SESA
model-analyser

Paul Berthold

Humboldt University of Berlin

2002

SESA Tool Description

SESA is our tool for the analysis of signal-net systems. Typical properties which can be verified are boundedness of places, liveness of transitions, and reachability of markings or states. General properties can be expressed in CTL and verified by the model checker of SESA. To reduce the size of the state space and the time for its construction, SESA offers several reduction methods. SESA also derives some analysis results from the underlying Petri net of a signal-net system. SESA inherited much of its code from the Petri net tool INA [RS98].

How to run SESA

To start the program, open a command shell, change to the directory which contains the net to be analysed, and into which the files to be constructed should be stored. Then enter `SESA.exe` at the shell prompt. Up to this version of SESA, there is no graphical user interface available. All output is displayed on the terminal of the command shell, where `SESA.exe` was started, and all input has to be typed in by the keyboard. In this section, we denote the typing of the key X by `<X>`. If SESA asks you for an output file name and you type the escape key `<esc>`, the output will be displayed on the terminal. On the other hand, if you type `<esc>` at an input file prompt, SESA expects your input from the keyboard.

If the file `OPTIONS.sna` is in the working directory, it will be read. On the other hand, the file `COMMAND.sna` will only be executed if you answer the question `same procedure as last time?` (which then appears at the beginning) with `<Y>`. In this case, the stored commands of the last session are repeated until `<H>` (for halt) is entered, or all commands have been executed. If there is no `COMMAND.sna`, or the question above has been answered with `<N>`, the main menu appears on the screen and the selected options are displayed:

```
>>>>>>>>>>>  Welcome to the Signal Net Analyzer  S E S A  <<<<<<<<<<<
Version 1.x                   Humboldt-University Berlin xx xyz 2002"

Current net options are:
    token  type: black
    time option: no times
    firing rule: arbitrary maximal steps
    synchros   : not to be used
    greediness : not to be used
    priorities : not to be used
    reductions : not to be used

Do You want to
    edit ? ......................................E
    fire ? ......................................F
    analyse ? ...................................A
    read the session report ? ...................S
    delete the session report ? .................D
    change options ? ............................O
    quit ? ......................................Q
```

```
choice >
```

By pressing the indicated letters the corresponding functions are selected. To stop a command, or quit a menu, `<Q>` can be entered in most cases. At many points this will also cancel running computations prematurely.

During a session, the program writes all analysis results and deductions into the file `SESSION.sna`. In the main menu, two menu items are offered, one to display the session report, and the other to erase it.

Options

After the start of SESA the current net options are displayed in the main menu. By entering `<O>`, these can be changed. There, the token type, the time option, the firing rule, the use of synchro sets, the use of greedy transitions, the use of priorities, the use of reductions, and line length are requested subsequently. The selected options are saved in the file `OPTIONS.sna`. The default net options of SESA can also be changed via the command line options at the start of the program.
In the following list, all possible options are individually presented:

token types With this option, you can determine whether the tokens are coloured.

```
<B>        black (indistinguishable) tokens
```

Hereby, you choose to work with (uncoloured) signal-net systems, whose markings consists of black indistinguishable tokens. The command line option therefore is `-black`.

```
<C>        coloured tokens
```

With this selection, you will work with coloured nets, which allows to work with coloured, i.e. distinguishable tokens. The command line option therefore is `-colour`. For more information about coloured nets, please, confront the section 3 on page 11.

If you have set the token type of a coloured net to black, SESA will ask: `Forget the colour structure?`; with `<Y>`, the colours are deleted. Warning: Information about the net can get lost this way!

If, on the other hand, you change the token type in the opposite way, the question appears: `Fold the Net?` If your answer is `<Y>`, you have to indicate how SESA should fold it. You can choose a maximal folding, a user-defined folding or no folding.

time option Here you can choose a clocked net type:

```
<N>        no time constraints
```

With this selection, no clocks will be used. The command line option therefore is `-notimes`.

```
<A>        arc timed
```

A time interval can be assigned to the input arcs of transitions. The command line option therefore is `-arctimed`. For more information about arc timed nets, refer to section 2 on page 8.

firing rule Here you can choose between different firing rules:

```
<N>        normal:  arbitrary maximal steps
```

