Cabernet User Manual Contents

Cabernet (Computer Aided software engineering environment Based on ER NETs) is an environment that supports the specification of real time systems. It is based on ER nets, Petri nets augmented with data, functionalities and time. It provides basic functionalities for editing and storing ER nets. It includes facilities for validating the specifications, in particular for executing, simulating, and animating the sepcifications an for formally proving temporal properties. Additional facilities allows formal refinement of ER nets and customization of the set of functionalities provided by the environment.

Menu

Acknowledgements

Cabernet was developed at CEFRIEL and Politecnico di Milano by a group of researchers and graduate students coordinated by Mauro Pezze.

Major contributions and strong support to the Cabernet project have been given by Carlo Ghezzi, Miguel Felder and Carlo Bellettini.

Software production managed by Miguel Felder and Carlo Bellettini.

Implementation: Marco Braga, Paolo Brunasti, Paola Cherubini, Istok Fekinja, Stefano Gangai, Fabio Lameri, Marco Piantanida, Joze Strucl, Serena Manca, Roberto Palmer, Marco Piantanida, Kim Portman, Michel Sfondrini, Roberto Zambetti and Marco Zoccolante

Windows version: Mauro Cavagna, Luca Del Bon and Claudio Valoti

Manual: Sergio Silva

Bibliography

[BFP93]

C. Bellettini, M. Felder and M. Pezze'. Merlot: A tool for analysis of real-time specifications. In **Proceedings of the 7th International Workshop on Software Specification and Design**, Redondo Beach (CA), December 1993.

[FGP93]

M. Felder, C. Ghezzi and M. Pezze'. Analyzing refinements of state based specifications: the case of TB nets. In **Proceedings of International Symposium on Software Testing and Analysis**, Boston (MA), June 1993.

[GMMP91]

C. Ghezzi, D. Mandrioli, S. Morasca and M. Pezze'. A unified high level Petri net formalism for time-chritical systems. **IEEE Transactions on Software Engineering**, 17(2), February 1991.

[GMP94]

C. Ghezzi, S. Morasca and M. Pezze'. Timing analysis of time basic nets.

Journal of systems and software, To appear 1994.

[GP93]

C. Ghezzi, M. Pezze'. Towards extensible graphical formalism. In **Proceedings of International Symposium on Software Testing and Analysis**, Boston (MA), June 1993.

[Mur89]

T. Murata. Petri nets: properties, analysis and applications. In **Proceedings of IEEE**, Vol. 77, April 1989.

[PG92]

M. Pezze', C. Ghezzi. Cabernet: an environment for the specification and verification of real-time systems. In **Proceedings of 1992 DECUS Europe Symposium**}, Cannes (F), September 1992.

[Pez94]

M. Pezze'. Cabernet: a customizable environment for the specification and verification of real-

time systems. submitted for publication. 1994.

[Str92]
Stroustrup, Bjarne. **The C++ programming language**. Addisson-Wesley, 1992.

–

Tool generator elementary tools

In the following list, input parameters are indicated by **:I** and output by **:O**. Parameters types are:

Net: Any net.
Set: Set of objects.
KString: String.
TimedEnabling: Time enabling.
TimedNet: Net with time properties.
NKobject: Any Cabernet object.
Transition: Transition node.
Place: Place node.
Arc: Arc nodes.

Menu

General tools

evaluatePredicates(Net:I,Set:I,Set:O)

Verifies the predicates on the input tuples (**Set:I**) and returns the effective enabling tuples in **Set:O**.

chooseEnabling(Set:I,TimedEnabling:O,KString:I,TimedNet:I)

Chooses a transition **TimedEnabling** from the firable ones (**Set**).

SetKString indicates the choosing mode (Execution options). Note that this works for timed nets as indicated by the last input parameter **TimedNet**.

chooseFiring (TimedNet:I,TimedEnabling:I,KString:I,KString:I,KString:I,KString:O)

Returns a firing time in **KString**. Input parameters are, the timed net **TimedNet**, timed enablings in **TimedEnabling** and in that order: current time, next deadline (minimum of maximum firing times) and choosingmode (Execution options), each in a **KString**.

executeAction(TimedNet:I,TimedEnabling:I,TimedEnabling:O,KString:I)

Given a **TimedNet**, a **TimedEnabling:I** and a firing time in **KString**, Evaluates the action of the chosen transition, produced tokens are left in **TimedEnabling:O**

fire (TimedNet:I,TimedEnabling:I,TimedEnabling:I,KString:I,KString:I,TimedNet:O)

Updates the input net **TimedNet:I** to the state it must have after firing the chosen transition, the updated net is left in **TimedNet:O**. Other input parameters are: chosen enabling and tokens to produce after the firing, each in a **TimedEnabling**; current and firing times, each in a **KString**.

Note that to execute the firing, it is necesary to know the produced tokens. i.e. prior to its execution, it is necesary to execute the **executeAction** tool.

getNet(Net:O)

Presents a dialog box similar to the one used in the File menu (File)to select a file to be loaded into net Net.

loadNet(KString:I,Net:O)

Loads a net which name is given in **KString** into the net **Net**.

duplicate(NKobject:I,NKobject:O)

Given any object creates a copy of it.

inc(KString:I)

Increments the value of the input, which must be a string representing a numeric value.

decr(KString:I)

Decrements the value of the input, which must be a string representing a numeric value.

Sets

getFromSet(Set:I,NKobject:O)

Extracts an object which is left in **NKobject** from a given set **set**. After the operation the set is the same but without the extracted element. The extracted object is the last added object.

addToSet(Set:I,NKobject:I,Set:O)

Given an input set and an object, creates an new set into **NKobject:O** which is the result of including the object in the original set.

removeFromSet(Set:I,NKobject:I,Set:O)

Given a set **Set:I** and an object **NKobject**, generates another set in **Set:O** which is the set relusting from extracting the specified object from the given set.

unione(Set:I,Set:I,Set:O)

Performs the union of the two input sets into **Set:O**

intersect(Set:I,Set:I,Set:O)

Puts into **Set:O** the intersection of the two input sets.

Getting properties

getCurrentTime(TimedNet:I,KString:O)

Given a timed net, returns its current time value in a **KString**.

getPresetTokens(Net:I,Transition:I,Set:O)

Given a net (**Net**) and a transition (**Transition**) in that net, builds a set (**Set**) with the tokens of the preset places.

getTokens(Place:I,Set:O)

Builds a set **Set** with the tokens of the input place **Place**.

getPresetPlaces(Transition:I,Net:I,Set:O)

Given a net **Net** and a transition **Transition** in it, builds a set **Set** with the places in the preset of the transition.

getPostsetPlaces(Transition:I,Net:I,Set:O)

Given a net **Net** and a transition **Transition** in it, builds a set **Set** with the places in the postset of the transition.

getEnabledTransition(TimedEnabling:I,Transition:O)

Given an enabling **TimedEnabling**, leaves the transition in it into **Transition**.

getTransitions(Net:I,Set:O)

Given a net **Net**, builds a set **Set** with its transitions.

getPlaces(Net:I,Set:O)

Given a net **Net**, builds a set **Set** with its places.

getType(Place:I,KString:O)

Given a place **Place**, returns its type in **KString**.

getName(Name:I,KString:O)

Given an element **Name** it returns its name as a string in **KString**.

getAction(Transition:I,KString:O)

Lets in **KString** the action string of the input transition **Transition**.

getPredicate(Transition:I,KString:O)

Leaves in **KString** the predicate string of the input transition **Transition**.

