  pg2pdf             Converts a Petri game in APT format to a pdf file by
                     using Graphviz (dot has to be executable).
  ex_win_strat       Returns true if there is a deadlock-avoiding winning
                     strategy (system players) for the in APT format
                     given Petri game.
  win_strat_graph    Creates a deadlock-avoiding winning strategy (system
                     players) in the finite graph game of the in APT
                     format given Petri game if existent. Saves the
                     strategy in the given folder as dot, and if dot is
                     executable, as pdf.
  win_strat_pg       Creates a deadlock-avoiding winning strategy (system
                     players) for the in APT format given Petri game if
                     existent. Saves the strategy in the given folder as
                     dot, and if dot is executable, as pdf.
  win_strat          Creates a deadlock-avoiding winning strategy (system
                     players) in the finite graph game and the Petri game
```

```
                 for the in APT format given Petri game if existent.
                 Saves the strategies in the given folder as dot, and
                 if dot is executable, as pdf. Adding '_strat_fg' for
                 the graph and '_strat_pg' for the Petri game
                 strategy to the filename, respectively. Also saves
                 the input Petri game with the partition of the
                 places within the same folder.
  bench          Just for benchmark purposes. Does not check any
                 preconditions of the Petri game. Tests existence of
                 strategy, calculates graph and Petri game strategy,
                 saves Petri game strategy as dot without rendering
                 it to a pdf file.
  export         Exports some data from ADAM. At the moment only a
                 LaTeX export for the help dialogues is implemented.
  gen_phil       Generates the philosopher problem for the given
                 number of philosophers and saves the resulting net
                 in the APT and dot format and, if dot is executable,
                 as pdf.
  gen_clerks     Generates the given number of clerks signing a
                 document and saves the resulting net in APT and dot
                 format and, if dot is executable, as pdf. This
                 module generates the Document Workflow examples of
                 the ADAM paper.
  gen_workflow   Generates the workflow examples. Saves the resulting
                 net in APT and dot format and, if dot is executable,
                 as pdf. This module generates the Job Processing
                 example of the ADAM paper.
  gen_robots     Generates the self-organizing robots examples. Saves
                 the resulting net in APT and dot format and, if dot
                 is executable, as pdf. This module generates the
                 Self-reconfiguring Robots example of the ADAM paper.
  gen_workflow2  Generates the workflow2 examples. Saves the
                 resulting net in APT and dot format and, if dot is
                 executable, as pdf. This module generates the
                 Concurrent Machines example of the ADAM paper.
```

A module can be executed by typing: 'sh adam.sh <module>' or 'java -jar adam.jar <module>'. This prints a help dialogue how to use this module. All help dialogues can be found in the Appendix.

3.2 Executing the Modules

Executing

```
sh adam.sh
```

or

```
java -jar adam.jar
```

prints a list of all possible modules (see Sec 3.1). The modules themselves can be started by executing

```
sh adam.sh <module>
```

This prints a help dialogue stating the usage of this module and the available and necessary parameters. For each module this dialogue can be found in the Appendix.

Subsequently, we give some standard calls for ADAM:

```
sh adam.sh pg2pdf -i ./folder/name.apt
```

This call saves the visualisation of the in APT format given Petri game as pdf file in the same folder.

```
sh adam.sh win_strat -i ./folder/name.apt
```

This is a standard call for creating the winning strategies (finite graph and Petri game) if existent. It saves them (.dot, .pdf) in the same folder as the in APT format given Petri game. Also saves the visualization of the input Petri game with the used distribution of the places, visualized by different colouring of the places.