That is the default firing rule. The executable steps under this rule are formed by first picking up a nonempty set of enabled spontaneous transitions and then adding as many as possible of those transitions that are forced to fire by signal-events produced by transitions in the step. The command line option for this rule is `-maximal`.

```
<S>        maximal single spontaneous transition steps
```

With this firing rule all steps will disappear from the step list which contain more than one spontaneous transition. The command line option for this rule is `-single`.

synchronisation sets With this option, you can determine the use of synchronisation sets. The transitions in the same synchronisation set should fire simultaneously as much as possible. The synchro option can be set only under the normal firing rule. The command line options for synchronisation sets are `-sync` and `-nosync`.

greedy transitions With this option enabled, only steps containing at least one greedy (spontaneous) transition are executed. If at the current state no greedy transitions are enabled, then the other steps are executed too. The greediness option can not be set under the single firing rule. The command line options for greedy transitions are `-greedy` and `-nogreedy`.

priorities Under the priority option only the spontaneous transitions with the greatest occurring priority are enabled. The command line options for priorities are `-priorities` and `-nopriorities`.

reductions The firing rule can also be influenced by the command line options `-diamond`, `-stubborn` and `-symmetric`. With the option `-diamond` the list of enabled steps under the normal firing rule is reduced with respect to diamonds. This means, that steps will be deleted from the step list which can be safely omitted without missing reachable markings (because the diamond property holds). For details about this reduction, please, confront the section 7 on page 21. The option `-stubborn` turns on the stubborn set reduction for the state space analysis. For a description of this reduction, see section 8 on page 27. For the firing rule maximal single spontaneous transition steps you can use `-noapprox` to select the non-approximative computation of stubborn sets. With the option `-symmetric`, SESA uses the symmetries of a signal-net system for the state space analysis. The symmetries are computed on demand. For details about symmetric reduction, see section 9 on page 37. More explanations of the firing rules, synchronisation sets, greedy transitions and priorities can be found in section 1.2 on page 5ff.

line length With this option, you can determine the length of the output lines to fit your terminal.

At the command line, it is also possible to change the file names of the session report, the command file and the options file. You can specify these names by the command line options `-session <sessfile>`, `-cmd <cmdfile>` and `-opt <optfile>`. With the commandline option `-prefix <prefix>` you can add a prefix for each of these file names. This is useful for running more than one instance of SESA in the same directory.

With the command line options `-noopt` and `-nocmd` you can prevent the processing of the files `OPTIONS.sna` and `COMMAND.sna` at the start of a session. The command line option `-reset` sets all options back to their default value. The order of command line options matters.

If you want the names of transitions and places to be displayed at the terminal and written in the session report, you can specify the option `-names` at the commandline, otherwise specify `-nonames`.

At the end of a command line, you can specify the file name of the net you want to analyse. With `-help` you get a short list of all possible command line options before the program starts:

```
SESA command line options summary

 -black   -colour
 -notimes -arctimed
 -maximal -single
 -[no]priorities
 -[no]greedy
 -[no]sync
 -stubborn -diamond -symmetric
 -[no]names
 -[no]opt <optfile>
 -[no]cmd <cmdfile>
 -session <sessfile>
 -prefix <prefix>
 -reset   -help
 <filename>
```

The net editor

By pressing `<E>` in the main menu, the menu of the editor is shown:

```
Do You want to
     Quit the editing process.....................Q
```

```
          execute Input operations.....................I
          execute Output operations....................O
          Change something in the current net..........C
          Delete something in the current net..........D
          check the EFC property.......................E
          Search for signal circuits...................S
          Test connectedness of the current net........T
          decompose or Merge...........................M
     edit>
```

If no net is loaded, then the edit menu is substituted by the file input menu, which appears also by pressing `<I>` in the editor.

In the file input menu, nets can be entered with the command `<T>`. Besides, if a net is requested as a file, you can switch to the terminal mode with `<esc>` and enter the net as described in the following.

First of all, the net number (default value: 0) and the net name (16 characters maximum) are requested. It is recommendable to fill out both, because the net number and name appear at many points in the protocol and in the saved files, and may therefore help to prevent confusion.

In the case of a uncoloured net SESA expects a list of places with their pre and post arcs next. First, SESA asks for the number of the place by the prompt `place nr.` and after typing in this number you can specify the `Token load (default = 0)` of that place. Then SESA asks for the numbers of the pre-transitions by the prompt `from transition nr.`, which can be stopped by pressing `<Q>` at the prompt. After that, the numbers of the post-transitions are expected by the prompt `to transition nr.`, which can also be stopped by pressing `<Q>`. Then SESA asks for the number of the next place, and so on... You can stop the input of the list of places by pressing `<Q>` at the `place nr.` prompt.