getValue(Token:I,NKobject:O)

Given a token in **Token**, leaves its value in **NKobject**

Displaying objects

displayNet(Net:I)

Dsiplays the given net **Net** on the screen.

draw(Net:I,KString:I)

Draws the given net **Net** on the screen using the color specified in **KString**. Color must be a number between 0 and 9, according to the colors selected by the user (Colors).

drawAndSetColorNet(Net:I,KString:I)

Draws the given net **Net** on the screen using the color specified in **KString** and sets this as the foreground color. Color must be a number between 0 and 9, according to the colors selected by the user (Colors).

displayNodes(Net:I,Set:I)

Given a set of nodes **Set** in the net **Net** displays them on the screen.

deleteNodes(Net:I,Set:I)

Given a set of nodes **Set** in the net **Net** hides them on the screen, i.e. nodes are **visually** deleted.

deleteTokens(Net:I,Set:I)

Given a set of places **Set** in the net **Net**, deletes the tokens from them. This is an effective deletion, not only visual.

displayTransition(Net:I,Transition:I,KString:I)

Given a transition **Transition** in the net **Net** and a color **KString**, draws the transition using the specified color. Color must be a number between 0 and 9, according to the colors selected by the user (Colors).

drawTrans(Net:I,Transition:I,KString:I)

Given a transition **Transition** in the net **Net** and a color **KString**, draws the transition using the specified color using the shape numbered as 4 in the user symbols table (Symbols). Color must be a number between 0 and 9, according to the colors selected by the user (Colors).

clearTrans(Net:I,Transition:I)

Given a transition **Transition** in the net **Net**, draws the transition using the shape numbered as 1 in the user symbols table (Symbols).

highlightTrans(Net:I,Transition:I,KString:I)

Changes color of a transition **Transition** in the net **Net** to the indicated color **KString**. On screen refresh, transition color returns to the foreground color. Color must be a number between 0 and 9, according to the colors selected by the user (Colors).

displayPlace(Net:I,Place:I,KString:I)

Given a place **Place** in the net **Net** and a color **KString**, draws the place using the specified color. Color must be a number between 0 and 9, according to the colors selected by the user (Colors).

highlightPlace(Net:I,Place:I,KString:I)

Changes color of a place **Place** in the net **Net** to the indicated color **KString**. On screen refresh, place color returns to the foreground color. Color must be a number between 0 and 9, according to the colors selected by the user (Colors).

displayArc(Net:I,Arc:I,KString:I)

Given an arc **Arc** in the net **Net** and a color **KString**, draws the arc using the specified color. Color must be a number between 0 and 9, according to the colors selected by the user (Colors).

highlightArc(Net:I,Arc:I,KString:I)

Changes color of an arc **Arc** in the net **Net** to the indicated color **KString**. On screen refresh, arc color returns to the foreground color. Color must be a number between 0 and 9, according to the colors selected by the user (Colors).

Writing messages

print(KString:I)

Prints the input string **KString** in the message area.

writeln()

Prints a newline character in the message area.

writespace()

Prints a white space in the message area.

println(KString:I)

Prints the input string plus a newline character on the **std error**.

Identifying enablings

identifyPotentialEnabligs(Net:I,Set:O)

Given a net **Net**, generates a set **Set** with all the transitions with at least one token in every place of their preset.

identifyTimingEnablings(Net:I,Set:I,KString:I,Set:O,KString:O)

Given a net **Net**, a set of potential enablings **Set:I** and the current time **KString:I**, generates a set **Set:O** with the time enablings. **KString:O** contains the deadline.

Predicates

Predicates are used to test and evaluate properties.

!

Logical negation.

emptyset(Set:I)

True if the set **Set** is empty.

equal(NKobject:I,NKobject:I)

True if the two given objects have the same value.

0<(KString:I)

True if the value in **KString** is greater than zero. The given value must be numeric.

0>(KString:I)

True if the value in **KString** is lesser than zero. The given value must be numeric.

0<=(KString:I)

True if the value in **KString** is greater than or equal to zero. The given value must be numeric.

0>=(KString:I)

True if the value in **KString** is lesser than or equal to zero. The given value must be numeric.

>(KString:I,KString:I)

True if the value in the first **KString** is greater than the one in the second. Both values must be numeric.

<(KString:I,KString:I)

True if the value in the first **KString** is lesser than the one in the second. Both values must be numeric.

>=(KString:I,KString:I)

True if the value in the first **KString** is greater than or equal to the one in the second. Both values must be numeric.

<=(KString:I,KString:I)

True if the value in the first **KString** is lesser than or equal to the one in the second. Both values must be numeric.

Getting started

How to get Cabernet, system requirements to run it, and setup procedures.

Menu

How to get Cabernet

Free copies of Cabernet can be obtained through :

Prof. M. Pezze'

POLITECNICO DI MILANO

Dipartimento di Elettronica

P.za Leonardo da Vinci 32

20133, Milano, Italia

Ph: +39 (2) 2399 3523

Fax:+39 (2) 2399 3411

e-mail pezze@elet.polimi.it

For further information, please contact Prof. M. Pezze' at the address previously given.

Requirements

To be executed, Cabernet requires:

a C++ compiler properly installed, in order to be able to compile and execute TER nets.

2 Mb main memory

2 Mb free space on hard disk

386 CPU

Windows 3.1

MS-DOS 5 or later

Cabernet setup

To install Cabernet, first verify its hardware and software requirements (Requirements). Once you have checked this, procede to execute the following steps:

Run '*Windows*'

Insert Cabernet disk into drive *A*

Select menu-item '*RUN'* from '*FILE'* Windows menu

Type "*A:\install*" and press '*Enter'* key

Tutorial

This is a draft of the tutorial, it does only include a few examples to be used in better understanding the use of Cabernet.

Most Cabernet facilities not addressed in this tutorial (e.g., how to open a file how to edit a net or how to use menus) are easily understandable by direct use.

Menu

Consumer-producer I

File

cons.net

Tool

Editor and executor

Description

This is a simple consumer-producer net. i.e., there is a place which produces tokens which are consumed by another places.

Usage

Load (Open) the file **cons.net**. When the file is loaded into a window you can try some of the executor facilities (Executor).

To execute the net, first compile it using the **Compile** option from the **Executor** menu (Making an executable net). Once the net is compiled, try some of the executor options (Execution options).

The first can be the number of firings to execute, change it and run the net.

Select the option **Stop every firing** in the **Execution options** pop up, in this way, the execution of the net will be stopped prior to every firing. After that, try selecting the **Stop every phase option**, in this manner, execution can be followed at a more detailed level (Execution options).

Select **TER Execution Mode** in the **Execution options** pop up and **User choice** in the **Firing time choice** option in the same pop up.

By doing this, you will be asked for a time value whenever it is required by Cabernet.

Power station

File

pwrstati.net

Tool

Editor and executor

Description

Electrical station. This is a big size example developed using Cabernet, at the moment it is not yet fully documented and available for distribution. You can open it to see the kind of problems that can be managed with Cabernet.

Usage

Load the file **pwrstati.net** into a Cabernet window (Open), and try some of the editor facilities (Editor). For example, try zooming in and out the net (Zoom) using the respective buttons in the toolbar.

Warning: If you want to save changes made to this or other examples, it is recommended to make a copy of the original file.

Tasks

File

task.net, task.grp

Tool

Analyzer

Description

This example deals with the case of three tasks sharing resources (processors, disk and printers) and the way they use and compete for these resources.