```
sh adam.sh gen_workflow2 -m 2 -w 4 -o cw24
```

This call generates the Concurrent Machines (CM) examples of the Adam paper for two machines and four orders. It saves the resulting Petri game as APT, dot and pdf file to the given file name (cw24.apt, cw24.dot, cw24.pdf).

Most of the optional parameters are self-explanatory. Thus, we only want to list some special ones.

The parameter *skip* forces Adam to skip all tests regarding the wellformedness of the Petri game in the input file. That is, it skips the test checking if the underlying Petri net is safe. Other tests, like checking if there is only one environment token, are not yet implemented. The tests of the wellformedness of the Petri net aren't skipped.

The parameter *exp* is a flag, which forces Adam to use an experimental algorithm. In Section 3.3. we explain the input format of the Petri games which Adam can use. There is stated that we need to distribute the places of the Petri game in disjunct sets satisfying some properties. This experimental version is much slower, but has the ability to calculate the strategies for a subgroup of underlying Petri nets (concurrency-preserving), without the need of any distribution of the places.

The parameters starting with *lib* concern the BDD library. With them you can exchange the library used for the BDD calculation, if you have compiled C libs in the same folder as Adam, and set some parameters for the node table and its usage.

3.3 Creating your own input files

The data format for modelling a Petri game is described here in detail. There is also a formal grammar for parsing Petri nets given within the document. In the following, we give a quick summary.

Adam works on safe Petri nets. If your net is not safe, you should transform it into a safe net before running Adam by adding additional places. Another precondition is that you can use an arbitrary (bounded) number of system players, but only one environment player. Thus, you have to make sure that there is no reachable marking in which two environment places are marked at the same time. Furthermore, an environment player cannot convert into a system player or vice versa. Thus, make sure that no environment token can occupy a system place or vice versa.

An input file contains of sections for places (*.places*), transitions (*.transitions*) and the connections between them (*.flows*). You have to name the Petri game (*.name ”my name”*) and set its type (*.type LNP*). If you like to, you can give a description of the game with *.description ”lorem”*. The section *.initial_marking* contains the initial marking of the Petri game.

Please do not use any underscores ('_') within the names of your places while creating your own input file in the APT file format. This causes trouble during calculating the Petri game strategy.

For typing a place as an environment place annotate him with *[env=”true”]*, all other places are automatically typed as belonging to the system players. Do the same for bad places with *[bad=”true”]*.

The places of the Petri game aren't just divided into environment and system places, but also into groups of places stating these are the places a given token can lie on. Thus, you can annotate the places with the *[token=<number>]* key-value pair, where <number>is the number of the group this place belongs to. You only have to annotate the system places. Environment places are automatically marked as group 0. Thus, the numbers annotated must start from 1 and it is not allowed to omit some natural number between 1 and the maximum number annotated. Be aware that the partition has to be disjunct in the sense that no two places in one group can be marked at the same time. ADAM has a feature to support you by annotating the places or even completely annotate the places on its own. To use this, just don't annotate any places with *[token=<number>]* and use, for example, the *win_strat* module. If ADAM can annotate the places on its own, you can read the annotation by the colouring of the places in the resulting pdf file of the Petri game. Otherwise, ADAM prints some invariants which should support you by following the token through the net.

Here is an example input file:

```
.name "my name"
.type LNP

.places
env1[env="true"]
bad0[token=1, bad="true"]
bad1[token=2, bad="true"]
good[token=2]
...
env0[env="true"]

.transitions
t1 t2
t3
...

.flows
t1: {env1} -> {env0}
t1: {bad0, bad1} -> {bad0}
...

.initial_marking {env1, good}
```

For more examples, see the examples folder within the tarball (*./examples/*) or the file defining the input format.

4 Contact

We appreciate your feedback on Adam! Please send any bugs, comments, or questions to: *manuel.gieseking(at)informatik.uni-oldenburg.de*

Thank you for using Adam!

A Appendix

Following the help dialogues of each module. That is, how to call the module, the possible and needed parameters including their explanations.

Module: pg2dot

Converts a Petri game in APT format to a dot file. The help dialogue:

```
usage: java -jar adam.jar pg2dot [-h] -i <file> [-o <file>]
Converts a Petri game in APT format to a dot file.
 -h,--help             Prints this dialog.
 -i,--input <file>     The path to the input file in APT format, which
                       should be converted.
 -o,--output <file>    The path to the output file. If it's not given the
                       path from the input file is used.
```

Module: pg2pdf

Converts a Petri game in APT format to a pdf file by using Graphviz (dot has to be executable). The help dialogue:

```
usage: java -jar adam.jar pg2pdf [-h] -i <file> [-o <file>]
Converts a Petri game in APT format to a pdf file by using Graphviz (dot
has to be executable).
 -h,--help             Prints this dialog.
 -i,--input <file>     The path to the input file in APT format, which
                       should be converted.
 -o,--output <file>    The path to the output file. If it's not given the
                       path from the input file is used.
```

Module: ex_win_strat

Returns true if there is a deadlock-avoiding winning strategy (system players) for the in APT format given Petri game. The help dialogue:

```
usage: java -jar adam.jar ex_win_strat [-exp] [-h] -i <file> [-l <file>]
       [-lib <lib>] [-mi <nb>] [-nnb <nb>] [-oc <nb>] [-s] [-v]
Returns true if there is a deadlock-avoiding winning strategy (system
players) for the in APT format given Petri game.
 -exp,--experimental      Use the experimental version. Trys to find a
                          strategy without a given distribution annotated
                          to the places. The Petri net must be
                          concurreny-preserving. Currently still very
                          slow. No other optional parameters have any
                          effect.
 -h,--help                Prints this dialog.
 -i,--input <file>        The path to the input file in APT format, which
```

```
                          should be investigated.
 -l,--logger <file>       The path to an optional logger file. If it's
                          not set, the information will be send to the
                          terminal.
 -lib,--BDDlib <lib>      The BDD library which you would like to use.
                          Possible values: 'buddy', 'cudd', 'cal',
                          'java', 'jdd'.  If the chosen C library isn't
                          available, ADAM automatically falls back to the
                          JavaBDD library ('java'). For more information
                          see: http://javabdd.sourceforge.net/.
 -mi,--libMaxInc <nb>     Sets the maximum number of nodes by which to
                          increase node table after a garbage collection
                          for the BDD library.
 -nnb,--libNodeNb <nb>    Sets the initial node table size for the BDD
                          library.
 -oc,--libOpCache <nb>    Sets the operation cache size for the BDD
                          library.
 -s,--skip                Skips the tests like bounded. Saves time, but
                          should only be used if you are asure that your
                          net fullfills  all necessary preconditions!
 -v,--verbose             Makes the tool chatty.
```

Module: win_strat_graph

Creates a deadlock-avoiding winning strategy (system players) in the finite graph game of the in APT format given Petri game if existent. Saves the strategy in the given folder as dot, and if dot is executable, as pdf. The help dialogue:

```
usage: java -jar adam.jar win_strat_graph [-exp] [-h] -i <file> [-l
       <file>] [-lib <lib>] [-mi <nb>] [-nnb <nb>] [-o <file>] [-oc
       <nb>] [-s] [-v]
Creates a deadlock-avoiding winning strategy (system players) in the
finite graph game of the in APT format given Petri game if existent.
Saves the strategy in the given folder as dot, and if dot is executable,
as pdf.
 -exp,--experimental      Use the experimental version. Trys to find a
                          strategy without a given distribution annotated
                          to the places. The Petri net must be
                          concurreny-preserving. Currently still very
                          slow. No other optional parameters have any
                          effect.
 -h,--help                Prints this dialog.
 -i,--input <file>        The path to the input file in APT format, which
                          should be examined.
 -l,--logger <file>       The path to an optional logger file. If it's
                          not set, the information will be send to the
                          terminal.
 -lib,--BDDlib <lib>      The BDD library which you would like to use.
                          Possible values: 'buddy', 'cudd', 'cal',
                          'java', 'jdd'.  If the chosen C library isn't
                          available, ADAM automatically falls back to the
                          JavaBDD library ('java'). For more information
                          see: http://javabdd.sourceforge.net/.
 -mi,--libMaxInc <nb>     Sets the maximum number of nodes by which to
                          increase node table after a garbage collection
                          for the BDD library.
 -nnb,--libNodeNb <nb>    Sets the initial node table size for the BDD
                          library.
 -o,--output <file>       The path to the output file. If it's not given
                          the path from the input file is used.
 -oc,--libOpCache <nb>    Sets the operation cache size for the BDD
                          library.
 -s,--skip                Skips the tests like bounded. Saves time, but
                          should only be used if you are asure that your
```