Because there can be isolated transitions in signal-net systems, the next question of SESA is: `Give the numbers of transitions without pre- and post-arcs`. The input of the numbers of these transitions can also be stopped by pressing `<Q>`.

After that, you have to type in the names for all the places and transitions you introduced before. After the name of each transition, SESA asks also for the numbers of places, which have a condition to that transition and the numbers of transitions, which have a signal to that transition. The input of these numbers can be stopped by pressing `<Q>`.

The input of a coloured net is a little bit more complicated. First, all places and their colours and token load have to be typed in. For each place and each colour you have to supply a name. Then SESA asks for the transitions and their colours. After the input of the names for the colours for a transition, you have to supply the pre- and post-arcs for this transition. Unfortunately conditions and signals cannot be typed in. But you can insert such arcs into a coloured net by using the change menu.

When you read a uncoloured net from a file by pressing `<F>` in the input file menu, the file must be accepted by the following grammar in EBNF:

```
        <netfile> ::= "P   M   PRE,POST  NET " <nr> ":" <name> "<cr>"
```

```
                    <flowarcs>
                    "@<cr>"
                    "pl-nr. name             icp<cr>"
                    <places>
                    "@<cr>"
                    "<cr>"
                    "tr-nr. name             pri md conditions; signals;<cr>"
                    <transitions>
                    "@<cr>"
     <flowarcs> ::= { <nr> " " <tokens> " "
                      [ <prelist> ] [ "," <postlist> ] "<cr>" }
      <prelist> ::= { <nr> [ ":" <mult> ] " " }
     <postlist> ::= { <nr> [ ":" <mult> ] " " }
       <places> ::= { <nr> ": " <name> " " <icp> "<cr>" }
  <transitions> ::= { <nr> ": " <name> " " <priority> " " <modus> " "
                      [ <conditions> ] ";" [ <signals> ] ";<cr>" }
   <conditions> ::= { <nr> [ ":" <mult> ] " " }
      <signals> ::= { <nr> " " }
```

The following grammar describes the format of coloured net files:

```
   <colnetfile> ::= <netfile>
                    "AGGREGATION:<cr>"
                    "places:<cr>"
                    <plcolours>
                    "@<cr>"
                    "transitions:<cr>"
                    <trcolours>
                    "@<cr>"
    <plcolours> ::= { <nr> ":" <name> " " { <nr> " " } "<cr>" }
    <trcolours> ::= { <nr> ":" <name> " " { <nr> " " } "<cr>" }
```

After loading or typing in a net under the input file menu, you can perform output operations (by pressing `<O>`) or change the structure of the net (by pressing `<C>` or `<D>`) in the edit menu.

There are also test functions for some structural properties of the current net:

`<E>`	`check the EFC property`

With this function, you can check the EFC property of a signal-net system. For the definition of the EFC property see section 15 on page 72.

`<S>`	`Search for signal circuits`

This function searches for circuits in the signal relation of a net. In most cases, a signal circuit in a net is the result of an input error.

`<T>`	`Test connectedness of the current net`

With this function, you can test the connectedness of the nodes of a net with respect to the flow relation, the condition relation and/or the signal relation. If the net consists of more than one component, then you can write these components to separate files.

```
<M>        decompose or Merge
```

This function offers to double a node or to merge two nodes or two nets. Further, you can decompose a net into its elementary modules. For composition of modules see section 16 on page 79.

The simulator

By pressing `<F>` in the main menu, the program changes into the simulation mode. At the beginning and after each operation in this mode the current marking and a list of its executable steps are shown. Each step has its own number, so you can fire this step by typing in this number. If you want to cancel the execution of the last step, then press `<b>`. By pressing `<r>`, you reset the current marking to the initial marking.

If you want to see the stubborn set used by the stubborn reduced reachability graph, then press `<s>`. Press `<c>`, if you want to see the construction of this set. For more information about the stubborn set reduction, see section 8 on page 27. For the firing rule maximal single spontaneous transition steps you can toogle between the approximative and the non-approximative computation of stubborn sets by pressing `<a>`.