Usage

Load the file **task.net** into a cabernet window (Open), then create a new window and load the file **task.grp** into that window. The last is the time reachability tree produced by Cabernet to determine if there exists at least one feasible scheduling, in which the two tasks can be finished within 120 time units.

The property is indicated to Cabernet as **t1_T6&&t2_T6** and **Time limit** is set to **T0+120**.

***Warning*:** Proving some properties may take a long time. Consider this before starting properties analysis. If you are just looking at the analyzer with no special porpouse, try properties easily analyzable, e.g., those regarding the firing of transitions near the starting point.

Writing professors

Files

prof.net, prof2.net, prof.grp

Tool

Analyzer

Description

This is the net corresponding to the **writing professors** problem presented in [BFP93], this problem is similar to the one of the dining philosophers, the main difference is that in this case temporal constraints are considered.

Usage

Load the file **prof.net** into a Cabernet window, then create a new window load the file **prof.grp** into that window. The last is the TRT used to prove if professor 1 can get a pencil within a period of 30 days. The property is indicated to Cabernet as **PckPncl_4** which is the name of the transition which firing is to be verified. **Time limit** is set to **T0+60**.

File **prof2.net** is the net which results from selecting **Instantiate net** while examining the final node of the TRT (Further examination).i.e., this is a net which satisfies the condition.

Warning: Proving some properties may take a long time. Consider this before starting properties analysis. If you are just looking at the analyzer with no special porpouse, try properties easily analyzable, e.g., those regarding the firing of transitions near the starting point.

Gas burner

File

gas_burn.net, gas_burn.grp

Tool

Analyzer

Description

This example considers a gas burner and its possible states, in order to verify if it is safe.

Usage

Load the file **gas_burn.net** into a Cabernet window (Open). Create and load the file **gas_burn.grp** in another window. The last is the TRT used to verify if there is a path such that the gas concentration reaches a value of 10 or more units, within a 60 seconds time.

If the property is not satisfied, it means the system is safe. The property is indicated as **Concentration >= 10**. **Time limit** is set to **T0+60**.

Warning: Proving some properties may take a long time. Consider this before starting properties analysis. If you are just looking at the analyzer with no special porpouse, try properties easily analyzable, e.g., those regarding the firing of transitions near the starting point.

Print places

File

print_pl.net

Tool

Tool generator

Description

Tool created using the tool generator, this prints the names of all places in the target net.

Usage

Load file **print_pl.net** into a window 1 (Open) create another window and load into the second window 2 the same file. If you want to observe tool net evolution during execution return to window 1. Run the tool using the **Run tool** option from the **Tool generator** menu. Place names of the net loaded in window 2 will be printed in the message area (Graphical interface).

Reload the tool net in window 1 and try now to execute it using the **Debug tool** option from the **Tool generator** menu. Execution will pause on every step, resume it using the **GO** button.

Print transitions

File

print_tr.net

Tool

Tool generator

Description

Tool created using the tool generator, this prints the names of all transitions in the target net.

Usage

Load file **print_tr.net** into a window (Open). Run the tool using the **Run tool** option from the **Tool generator** menu. This tool performs an operation similar to print places (Print places)

the difference is that in this case, transition names are printed out instead of place names of the net loaded in window 2.

Reload the tool net in window 1 and try now to execute it using the **Debug tool** option from the **Tool generator** menu. Execution will pause on every step, resume it using the **GO** button.

Print places and transitions

File

print_pt.net

Tool

Tool generator

Description

Tool created using the tool generator, this prints the names of all places and transitions in the target net. This tool calls another tools previously created with the tool generator.

Usage

Load file **print_pt.net** into window 1 (Open) switch to window 2 using the **Window** menu (Window}) load another Cab net file. If you want to observe tool net evolution during execution return to window 1. Run the tool using the **Run tool** option from the **Tool generator** menu. This tool prints all place names and all transition names in the message area (Graphical interface).

Reload the tool net in window 1 and try now to execute it using the **Debug tool** option from the **Tool generator** menu. Execution will pause on every step, resume it using the **GO** button.

Open **Modify transition** pop up for the transition **print_places** (Transitions). In the **Transition subnet** field you will find a file name (**print_places.net**), this is the file containing the net corresponding to a previously created tool (Print places).

Consumer-producer II

File

cons_h.trn

Tool

Hierarchy manager

Description

This is the same consumer-producer example presented earlier (Consumer-producer I), in this case some refinement rules have been applied (Refinement tool).

Usage

Load the file **cons_h.trn** into a Cabernet window (Open).

Once the file is open, try going down and up in the hierarchy using the navigation facilities (Navigation tool). To go down one level in the hierarchy, use the **Go down** option in the **Hierarchy** menu. Option **Go up** in the same menu will take you up one level in the hierarchy.

Introduction

Cabernet is a software engineering environment for the specification and analysis of real time systems. It provides an integrated set of tools for specifying and analyzing specifications of real time systems based on Petri nets augmented with data, predicates, actions and temporal information. Its functionalities span from well known facilities like editing, printing and saving to innovative functionalities not yet provided by most tools available for Petri nets, e.g. automatic proof of liveness and safety properties and generation of new tools starting from basic functionalities.

Cabernet has been developed by researchers and students of Politecnico di Milano (the major engineering school in Italy) and CEFRIEL (a new research laboratory partially sponsored by Politecnico di Milano), with the aim of understanding the benefits of formal techniques for the development of real time systems.

Software engineering students are the natural users of Cabernet, through its use, they can learn how to operate with Petri nets, how to use them for the development of medium to large size software specifications, and how to solve problems related to the development of real time systems. After the successful use of Cabernet within our institution for the training of few hundreds of students, Cabernet has been distributed to other research and training institutions (almost 30 in the first six months). Its performances among students and its user friendliness make us hope that Cabernet can also be used by system engineers for the specification of real systems, at least as a prototype for evaluating the approach.

Cabernet provides a rich set of tools comprising: graphical editor, executor and animator, analyzer, hierarchy manager and tool generator. Additional tools are under development for further enhance Cabernet use.

The graphical interface is particularly easy to use: all major graphical editing facilities are provided: copy, paste, select, cut, modify, gridding, printing, zooming. The executor and animator allows the evolution of specification to be studied by direct inspection.
Cabernet has shown that execution and animation can greatly help in revealing faults in the early stage of development. In this way such faults can be removed with great benefits on the final software. The same faults are very difficult to reveal and expensive to remove if left uncaught until late in the development of the final product.

The analyzer, which allows liveness and safety properties to be formally proven, is one of the most innovative components of Cabernet. It is being used to study the benefits of formal techniques for the development of real time systems.

The hierarchy manager drives the refinement of specifications guaranteeing the temporal properties already proven to be preserved in different levels of refinement. In this way, the complexity of formally proving temporal properties using the analyzer can be reduced, since only the first specification level has to be analyzed.

The advanced tool generation facilities allow Cabernet basic functionalities to be assembled to

obtain new tools, as powerful as debuggers and tracers for Petri nets. Cabernet is still evolving: in the near future, it will comprise better tools for safety analysis, it will support the design phase and besides it will provide facilities to customize end user interfaces according to the requests of the end users and the needs of the specific
application.

Cabernet differs from traditional case tools for the large set of innovative functionalities provided and the specific target: most existing case tools address a large set of graphical notations, without formal semantics. They usually support only few syntactic checks. Few of them do provide animating facilities. Cabernet focuses on a specific model (Petri nets), formally defined, and provides a large set of advanced functionalities, like flexible executors, analysis tools, hierarchy manager, and tool generation facilities. In the future Cabernet will be able to offer all its functionalities for a large customizable set of notations.