```
                             net fullfills  all necessary preconditions!
 -v,--verbose                Makes the tool chatty.
```

Module: win_strat_pg

Creates a deadlock-avoiding winning strategy (system players) for the in APT format given Petri game if existent. Saves the strategy in the given folder as dot, and if dot is executable, as pdf. The help dialogue:

```
usage: java -jar adam.jar win_strat_pg [-exp] [-h] -i <file> [-l <file>]
       [-lib <lib>] [-mi <nb>] [-nnb <nb>] [-o <file>] [-oc <nb>] [-s]
       [-v]
Creates a deadlock-avoiding winning strategy (system players) for the in
APT format given Petri game if existent. Saves the strategy in the given
folder as dot, and if dot is executable, as pdf.
 -exp,--experimental         Use the experimental version. Trys to find a
                             strategy without a given distribution annotated
                             to the places. The Petri net must be
                             concurreny-preserving. Currently still very
                             slow. No other optional parameters have any
                             effect.
 -h,--help                   Prints this dialog.
 -i,--input <file>           The path to the input file in APT format, which
                             should be examined.
 -l,--logger <file>          The path to an optional logger file. If it's
                             not set, the information will be send to the
                             terminal.
 -lib,--BDDlib <lib>         The BDD library which you would like to use.
                             Possible values: 'buddy', 'cudd', 'cal',
                             'java', 'jdd'.  If the chosen C library isn't
                             available, ADAM automatically falls back to the
                             JavaBDD library ('java'). For more information
                             see: http://javabdd.sourceforge.net/.
 -mi,--libMaxInc <nb>        Sets the maximum number of nodes by which to
                             increase node table after a garbage collection
                             for the BDD library.
 -nnb,--libNodeNb <nb>       Sets the initial node table size for the BDD
                             library.
 -o,--output <file>          The path to the output file. If it's not given
                             the path from the input file is used.
 -oc,--libOpCache <nb>       Sets the operation cache size for the BDD
                             library.
 -s,--skip                   Skips the tests like bounded. Saves time, but
                             should only be used if you are asure that your
                             net fullfills  all necessary preconditions!
 -v,--verbose                Makes the tool chatty.
```

Module: win_strat

Creates a deadlock-avoiding winning strategy (system players) in the finite graph game and the Petri game for the in APT format given Petri game if existent. Saves the strategies in the given folder as dot, and if dot is executable, as pdf. Adding '_strat_fg' for the graph and '_strat_pg' for the Petri game strategy to the filename, respectively. Also saves the input Petri game with the partition of the places within the same folder. The help dialogue:

```
usage: java -jar adam.jar win_strat [-exp] [-h] -i <file> [-l <file>]
       [-lib <lib>] [-mi <nb>] [-nnb <nb>] [-o <file>] [-oc <nb>] [-s]
       [-v]
Creates a deadlock-avoiding winning strategy (system players) in the
finite graph game and the Petri game for the in APT format given Petri
game if existent. Saves the strategies in the given folder as dot, and
if dot is executable, as pdf. Adding '_strat_fg' for the graph and
'_strat_pg' for the Petri game strategy to the filename, respectively.
Also saves the input Petri game with the partition of the places within
the same folder.
 -exp,--experimental      Use the experimental version. Trys to find a
                          strategy without a given distribution annotated
                          to the places. The Petri net must be
                          concurreny-preserving. Currently still very
                          slow. No other optional parameters have any
                          effect.
 -h,--help                Prints this dialog.
 -i,--input <file>        The path to the input file in APT format, which
                          should be examined.
 -l,--logger <file>       The path to an optional logger file. If it's
                          not set, the information will be send to the
                          terminal.
 -lib,--BDDlib <lib>      The BDD library which you would like to use.
                          Possible values: 'buddy', 'cudd', 'cal',
                          'java', 'jdd'.  If the chosen C library isn't
                          available, ADAM automatically falls back to the
                          JavaBDD library ('java'). For more information
                          see: http://javabdd.sourceforge.net/.
 -mi,--libMaxInc <nb>     Sets the maximum number of nodes by which to
                          increase node table after a garbage collection
                          for the BDD library.
 -nnb,--libNodeNb <nb>    Sets the initial node table size for the BDD
                          library.
 -o,--output <file>       The path to the output file. If it's not given
                          the path from the input file is used.
 -oc,--libOpCache <nb>    Sets the operation cache size for the BDD
                          library.
 -s,--skip                Skips the tests like bounded. Saves time, but
                          should only be used if you are asure that your
                          net fullfills  all necessary preconditions!
 -v,--verbose             Makes the tool chatty.
```

Module: bench

Just for benchmark purposes. Does not check any preconditions of the Petri game. Tests existence of strategy, calculates graph and Petri game strategy, saves Petri game strategy as dot without rendering it to a pdf file. The help dialogue:

```
usage: java -jar adam.jar bench [-h] -i <file> [-o <file>] [-ob <file>]
       [-s] [-v]
Just for benchmark purposes. Does not check any preconditions of the
Petri game. Tests existence of strategy, calculates graph and Petri game
strategy, saves Petri game strategy as dot without rendering it to a pdf
file.
 -h,--help                 Prints this dialog.
 -i,--input <file>         The path to the input file in APT format,
                           which should be examined.
 -o,--output <file>        The path to the output file. If it's not given
                           the path from the input file is used.
 -ob,--out_bench <file>    The path to the output file for the internal
                           benchmark data.
 -s,--short                Using the short ouput version.
 -v,--verbose              Makes the tool chatty.
```

Module: export

Exports some data from ADAM. At the moment only a LaTeX export for the help dialogues is implemented. The help dialogue:

```
usage: java -jar adam.jar export [-eh] [-h] -o <file> [-v]
Exports some data from ADAM. At the moment only a LaTeX export for the
help dialogues is implemented.
 -eh,--exp_help     Exporting the help dialogues to LaTeX files in the
                    given folder.
 -h,--help          Prints this dialog.
 -o,--out <file>    The path to the output folder for the data to export.
 -v,--verbose       Makes the tool chatty.
```

Module: gen_phil

Generates the philosopher problem for the given number of philosophers and saves the resulting net in the APT and dot format and, if dot is executable, as pdf. The help dialogue:

```
usage: java -jar adam.jar gen_phil [-h] [-ng] [-np] -o <file> -p
       <numberOfPhilosophers>
Generates the philosopher problem for the given number of philosophers
and saves the resulting net in the APT and dot format and, if dot is
executable, as pdf.
 -h,--help                               Prints this dialog.
 -ng                                     If set, it will generated the
                                         non-guided variant, where the
                                         environment is just one of the
                                         philosophers, otherwise the
                                         guided variant will be
                                         generated, where the environment
                                         orders every philosopher to eat
                                         after another. (every clerk has
                                         to vote for yes).
                                         Concurrency-preserving version
                                         (DW in ADAM paper).
 -np,--no_partition                      If set, no automatical partition
                                         of the places is done. Thus, no
                                         annotation from token to places
                                         is printed in the resulting file
                                         in APT format.
 -o,--output <file>                      The output path where the
                                         generated Petri game should be
                                         saved.
 -p,--nb_phils <numberOfPhilosophers>    The desired number of
                                         Philosophers eating (>= 2).
```

Module: gen_clerks

Generates the given number of clerks signing a document and saves the resulting net in APT and dot format and, if dot is executable, as pdf. This module generates the Document Workflow examples of the ADAM paper. The help dialogue:

```
usage: java -jar adam.jar gen_clerks -c <numberOfClerks> [-h] [-np] -o
       <file> [-s]
Generates the given number of clerks signing a document and saves the
resulting net in APT and dot format and, if dot is executable, as pdf.
This module generates the Document Workflow examples of the ADAM paper.
```

```
 -c,--nb_clerks <numberOfClerks>   The desired number of Clerks signing
                                   the document (>= 1).
 -h,--help                         Prints this dialog.
 -np,--no_partition                If set, no automatical partition of
                                   the places is done. Thus, no
                                   annotation from token to places is
                                   printed in the resulting file in APT
                                   format.
 -o,--output <file>                The output path where the generated
                                   Petri game should be saved.
 -s,--simple                       If set, it will be generated the
                                   liberal version (every clerk has to
                                   vote for yes). Concurrency-preserving
                                   version (DWs in ADAM paper).
```

Module: gen_workflow

Generates the workflow examples. Saves the resulting net in APT and dot format and, if dot is executable, as pdf. This module generates the Job Processing example of the ADAM paper. The help dialogue:

```
usage: java -jar adam.jar gen_workflow [-h] -m <numberOfMachines> [-np]
       -o <file>
Generates the workflow examples. Saves the resulting net in APT and dot
format and, if dot is executable, as pdf. This module generates the Job
Processing example of the ADAM paper.
 -h,--help                             Prints this dialog.
 -m,--nb_machines <numberOfMachines>   The desired number of machines
                                       (>= 2).
 -np,--no_partition                    If set, no automatical partition
                                       of the places is done. Thus, no
                                       annotation from token to places
                                       is printed in the resulting file
                                       in APT format.
 -o,--output <file>                    The output path where the
                                       generated Petri game should be
                                       saved.
```

Module: gen_robots

Generates the self-organizing robots examples. Saves the resulting net in APT and dot format and, if dot is executable, as pdf. This module generates the Self-reconfiguring Robots example of the ADAM paper. The help dialogue:

```
usage: java -jar adam.jar gen_robots -d <numberOfDestroyPhases> [-h]
       [-np] -o <file> -r <numberOfRobotsAndTools>
Generates the self-organizing robots examples. Saves the resulting net
in APT and dot format and, if dot is executable, as pdf. This module
generates the Self-reconfiguring Robots example of the ADAM paper.
 -d,--nb_destroy <numberOfDestroyPhases>   The desired number of phases
                                           to destroy tools (>= 0). In
                                           every phase one tool will be
                                           destroyed.
 -h,--help                                 Prints this dialog.
 -np,--no_partition                        If set, no automatical
                                           partition of the places is
                                           done. Thus, no annotation
                                           from token to places is
                                           printed in the resulting file
                                           in APT format.
 -o,--output <file>                           The output path where the
                                           generated Petri game should
                                           be saved.
 -r,--nb_robots <numberOfRobotsAndTools>      The desired number of robots
                                           and tools to use (>= 2).
```

Module: gen_workflow2

Generates the workflow2 examples. Saves the resulting net in APT and dot format and, if dot is executable, as pdf. This module generates the Concurrent Machines example of the ADAM paper. The help dialogue:

```
usage: java -jar adam.jar gen_workflow2 [-h] -m <numberOfMachines> [-np]
       -o <file> -w <numberOfWorkpieces>
Generates the workflow2 examples. Saves the resulting net in APT and dot
format and, if dot is executable, as pdf. This module generates the
Concurrent Machines example of the ADAM paper.
 -h,--help                                    Prints this dialog.
 -m,--nb_machines <numberOfMachines>          The desired number of
                                              concurrent machines (>= 2).
 -np,--no_partition                           If set, no automatical
                                              partition of the places is
                                              done. Thus, no annotation
                                              from token to places is
                                              printed in the resulting file
                                              in APT format.
 -o,--output <file>                           The output path where the
                                              generated Petri game should
                                              be saved.
 -w,--nb_workpieces <numberOfWorkpieces>      The desired number of
                                              workpieces to produce (>= 1).
```