To see the step list reduced with respect to diamonds, press `<d>`. For details about the diamond reduction, see section 7 on page 21.

You can write the current marking into a `.mar` file by pressing `<w>`. To leave the simulation mode and return to the main menu enter `<q>`.

Analysing signal-net systems

By pressing `<A>` in the main menu, you enter the most important menu of SESA the analysis menu. In the analysis menu, different analysis procedures are offered depending on the net type and the status of the analysis. Only those procedures are offered which can lead to statements about the important dynamic properties. To return to the main menu, enter `<Q>`.

```
Analysis menu:

Non-reachability test of a partial marking using the state equation........N
Compute a minimal path from the initial state to satisfy a predicate.......P
Compute a minimal path from the initial state to a (sub-)marking...........O
Check a CTL-formula........................................................F
Compute a reachability graph...............................................R

Compute the symmetries of the net..........................................Y
Define a concession predicate..............................................D
```

```
For the underlying Petri net (signal arcs ignored):
   Decide structural boundedness.....................................S
   Decide boundedness................................................B
   Compute a base for all S/T-invariants [non-reachability test]...........I

choice >
```

Before the analysis menu is displayed, SESA executes a pre-analysis of the net, which investigates structural properties, and also checks which functions are available for the given net. At the beginning, you can set writing options. These include, for example, the output option for static conflicts: `Print the static conflicts?`

The progress of the analysis is indicated in a status line above the Analysis menu:

```
SCV SCF FtO tFO FpO pFO CPI CTI  B   SB  REV DSt BSt DTr DCF  L   LV L&B WL  CL
 Y   N   Y   N   N   N   ?   ?   Y   Y   ?   ?   ?   ?   ?   ?   ?   ?   ?   ?
```

The possible properties of the current net are listed. The symbol below each property indicates whether the property is fulfilled (`Y`), not fulfilled (`N`), or no decision could have been made yet (`?`). For an explanation of each property see the section on page 116.

In the following list, all possible functions of the analysis menu are presented:

```
<N>        Non-reachability test of a partial marking using the state
           equation
```

This function can decide the non-reachability of a marking using the state equation. A partial marking is sufficient here: the marking of the remaining places is considered to be not defined.

```
<O>        Compute a minimal path from the initial state to a
           (sub-)marking
```

A marking is tested for reachability, and, if possible, a path is displayed. The path is minimal with respect to the length (i.e. number of firing procedures) or the cost (the value of a path is the sum of the values of the fired transitions). Here, a partial marking is sufficient. The marking of the remaining places is considered as not defined.

The marking to be examined can be read from a file or, by pressing `<esc>`, entered directly.

For a net with time allocation, in addition to the computation of a minimal path with respect to length or cost, it is also possible to compute a path which is minimal with respect to time (i.e. a fastest one). Time allocation will be inquired anyway.

After entering (or reading) the target marking, you can enable some reduction methods for the construction of the state space. You can use the stubborn set reduction or the diamond reduction. In combination with the normal firing rule you will only get an upper bound of the minimal length, due to reducible steps.

In all other cases (single spontaneous transition steps or minimal values) you will get exact results. Reachability is preserved by both reductions. For details refer to section 7 on page 21 and section 8 on page 27.

It is also possible to cut of the construction of the state graph at states, which satisfy a defined "bad" predicate.

Then, SESA starts to construct the state graph of the net breadth first. The number of states so far computed is displayed: `States generated`.

If SESA encounters a marking that agrees with the target marking on the places defined, the examination is cancelled. The target marking is reported on the screen as reachable. The path can be written into a separate file with the extension `.tra` or in the session report.

If the target marking is not reachable, there are two cases: if the entire state graph can be constructed and saved, the marking will be regarded as unreachable (`The marking is not reachable`); otherwise (always in unbounded nets), SESA cancels with the message `Node overflow` and states that no decision can be made: `No decision possible.`

By pressing `<Q>`, the computation process can be cancelled at any time. Any dead states which may have been found are taken into account in the further analysis.

`<P>`	`Compute a minimal path from the initial state to satisfy a predicate`

This function computes a minimal path to a state which satisfies a previously specified state predicate. Otherwise, this function is similar to the one with `<O>`.