Cabernet differs also from Petri net based tools. Such tools usually deal with pure nets (without data, predicates, actions and time); few of them deal with high level nets (typically Colored nets); none of them deals with time formally and focus on a specific application of nets. Cabernet on the contrary deals with data, predicates, actions and time in an extended way and has a specific goal, namely the support of specification of real time systems.

Background

Menu

ER Nets
Cab nets
Proving temporal properties
Hierarchical decomposition

The Cabernet formal kernel is based on a class of high-level Petri nets called **ER nets**. Petri nets present several advantages:

simple formal definition, preventing ambiguous interpretations of the specifications.

intuitive graphical representation.

operational semantics, supporting execution and simulation.

availability of analysis techniques for proving several interesting properties.

successful experience of use in many different application areas.

In addition, ER nets allow data to be described and provide facilities for representing and analyzing temporal and functional properties.

ER Nets

ER nets and their ability to represent temporal and functional properties are described in [GMMP91] there a sketchy and informal overview of the formalism is given and it is shown how timing aspects can be defined in it. A variation of ER nets called Cab nets is used in the implementation of Cabernet.

In the following sections, analysis techniques available for Cab nets and the facilities available for handling large specifications are explained. It is assumed that the reader is familiar with the general concepts of Petri nets. Otherwise (s)he can refer to [Mur89] for a complete overview.

ER nets are Petri nets where tokens carry information, and transitions are augmented with predicates and actions. Predicates select the tokens that actually enable the transitions, according to the information attached to the tokens. Actions describe how the token values produced by firings depend on the values of the tokens removed.

Time can be represented with ER nets by adding to each token a field **time** representing the time at which the token has been created. The manipulation of **time** fields is determined by predicates and actions associated with the transitions (Transitions).
The value of **time** associated with the tokens produced by a firing represents the **firing time** of the transition.

Actions can represent any relation between the values of the tokens removed by the firings (including the values of field **time**) and the values of the tokens produced by the firings. Suitable axioms constrain variable **time** in order to ensure the representation of an intuitive concept of time, which is non decreasing with respect to sequences of firings.

Depending on the set of axioms, two different semantics can be given: **weak** and **strong** time semantics. The main difference between the two semantics is the interpretation of the possible set of firing times of a transition. i.e., the values of variable **time** that can be associated with the tokens produced by a firing. Weak time semantics interpret such set of values as the set of times at which the event represented by the transition **may** occur, if it ever occurs. Strong time semantics interprets such set of values as the set of values within which the event represented by the transition **must** occur, unless it is disabled by some other firing.

Cab nets

Cabernet uses its own notation that slightly modifies the formal kernel proposed in [GMMP91]. The formal notation of the Cabernet kernel, hereafter referred to as **Cab nets**, is a typed version of ER nets: Each place is associated with a type that represents the type of the tokens that can be produced in the place, predicates and actions refer to tokens as typed parameters. In order to facilitate the construction of an executor, it was decided to use C++ syntax (Syntax of properties) to express types, variables, predicates, and actions.

Cab nets support **mixed time semantics**, i.e., both transitions with weak time semantics (called **weak transitions**) and transitions with strong time semantics (called **strong transitions**) may appear in the same net.
Strong transitions are forced to fire within their firing interval, weak transitions are allowed to fire as well, but not later than the minimum among all the maximum firing times of the enabled strong transitions, this time is the (**deadline**), (Executor).

Proving temporal properties

ER nets, and consequently Cab nets, can be analyzed in several different ways. A first way is provided by testing: the operational semantics of ER nets support a straightforward way of executing a net by playing the token game, and a visualization of the firing occurrences provides a simulation of the system behavior. If time information is attached to transitions, the simulation allows the sequence of temporal events represented by the net to be observed. It is therefore possible to check whether the behaviors specified by the net meet some deadlines that the application is supposed to satisfy (Executor).

Besides execution and animation, Cabernet supports a specific analysis technique that takes into account temporal and/or functional aspects of the ER net. Such technique, described in [GMP94], supports the analysis of bounded invariance and bounded response properties for ER nets. Bounded invariance properties state that certain behaviors do not arise within a given temporal bound. Bounded response properties state that certain behaviors arise within a given temporal bound. Roughly speaking, bounded invariance and bounded response properties differ from the usual safety and liveness properties because they are expressed for a given time interval (Analyzer).

Hierarchical decomposition

Cabernet's hierarchical decomposition mechanisms are based on a rigorous formal theory that guarantees the validity of temporal properties through different levels of abstraction: all the temporal properties proven for specifications at the higher level are valid in the more detailed specifications. In this it is possible to obtain the maximum advantage from the verification mechanisms with the minimum effort. In fact, properties need to be proven only once and at the highest abstraction level, where the amount of details is minimum, and do not need to be proven at all levels. Moreover, since the results of proofs at the highest abstraction level are preserved in the whole hierarchy, the end-user is encouraged to prove properties as early as possible, thus preventing uncaught errors to propagate down in the hierarchy.

The theory underlying the decomposition mechanisms used in Cabernet is described in [FGP93]. The refinement rules applied at each step guarantee the validity of all the temporal properties through the whole refinement sequence.

Validity of temporal properties is guaranteed if the **temporal behaviors** of the nets are in a given relation. A temporal behavior of a net is defined as a sequence of set of firings on the temporal axis: a set of firings is associated with time **t** in a temporal behavior if all the transitions corresponding to the firings in the set fire at time **t** in the same firing sequence. Roughly speaking, temporal behaviors describe how an external observer that can only see the occurrence of events, but not the net itself and its structure,
can see the dynamic evolution of the net.

A refinement satisfies the same set of temporal properties if it does present only temporal behaviors defined in the higher-level net. This definition captures the idea that from the validation point of view we are only interested in sequences of events how they can be experienced by external observes and not in the way the control is implemented (Hierarchy manager).

General

Basic information on how to use Cabernet graphical interface and general topics (storing and retrieving files, printing the nets you work on, tailoring the interface and windows).

Menu

Graphical Interface	Description of its componenets.
File	Saving, restoring an printing
Option	Colors, symbols and general settings.
Window	Operations on the windows.

Graphical interface

The Cabernet graphical interface is composed of four main parts:

Canvas: An area where you can create and modify Cab Nets using the facilities provided by the graphical editor (Editor).

The canvas is bigger than the work area you see. Thus, you are not restricted to use the visible area. To move to the non visible part of the canvas, you must use the **scroll bars**. These are located on the right side of the canvas and at the bottom.
Scroll bars end in buttons with the shape of an arrow. Clicking on this buttons, causes the canvas to ***move*** in the indicated direction.

Scroll bars also have two buttons. Dragging them along the bars causes the same effect as multiple clicks on the arrows in the indicated direction.

Message window: Located at the bottom of the Cabernet window, in this area, messages generated by Cabernet are prompted to the user.

This window has a small button on the upper right corner, when the mouse pointer is on it takes the shape of a cross. Clicking and dragging this button allows you to adjust the portions of the Cabernet window used by the canvas and the message window.

Toolbar: Set of buttons located on the left of the Cabernet window, most of them represent functions of the graphical editor (Editor); except buttons Stop and GO, which are used by the executor (Executor) and the analyzer (Analyzer).

Main menu: Menu bar located on top of the Cabernet window, it displays all options provided by Cabernet.

To select an option from the menu, click on it with the left mouse button. Another way to access the main menu is pressing the Alt key, which activates it, once on the main menu, you can use the cursor arrow keys in the numeric pad to navigate through it. A third option is using the Alt key and the underlined letter of the desired menu item. To leave the main menu, use the ESC key.

File

In Cabernet, nets can be stored in files. Files can be created, stored, retrieved and printed. Options to perform these operations are located in the File menu.

Operations here described can be also applied to time reachability trees (TRT files have **.grp e**xtension while net files use **.net**. Extensions use is explained later in this chapter) (Analyzer).

Menu

New	How to create a new file.
Save and Save as	How to save files and their modifications.
Open	How to load a previously stored file.
Export	How to create a windowsmetafile file
Print	How to print a file.
Printer setup	How to set printer options.

New

To create a new file use the option New from the File menu, files are created into a Windows (Window).

When you ask Cabernet to start a new file, this is created in a new window .

Open

Stored files can be retrieved using the Open option from the File menu. Files are placed into windows (Window).

When you select this option a pop up window appears, in this you select the file to be opened. This window is composed of:

Filter: A text box where you can input the path and a file name pattern which specifies the kind of files to be shown in the file list.

By default the path is set to the directory from which Cabernet was launched and type is set to *.net, meaning all files with extension .net no matter how the first part of their names is composed. The latter is specified by the * character. Instead of it, you can use ? characters, in such case, each ? stands for a single character in the name, e.g. the string ??.net stands for all the names composed of two characters and extension .net.

When the filter is modified, Directories an Files lists are updated as necessary.

Directories: List with the names of the directories in the current path. This list must display at least two entries; **.** meaning ***this*** (current)directory and **..** meaning the ***parent*** directory.

Double clicking on an element of the list causes the selected element to become the current directory.
The same effect is achieved clicking on the Filter button while an element of this list is selected. On directory change, the path in the Filter is modified and the Files list is updated.

Files: List (possibly empty) where the names of the files matching the filter in the current directory are displayed.

Double clicking on a file name or clicking on the OK button while a file name is selected loads this file in the window.

Selection: Text box where the path and file name currently selected, if any, are displayed. You may also input here the name of the file, pressing the RET key or clicking on the OK button causes the file to be loaded.

Cancel: Button located at the right of the window, clicking on it causes the process to be canceled with no changes in the working window.

If you try to load a file which is not a proper Cabernet file, an error message is prompted.

Save and Save as

To store a file you can use two different options from the File menu, Save and Save as.

The Save option, saves the current net in the directory from where it was loaded with its original name. The first time a file is saved, if you use this option, it will be placed in the directory from where Cabernet was launched and its name will be untitl.net, which is the default name for new files.

If you change the Net name in the Modify net pop up (Modify), the name given to the net in that field will be used.

The Save as option allows you to give a name to the file you save. It displays a pop up window with the following objects in it:

Filter: A text box where you can input the path and a file name pattern which specifies the kind of files to be shown in the file list.

By default the path is set to the directory from which Cabernet was launched and type is set to *.net, meaning all files with extension .net no matter how the first part of their names is composed. The latter is specified by the * character. Instead of it, you can use ? characters, in such case, each ? stands for a single character in the name, e.g. the string ??.net stands for all the names composed of two characters and extension .net

When the filter is modified, the Directories an Files lists are updated as necessary.

Directories: List with the names of the directories in the current path. This list must display at least two entries; **.** meaning ***this*** directory and .. meaning the ***parent*** directory.

Double clicking on an element of the list causes the selected element to become the current directory.
The same effect is achieved clicking on the Filter button while an element of this list is selected. On directory change, the path in the Filter is modified and the Files list is updated.

Files: List (possibly empty) where the names of the files matching the filter in the current directory are displayed.

Double clicking on a file name or clicking on the OK button while a file name is selected saves the current window with this name.

Selection: Text box to input the name for the file to be saved. If you want the file to have the extension .net, you must add it to the name.

Cancel: Button located at the right of the window, clicking on it causes the process to be canceled with no changes in the working window or the existing files.

The same result as using this option is achieved by changing the Net name field in the Modify net pop up (Modify).

On file saving, if a file with the same name already exists, it is questioned to user if overwritting.

Warning: To save files where you have used the refinement facilities provided by Cabernet (Hierarchy manager), you must use the option Save from the Hierarchy menu (Navigation tool). Otherwise, Cabernet saves only the current level as a single net losing all the information regarding the hierarchy.

Export

File printing is performed using the Export option from the File menu, the result of the operation is the creation of a new file. This one is a standard Windowsmetafile file which can be imported from different applications, for example, *Winword*.
To give a name to these files, a pop up window similar to the one used in the Save as option is displayed (Save and Save as). The default extension in the filter is .wmf.

Print

The Print command prints the contents of the active edit window.
The Print command is disabled if the active window can not be printed.

See also :
Export

Printer setup

The Printer Setup command displays the Select Printer dialog box where you select which printer you want to use for printing nets.
With the Select Printer dialog box, you can select any of the printers listed.
If you want to set options specific to the printer you chose, choose Set Up.
Some printer drivers have their own help. You will see a Help button if this is so. Choose Help for more information about setting up your printer.

If you need more help or there is no help for the printer driver you are using, read the Configuring a Printer section in the "Control Panel" chapter of the Microsoft Windows User's Guide.

Printer and Port list box

The Printer and Port list box lists the Windows-installed printer drivers.

Setup dialog box

The Setup dialog box contains printer setup options. The options available depend on the capabilities of your printer.

Printer and Port list box

The Printer and Port list box lists the Windows-installed printer drivers.

Press Alt+Down arrow to display the list of installed printer drivers.

Setup dialog box

The Setup dialog box contains printer options that are specific to the printer that you already chose in the Select Printer dialog box.

The Setup button brings up the Window Printer setup dialog box where you change the way your printer is normally configured.

Option

Cabernet's graphical environment can be tailored to fit user's needs or preferences, thus, you can adjust things as display colors, nodes' shapes, and other graphical options, using the Option menu.

Menu

Colors	How to modify display colors.
Symbols	How to assign nodes' shapes.
Reduction factor	How to set objects' size.
Warning level	How to define which messages you want to receive from Cabernet.
Grid On/Off	How to show or hide the grid.
Grid spacing	How to adjust grid points distance
Save option	How to save your preferences.
Load option	How to restore your preferences.

Colors

Cabernet colors, which are specified by number from 0 to 6, are associated to specific objects, they are labeled by the name of such objects. e.g. color 0 (the first one) is associated to the background, color 1(the second one) is associated to the foreground, etc.

When defining objects properties in the editor (Modify) you must refer to this colors by their number. Although the default color for objects in the foreground is 1, using the mentioned option you can assign to them any of the valid colors from 0 to 6.
When you select the Colors option a pop up window appears. This window has the following components:

Scale: Graded from 0 to 15, every value has a color associated, such color can be seen in the color display area.

OK: Button, clicking on it applies the last change and exits the window.

Cancel: Button, clicking on it cancels the last change and exits the window.

Apply: Button, clicking on it causes the last change to be applied to the current work area.

Colors: List of the 7 available color labels. From this list you must select the label to which you want to assign a new color. The selected label appears with a black border.

Symbols

Cabernet uses a graphic formalism with a defined set of symbols, however, these may vary for the same notation in different application fields or due to user preferences. To solve this (possible) problem, Cabernet allows to change the icons representing a certain kind of node.

To manage symbols Cabernet uses two tables:

User's, this consists of 10 symbols enumerated from 0 to 9. The first three are associated to null, transition and place objects respectively. These are labeled with the name of those objects. The rest are labeled with numbers.