`<R>`	`Compute a reachability graph`

This command constructs the state graph of a signal-net system. For the construction of the state space, you can enable some reduction methods: You can use the symmetrical reduction, the stubborn set reduction or the diamond reduction. For these reductions refer to section 7 on page 21, section 8 on page 27 and section 9 on page 37. It is also possible restrict the depth of the state graph and to cut of the construction of the state graph at states, which satisfy a defined "bad" predicate.

Then, SESA starts to construct the state graph of the net depth first. The number of states so far computed is displayed: `States generated`.

Subsequently, different graph analyses can be executed, CTL-formulae and predicates can be created or checked, and certain results can be written either into separate files or in the session report. For more information about graph analysis in SESA see the section on page 113.

`<F>`	`Check a CTL-formula`

With this function, you can check CTL-formulae, i.e. determine their validity and generate proofs. For more information about checking CTL-formulae in SESA see the section on page 115.

`<Y>        Compute the symmetries of the net`

When computing the symmetry group of the current net, SESA considers possible time allocations.

First, you have to state whether fixed points should be set for places and transitions, or whether the initial state should be considered as symmetric: `Do you want to set fixpoints?` or `Initial state to be symmetric?`

Sometimes, other symmetries or none at all are found in this way, and the computation is cancelled: `Trivial transition partition!`

By answering the question `Write the symmetries to the session file?` with `<Y>`, the generators of the symmetry group are written into the session-report. With `<N>`, only the decompositions of the place and transition sets into equivalence classes are written.

During the computation, a counter records the generators found; the running computation can be aborted with `<Q>`.

At the end of the computation, the number of generators and the number of symmetries, which can be obtained by combining them, are displayed.

The function `<Y>` can also be used to have a (already once computed) symmetry group be re-computed. In order to do this, you only have to answer the question `Compute the symmetries once again?` with `<Y>`. For example, you can set new fixed points, or consider the initial marking.

`<D>        Define a concession predicate`

With this function, a predicate is defined based on a transition set to be specified; this predicate is satisfied exactly by those states in which at least one transition of the set is enabled.

`<S>        Decide structural boundedness`

This function decides whether the net is covered by P-invariants. If this is the case, it is structurally bounded, i.e. bounded in every initial marking.

`<B>        Decide boundedness`

This function decides the boundedness of a net by the computation of the coverability graph of the underlying Petri net. SESA computes the graph using the algorithm of Karp and Miller. In case the net is bounded, the coverability graph corresponds to the usual state graph of the underlying Petri net.

```
<I>       Compute a base for all S/T-invariants [non-reachability
          test]
```

This command computes a basis for the space of all invariants of the selected type. SESA states whether invariants were found, and possibly derives further deductions from it. The program decides whether the net is covered by invariants of the selected type, i.e. whether an invariant exists which is positive in all components. If such an invariant was actually computed, it will be displayed as well. If P-invariants were found, a fast non-reachability test is offered.

Further analysis of the reachability graph

After the (incomplete) construction of the reachability graph or after the model checking of a CTL-formula which led to a complete reachability graph, you enter the graph analysis menu. In this menu different graph analyses can be executed, CTL-formulae and predicates can be created or checked, and certain results can be written either into separate files or in the session report:

```
Graph analysis menu
Do You want to
     quit analysis of the computed graph ................ Q

     test the reachability/coverability of a marking ..... R
     convert a set of states to a predicate ............. C
     define a concession predicate ...................... E
     check a CTL-formula ................................ F
     compute distances .................................. A
     compute circuits ................................... K
     check liveness properties .......................... L
     compute strongly connected components .............. V
     check dynamic conflicts ............................ Y
     check for false diamonds ........................... U
     write the computed graph (states and arcs) .......... W
     write all arcs ..................................... X
     write all states ................................... M
     write all states satisfying a predicate ............ P
     write all states with a given successor ............ G
     write the dead states .............................. D
     write the bad states ............................... B
     write a trace to a state ........................... T
     write the list of executed steps ................... S
     inspect a result file .............................. I
choice>
```

Warning: If you leave this menu by pressing `<Q>` the memory of the reachability graph is freed. So, you have to construct the graph again, if you want to perform more graph analysis.
In the following list, all possible functions of the graph analysis menu are presented:

```
<R>        test the reachability/coverability of a marking
```

With this function, you can execute reachability or coverability tests, and find (not necessarily minimal) paths from the initial marking to a target marking.

```
<C>        convert a set of states to a predicate
```