Cabernet's, these are the symbols predefined in Cabernet each representedby a number as follows:

0. Vertical empty rectangle
1. Horizontal empty rectangle
2. Empty circle
3. Empty square
4. Horizontal filled rectangle
5. Horizontal line
6. Star
7. Queue
8. Stack
9. Vertical filled rectangle

You can associate to any entry in the user's table a symbol from this table.
By default Null is associated to 0, Transition to 1 and Place to 2.

The procedure to associate to an entry in the user's table a symbol from the Cabernet's table is:

1. Select the option Symbols from the Option menu.

2. A submenu will be displayed, this contains all entries in the user's table. Select the entry you want to modify.

3. A pop up appears, this window contains a scale graded from 0 to 9, each value represents a symbol in Cabernet's table as described before. Select the value of the desired symbol.

When you finish, all objects represented with the modified symbol change their shape to the new one. When modifying objects properties (Modify), you actually refer to entries in the user's table.

Reduction factor

The reduction factor sets the scale used in the display for objects.
Decreasing this value increases the size of the objects. Top and bottom values are 8 and 1 respectively. Trying to go beyond them generates an error message.

Another way to perform this operations is using the Zoom buttons in the toolbar (Zoom).

Warning level

This option sets the kind of messages you want to receive from Cabernet, i.e. you may set this to receive only important messages or to receive every message Cabernet may display.

When you choose this option from the Option menu, a pop up window appears, this contains a scale where you set this parameter. Although the scale is graded from 1 to 10, behavior is given in ranges as follows:

1 to 5 every message is prompted.
6 to 9 only important messages are prompted.
10 no message is prompted.

Grid On/Off

When you select the Grid On/Off option from the Option menu, the state of the grid is switched, i.e. if it is on turns off and viceversa.

The grid is a very useful tool to support the drawing process, where its use is highly recommended. However, when the grid is in use, Cabernet's processes turn slow due to the quantity of resources used by this kind of graphical facilities. It is suggested to turn the grid off once you have finished
drawing.

Grid spacing

A pop up window appears asking for an ALIGN GRID value. This value may be an integer between 32 and 128 which represents the separation between grid points, so giving a value of 32 results in a grid that practically covers the whole canvas and, on the other hand, giving a value of 128 results in a grid of highly separated points. Recommended values are between 32 and 64.

Save option

Using the Save option from the Option menu allows you to save in a file the current settings for colors and reduction factor. When you select this option from the menu, a pop up window appears. In this window you must specify the path and file name for the options. This window is composed as follows:

Filter: A text box where you can input the path and a file name pattern which specifies the kind of files to be shown in the file list.

By default the path is set to the directory from which Cabernet was launched and type is set to *.opt, meaning all files with extension .opt no matter how the first part of their names is composed. The latter is specified by the * character. Instead of it, you can use ? characters, in such case, each ? stands for a single character in the name, e.g. the string ??.opt stands for all the names composed of two characters and extension .opt When the filter is modified, the Directories an Files lists are updated as necessary.

Directories: List with the names of the directories in the current path. This list must display at least two entries; **.** meaning ***this*** directory and **..** meaning the ***parent*** directory.

Double clicking on an element of the list causes the selected element to become the current directory.
The same effect is achieved clicking on the Filter button while an element of this list is selected. On directory change, the path in the Filter is modified and the Files list is updated.

Files: List (possibly empty) where the names of the files matching the filter in the current directory are displayed.

Double clicking on a file name or clicking on the OK button while a file name is selected saves the current settings with this name.

Selection: Text box where you must input the name for the options file. If you want the file to have the extension .opt, you must add it to the file name.

Cancel: Button located at the bottom of the window, clicking on it causes the process to be canceled with no changes current settings or the existing files.

Load option

To restore settings previous saved in an option file (Save option), you must select Load option in the Option menu. When you select this, a pop up window appears where you must specify the option file to be restored. This window is composed as follows:

Filter: A text box where you can input the path and a file name pattern which specifies the kind of files to be shown in the file list.

By default the path is set to the directory from which Cabernet was launched and type is set to *.opt, meaning all files with extension .opt no matter how the first part of their names is composed. The latter is specified by the * character. Instead of it, you can use ? characters, in such case, each ? stands for a single character in the name, e.g. the string ??.opt stands for all the names composed of two characters and extension .opt.

When the filter is modified, the Directories an Files lists are updated as necessary.

Directories: List with the names of the directories in the current path. This list must display at least two entries; **.** meaning ***this*** directory and **..** meaning the ***parent*** directory.

Double clicking on an element of the list causes the selected element to become the current directory.
The same effect is achieved clicking on the Filter button while an element of this list is selected. On directory change, the path in the Filter is modified and the Files list is updated.

Files: List (possibly empty) where the names of the files matching the filter in the current directory are displayed.

Double clicking on a file name or clicking on the OK button while a file name is selected saves the current settings with this name.

Selection: Text box where the file name and path currently selected, if any, are displayed. You may also input here the name of the file, pressing the RET key or clicking on the OK button causes the file to be loaded.

Cancel: Button located at the bottom of the window, clicking on it causes the process to be canceled with no changes to current settings or existing files.

Window

Cabernet provides 8 working areas called windows. Each window is independent from the others.
Each window has its own settings (Option). There is a group of operations you can perform on and between windows.

Menu

Windows and icons layout	
Switching to a window	How to navigate among windows.

Windows and icons layout

Window menu commands allow to put windows and icons on the desktop in order to make displaying easier.
Tile command reduces group open windows dimensions, allowing Program Manager window to hold them.
Cascode command cascodes group windows showing windows titles too.

To put group windows on desktop

Choose Tile or Cascade commands from Window menu.

Use Arrange icons command to put icons into a group window uniformly.

Switching to a window

When you are using more than one window, you will surely need to go from one to another, this operation is performed selecting the desired one from the windows list in the Window menu.

Switching among windows does not affect the contents of them.

Analyzer

The analysis toolset can be used either as a stand alone tool or as part of the Cabernet environment [BFP93]. In both cases you must provide the net to be analyzed and the property to be verified. As result you will obtain an answer on the validity of the property and the time reachability tree (TRT) as output.

The analysis toolset is composed by two major parts: the ***net checker***, which is intended to verify nets to be time correct so they can be analyzed, and the ***TRT builder***, which is intended to perform the construction of the TRT required to analyze the specified properties of the net.

Menu

Check net	Verifying net time correctness.
Time reachability tree	Analyzing net properties.
Time reachability tree Option	Inserting analysis properties

Check net

This option of the Analysis menu, activates the net checker, which verifies if the considered net is analyzable. To perform this operation, it controls that for all transitions:

Time function refers only to places belonging to the preset of the corresponding transition,

Each preset does not comprise two or more places with the same name,

Time functions can be parsed,

Time function refers only to timestamps.

If one of these conditions is not satisfied, the net is not well defined and a warning message is displayed. Otherwise the message Net is analyzable is displayed.

Check net also verifies the absence of infinite long sequences in a finite time. However, it verifies only a sufficient condition since the property is in general undecidable.

The tool checks for the absence of cycles in the net obtained considering only transitions that can fire with an infinitesimal delay. If the tool can prove such condition, a message saying that the net is analyzable is shown.
Otherwise, since it is only a sufficient condition, a message saying that the net ***may not*** be analyzable is displayed. In such case, the analysis can be carried out under user's responsibility.