With this command, a predicate for a set of states is defined which is satisfied exactly by these given states. You can construct the predicate with respect to all computed states, the dead states, states where a set of transitions is fireable, or a set to be specified by state numbers. The predicate defined can be saved in a file with the extension `.pdc`.

```
<E>        define a concession predicate
```

With this function, a predicate is defined based on a transition set to be specified; this predicate is satisfied exactly by those states in which at least one transition of the set is enabled.

```
<F>        check a CTL-formula
```

With this function, you can check CTL-formulae, i.e. determine their validity and generate proofs. For more information about checking CTL-formulae in SESA see the section on page 115.

```
<A>        compute distances
```

With this command, minimal and maximal distances between nodes of the state graph can be computed. The results are written into the session report.

```
<K>        compute circuits
```

With this command, circuits in the state graph can be computed and evaluated.

```
<L>        check liveness properties
```

With this command, a liveness analysis can be executed. This works only for completely computed state graphs without stubborn reduction.

If the net contains transitions which are dead in the initial state, then the liveness analysis is restricted to the transitions that are not dead. The net can be live if all dead transitions are considered as facts; the property `LV` (Liveness when ignoring dead transitions) is then set accordingly. Further weaker notions of liveness are explained on the screen, if necessary.

```
<V>        compute strongly connected components
```

With this function, the strongly connected components are computed, and, upon request, written into a file with the extension `.res`.

```
<Y>          check dynamic conflicts
```

With this function, you can search the set of reachable states for dynamic conflicts (see in section 10 on page 44).

```
<U>          check for false diamonds
```

This function looks for reachable states where the diamond property does not hold (see section 7 on page 21).

```
<...>        write ...
```

With these commands, the computed graph, or parts of it, are written into the session report, or, upon request, into a file. By entering `<esc>` on the file name prompt, you can redirect the output to the screen. In most cases, a selection of states can be defined.

```
<I>          inspect a result file
```

With this command, you can inspect different files created during the analysis.

CTL model checking

Testing the validity of Computation Tree Logic CTL-formulae in the initial state is called model checking. Thereby, witnesses for existence-quantified sub-formulae, and counter examples for all-quantified sub-formulae can be determined and displayed. See section 11ff for details about CTL and the timed and extended Computation Tree Logic.

At the `formula input file` prompt, you can then read a `.ctl` file, or, by pressing `<esc>`, enter a formula directly. A very short explanation of the CTL syntax is displayed when entering formulae by hand:

```
Syntax:

 Boolean combination: NOT f, f1 AND f2, f1 OR f2, f1 IMPL f2, f1 EQUIV f2
 Temporal operators : EX f, EF f, EG f, E[f1 U f2], E[f1 B f2]
                      AX f, AF f, AG f, A[f1 U f2], A[f1 B f2]
 Predicates         : disjunction of conjunctions of interval specifications
                      use P and a number or a file name (*.pdc with quotes)
                      to refer to a predicate e.g  P1  or  P"pred1.pdc"
 Atomic propositions: references to the marking of a place by number or name
                      and comparisons > < = <> >= <= of markings and marking
                      sums e.g.  p1   m(p1)=0  or   m(p1)+m(p2)=1
                      for reference by name use quotes e.g.  m(p"end")
                      a reference without m( ) is interpreted as  m( )>0
 Transition formulae: boolean combinations of references to transitions
                      attached to the quantifiers E and A to limit the
                      range of temporal operators e.g.  E t1 F m(p1)
                      or  A (t"one" OR t"two") G P"invar.pdc"

Please, type the formula to check:
```

A complete graphical diagram of the syntax of CTL used in SESA is shown in Fig. A.1 and A.2. Each nonterminal of the language is explained by a little diagram. The name of the nonterminal is shown in boldface on the top left corner of this diagram. Below this name, there is the entry point of the syntax diagram for the nonterminal. The boxes of the diagrams represent other nonterminals. Their names are written in the boxes. On the other hand, the circles represent the terminals. The symbols of these terminals are shown in the circles. To check the correctness of a sentence, start with the top left nonterminal "ctl" and then follow the arrows from the left hand side through the boxes and circles to the right end of the diagrams.

After typing in the formula and afterwards answering the question `ok?` with `<Y>`, you can write the formula to a file with the extension `.ctl`. Thereafter, SESA is parsing the formula and the predicates whose numbers were mentioned in the formula are requested. You can either read previously defined predicates from a file, or, by pressing `<esc>`, enter predicates directly. Before the computation is started, you can specify some output options for the proof of the formula. The progress of the computation is displayed by the number of generated states. The computation can be cancelled by pressing `<Q>`. After a successful computation, the value of the formula is displayed. SESA warns with `TRUE/FALSE in the computed subgraph` if a reduced or incomplete graph was generated and the truth value of a formula in the complete graph is not deducible (for reduction techniques see section 7 on page 21, section 8 on page 27 and section 9 on page 37). If the generated graph is complete, you enter the graph analysis menu, see the section on page 113. Otherwise, you can write the incomplete graph or check the next formula.

Properties

In the following list, all properties of the status line in the analysis menu and the graph analysis menu of SESA are explained:

`SCV` **subconservative**

A net is sub-conservative, if all transitions add at most as many tokens to their post-places as they subtract from their pre-places. The total number of tokens can therefore not increase.

`SCF` **statically conflict-free**

If two transitions have a common pre-place, they are in a static conflict about the tokens on this pre-place. Then, the net is not static conflict free.

Further information can be found in section 10 on page 44ff.

`Ft0` **transition without pre-place**

A net has $Ft0$-transitions, if there are spontaneous transitions without a pre-place and without a condition but with a post-place; $St = \emptyset$, $Ft = \emptyset$, $Cond(t) = \emptyset$, $tF \neq \emptyset$.

`tF0` **transition without post-place**

A net has $tF0$-transitions, if there are spontaneous transitions without a post-place and without any signal to another transition but with a pre-place; $St = \emptyset$, $tF = \emptyset$, $tS = \emptyset$, $Ft \neq \emptyset$.

`Fp0` **place without pre-transition**

A net has $Fp0$-places, if there are places without a pre-transition but with a post-transition; $Fp = \emptyset$, $pF \neq \emptyset$.

`pF0` **place without post-transition**

A net has $pF0$-places, if there are places without a post-transition but with a pre-transition; $pF = \emptyset$, $Fp \neq \emptyset$.

`CPI` **covered by place invariants**

A net is covered by place invariants, if there exists a P-invariant which assigns a positive value to each place. If this is the case, the net is structurally bounded, i.e. bounded under any initial marking.

`CTI` **covered by transition invariants**

A net is covered by transition invariants, if there exists a T-invariant which assigns a positive value to each transition.

Further information can be found in section 19 on page 95ff.

`B` **bounded**

A net is bounded, if there is a number k such that, in any reachable marking, there are never more than k tokens on a place.

`SB` **structurally bounded**

A net is structurally bounded, if it is bounded in every initial marking.

`REV` **reversible**

A net is reversible, if the initial state can be reached from every reachable state.

`DSt` **dead state reachable**

A dead state is reachable, if a state is reachable in which no transition can fire any more.

`BSt` **bad state reachable**

If a state satisfies a so-called "bad" predicate, it is not further developed when computing a state graph. In this case, the attribute `Bst` is set. However, after leaving the graph analysis, this attribute is reset to `?`.

`DTr` **dead transition exists (at the initial marking)**

This attribute indicates whether the net has dead transitions in the initial marking, i.e. facts.

`DCF` **dynamically conflictfree**

A net is said to be dynamically conflict free, if no state is reachable in which a step conflict or a transition conflict occurs.

Further information can be found in section 10 on page 44ff.

`L` **live**

A net is live, if all its transitions are live in the initial marking, i.e. no state is reachable in which a transition is dead.

`LV` **live if dead transitions ignored**

A net is live when ignoring dead transitions, if all its transitions, which are not already dead in the initial marking, are live. The transitions thereby ignored can be considered as unspecified facts.

`L&B` **live and bounded**

A net is live and bounded, if it is live and, if there is a number k such that, in any reachable marking, there are never more than k tokens on a place.

`WL` **weakly live**

A coloured net is weakly live, if all its transitions are weakly live, i.e. for each transition, there is a colour in which the transition is live in the initial marking.

`CL` **collectively live**

A coloured net is collectively live, if all its transitions are collectively live. A transition is collectively live, if for every reachable state a colour exists, such that in a state reachable from this marking, the transition can fire in this colour. In particular, every weakly live transition is also collectively live.