Time reachability tree

Calls the TRT builder. This builds the TRT and verifies the specified property.
The tree is shown in an available window, where it can be explored (the generated TRT can be stored in a file which can receive the same treatment as any other file). If there is no available window, the tool does not allow you to proceed. When this option is selected, it will first control the same condition controlled by the Check net option (Check net).

To express properties, a predicate language is used. A property can be a symbolic formula, a sequence formula or any composition of them. The syntax to be used in properties definition is provided in a separated section (Syntax of properties).

Menu

Executing	The Cabernet TRT builder.
Analyzing TRT	How to perform TRT analysis in Cabernet.
Syntax of properties	Syntax to be used in properties definition.

See also:
Time reachability tree Option

Executing

Attention: Before proceeding on net analysis, a dialog window, requiring the property to be proved, must be selected (Time reachability tree Option).

Choosing Build Tree menu-item starts the process. While the analysis is carried out, all other option provided by Cabernet are disabled, except for the Stop button.

If the tool cannot solve an expression, e.g. a non linear constraint, it asks the user for the correct result. Analysis can be stopped by clicking on the Stop button. If this happens, analysis will be stopped and the TRT built up to that moment is shown in a window.

On process ending, either by normal procedure finish or by user interruption, tool shows a message where the final (or partial, if stopped) result is displayed.

Warning: If the analyzed net is not time correct, the tool can enter in an infinite loop.

Analyzing TRT

Cabernet shows the TRT in a different window, the number of this window is displayed in the Message window. To see the TRT, switch to its window using menu Window.

The graphical representation of the tree gives a first view on the final result. Each node corresponds to a symbolic state of the TRT. Different symbols are used for the nodes depending on the final status of every state.
This enhances the readability of the final result.

Boxes represent Normal States

Circles represent Final States

Short lines indicates a subgraph which was not built.

Empty rectangles represent deadlock states, in such states there is at least one path that cannot be continued.

Full rectangles represent deadlock and final states.

Long lines indicates overtime states. Such states are generated by a transition that cannot fire within the time limit.

Stars represents unexplored states. Such states have not been explored because the analysis has been stopped, or their exploration was not necessary to reach the final result.

Every node is given a unique name automatically generated by the tool.

Menu

Further examination How to perform a more detailed examination of the TRT.

Further examination

To further examine the final TRT, Cabernet allows to select a node and to see how it is composed. To do this, select (Select) the node to examine and click on the Modify button (Modify).

A pop up window appears, this is composed by five fields:

State Name: Name of the state. This is the only editable field of the mask.

Last Fired Transition: Name of the fired transition that produced this state.

Last Firing Time: Symbolic expression representing the firing time of the last fired transition.

Symbolic Marking: Symbolic time values associated with each token. If a place is marked with two or more tokens, the corresponding symbols are separated by colons.

Constraint: Formula constraining the symbolic values. It defines a relation that must be satisfied by the tokens values in the marked places of that state. Each symbol is implicitly constrained to be non negative.

INSTANTIATE NET: Button, clicking on it generates an instance of the net satisfying the state described. The net is created in an available window, if there is not such a window, a message is displayed. The instantiated net has a name composed as follows **state_of_net_name.graph**, where **state** stands for the state from which the net was instantiated, e.g. **S2** and **net_name** stands for the name of the analyzed net.

This net net can be used as any other Cab Net, you can save it and execute it. To execute it you must first compile it (Making an executable net).

REDUCE CONSTRAINT: Button, using this you can see the constraints applied to only one subset of the variables. Clicking on it displays a pop up window where you must introduce the names of variable you want to analyze.

Names must be separated with ; and no spaces.

Syntax of properties

The syntax to be used on properties definition is:

property
::= **property** '&&' **property** (AND operator)

::= **property** '||' **property** (OR operator)

::= ' *!* ' **property** (NOT operator)

::= '*(* ' **property** '*)* '

::= **sequence**

::= **symbolic**

Menu

Sequence
Symbolic formula

Sequence

A sequence-formula refers to a sequence of events or states, or to any combination:

sequence
::= **sequence** ',' **sequence** (eventually follows)

::= **sequence** '-' **sequence** (immediately follows)

::= **event**

::= **state**

Precedence of operators is, in order: '-' and ',' , and their associativity is from left to right.

Menu

Events and states Definition of this syntactic classes.

Events and states

An event refers to a transition:

event
::= trans-id (trans-id is the name of a transition in the net to be analyzed)

A state represents a marking and hence corresponds to the number of tokens in the marked places. The number of tokens in a place at a given state is indicated with the name of the corresponding place:

state
::= ' @{} **state-def** '@' '

state-def
::= place-id **operator** `constant`
(place-id is the name of a place in the considered net and `constant` is any natural number)

::= **state-def** '&&' **state-def** = and

::= **state-def** ' || ' **state-def** = or

The first rule for ***state-def*** constraints the number of tokens in place ***place-id*** to be lesser or greater than, or equal or different to a constant.

The constraints on the timestamp values can refer to:

operator
::= '<'

::= '>'

::= '<='

::= '>='

::= '=='

::= '!=' (not equal)

All operators have the same precedence and their associativity is from left to right.

Symbolic formula

A symbolic formula establishes a constraint on the firing times of a single transition or on the firing times of a pair of transitions. Constraints can refer to all the firings of a transition (**#**) or to at least one firing (***?***). Thus, firing times of transitions are universally or existentially quantified. Corresponding variables referring to the firing times are directly indicated by the transition names. Formulas must be closed, i.e. all of their variables must be quantified.

symbolic
::= **quantifier** trans-id **operator** **quantifier** **t-time**}

::= **quantifier** trans-id **operator** `constant`

::= '[' **quantifier** trans-id ',' **quantifier** trans-id ']' ':' **formula** ':'

Precedence is: quantifier, operator. Their associativity is from left to right.

The first rule constraints one or all the firing times of a transition to be lesser or greater than, or equal or different to one or all of the firing times of another transition plus a `constant`.

The second rule constraints one or all the firing times of a transition to be lesser or greater than, or equal or different to a `constant`.

The third rule allows the user to write a more flexible constraint in two arguments.

In a formula, the two transitions are identified with the private identifier T#1 and T#2. Other valid identifiers are the symbols of the initial marking.
New symbols may be used to ***connect*** two or more formulas of this type (new symbols are implicitly existentially quantified).

Quantifiers are written as follows:
quantifier
::= '#' (for each)

::= '?' (exists)

The non-terminal **t-time** represents directly the firing times of a transition, or the firing times plus a `constant`:

t-time
::= trans-id

::= trans-id '+' constant

Menu

formula	Definition of the syntactic class.
expr	Definition of the syntactic class.

formula

formula
::= **formula** `&&' **formula** = AND

::= **formula** `||' **formula** = OR

::= `!' **formula** = NOT

::= `(' **formula** `)'

::= **expr operator expr**

Precedence for the operators is: `!',`&&',`||'. Their associativity is from left to right.
operator is defined in:

expr

expr
::= **expr** ' / ' **expr**

::= **expr** ' * ' **expr**

::= **expr** ' - ' **expr**

::= **expr** ' + ' **expr**

::= ' (' **expr** ') '

::= ' min(' **argm** ') '

::= ' max(' **argm** ') '

::= ident

::= `T#1',` T#2'

::= a number

Where ident can be any user defined identifier and ***a number*** can be any natural number.

Precedence of the operators is, as usual: min,max, `*', `/', `+',`-'. Their associativity is from left to right.

Time reachability tree Option

This dialog box allows to define analysis parameters:

1. NOT EXIST: Activating this indicates the analyzer to check for the absence of the specified property. Otherwise, it will check for the presence of a path satisfying it.

2. STOP ON DEADLOCK: Activating this option tells the tool to stop the analysis when a deadlock is found. Otherwise, it will continue on every possible path up to the time limit. Remember that a symbolic state represents several numeric states, this option indicates to stop the analysis whenever one of these numeric states represents a deadlock.

3. TER MODE: Activating this option indicates the tool to perform the analysis considering predicates and actions. Otherwise, the analysis is performed considering only time restrictions.

4. Strategy used: This indicates the tool the search strategy to apply when selecting the next node during TRT analysis. Possible options are: (DEPTH FIRST, BREADTH FIRST and USER CHOICE). Only one can be selected. If USER CHOICE is selected, when two or more states can be explored, the tool displays the TRT segment built up to that moment.
The user can examine each state and select the one to be analyzed.

5. Time limit: Here, the time bound for the analysis must be provided by writing a symbolic expression. This argument is mandatory.

6. Property: The property to be proved (Syntax of properties).
This argument is mandatory.

Clicking on the OK button options are saved.

Customization of facilities

Cabernet provides tools which allow you to customize the environment so it can fit your specific needs on notation and tools.

In these sense, two different sets of tools can be identified, the meta editor which allows notation tailoring, and the tool generator which allows the definition of new tools from existing ones.

At the moment the meta editor is implemented and being tested, but it is not yet distributed in the current version of Cabernet.

Menu

Meta editor	How to adjust the notation to your needs.
Tool generator	How to generate new tools from existing ones.

Meta editor

The meta editor is not distributed in the current version of Cabernet.

The meta editor allows the definition of new notations based on Cabernet's formal kernel. By doing so, the environment can be tailored to allow users who are not familiar with Petri Nets based notations, and specially with Cab Nets, to use the environment.

End users do not need to interact with the meta editor. Definition of new notations can be let to expert users, who will fit the tool to the needs of end users. End users are expected to be experts in the application domain but not in the tool or its default notation. In this way we can see two levels of users, end users and meta editor experts.

Tool generator

The Tool generator is not distributed in the current version of Cabernet.

Cabernet offers a set of tools to meet basic user's requirements. However, quite often users require specific tools which are not implemented. Satisfy all possible needs of every kind of users is almost impossible. Most existing tools provide a fixed set of tools. Cabernet offers the tool generator, which allows to compose tools from those already implemented.

Cabernet offers a set of elementary tools (Tool generator elementary tools) which can be composed through a tool composition language. The composition language is provided by safe untimed cab nets, i.e., Cab nets with no time an with at most one token in each place for each reachable marking. Places are untyped. Tokens in the tool net are Cabernet objects and elementary tools are actions in the transitions. The token in the initial marking of the tool must be named Net. Such token represents the **target** net.

The tool composition language is complemented with a set of predicates (Predicates}), which are placed in the Transition predicate are of the Modify transition pop up (Transitions).

Calls to user defined tools can be done using the Transition subnet field in the Modify transition pop up (Transitions).

To execute a tool, the tool net must be loaded in buffer 1 and the target net must be loaded (Open) into buffer 2. To start execution select option Run tool from the Tool generator menu. In the same menu there is another option Debug tool which performs a step by step execution of the tool, paused executions are resumed using the GO button on the toolbar.
The Stop button aborts tool execution.

To restart a stopped execution or to reexcute the tool, it is necessary to reload the tool net.

Hierarchy Manager

The hierarchical decomposition toolset comprises two basic tools: a refinement tool and a navigation tool. The refinement tool allows the end-user to apply the refinement rules to a High Level Timed Petri Net (HLTPN).
It first checks the selected element for compatibility with the selected rule (e.g. it checks that the transition sequencing rule is applied to atransition). It also checks that the net to which the rule is applied is an Merlin and Farber net (MF net), in fact the rules described in this document and implemented by the refinement toolset can be applied only to MF nets. It then acquires, if needed, the new time functions from the user; it checks for their validity with respect to the time functions associated to the transitions in the original HLTPN; it produces the time functions that can be automatically derived, and, finally, produces the new HLTPN.

Menu

Refinement Tool

Navigation Tool

Refinement Tool

The refinement tool is accessible via menu Refine. The end-user must first select the node (either a transition or a place) to which the refinement rule has to be applied using button select from the buttons on the left hand side, and then one of the refinement rules listed under menu Refine. If the net is not an MF net or the selected rule does not apply to the selected element the refinement is aborted and suitable messages are popped up. Otherwise the system asks the end-user for further details, if needed, through a suitable mask. Items Transition Cycle, User Choice, and Check are not implemented in the current version. Finally the new HLTPN is shown on the canvas window. We report here, for each refinement rule, the input that the end-user must provide, and the controls and computations performed by the tool.

Menu

Refinement Rules

Refinement Rules

In this section, we define some refinement rules that can transform a HLTPN into an implementation that is guaranteed to be correct by construction. An MF net is a HLTPN with strong time semantics, where time-functions associate each enabling tuple with an interval. Such interval can be described as a pair of constant values, called a Static Firing Interval, that represents the set of possible firing times, relative to the enabling time.

The transformation rules we identified are listed below informally. Given a net Ns the application of a rule produces a new net Ni which is an implementation of Ns. All rules have been proven to satisfy the definition of implementation presented in the former section, i.e. a net obtained by applying one of the rules presented below is guaranteed to be a correct implementation of the starting net.

Menu

Transition sequencing rule

This refinement rule replaces a transition t with the sequence of two transitions t1, t2 and a place p. The rule can be applied if there are no transitions conflicting with t (i.e. no transitions sharing some place of their preset with the places in the preset of t).

This rule allows the end-user to select the time function associated with one of the two produced transition. The lowest and the greatest firing time specified by the end-user must be no higher than the lowest and the greatest firing time of the refined transition respectively. If these requirements are verified, the tool derives the time function of the other produced transition. Otherwise, it refuses to apply the rule and does not generate a refined HLTPN.

Place splitting rule

Place p is replaced by two places p1 and p2 with the same preset and postset of p. If place p is initially marked, so will be places p1 and p2. Since no transition is transformed, the event transformation is empty.

No datum is required from the end-user. The tool automatically generates the new time functions associated with the transitions in the postset of the refined places.

Place sequencing rule

This rule establishes that a place p and the transitions in its postset can be replaced by places p1, p2, transition t and a set of transitions t(n+i), each of which corresponds to a transition t(n+i) in p.

The end-user must provide the time function associated with the newly produced transition. The tool checks that the lowest and the greatest firing time of the new transition are not greater than any of the lowest and greatest firing times associated with the transitions in the postset of the refined place respectively. It then generates the new time functions for the transitions in the postset of the refined place.

Transition splitting rule

This rule establishes that a transition t is replaced by transitions t1 and t2 which have the same preset and postset as t and the same static firing time intervals.

No datum is required from the end user. The tool automatically computes the time functions associated with the new transitions.

Firing Time reduction rule

Given a net N the firing time reduction refinement rule consists of substituting a generic transition t of N with a new transition t' having a more restricted static firing interval.

The end-user must provide the new time function. the toolset controls whether it is comprised in the old time function or not. If not, the refinement step is refused as incorrect.

Iteration rule

The iteration refinement rule consists of substituting a transition t with a set of transitions and places that represent the execution of the action modeled by transition t, as an action that may be repeated several times in a cycle. The structure of the resulting net guarantees an upper bound to the number of cycles so that the total execution time is not greater than the execution time of the original transition t.

Such rule has not been implemented yet in the current version.

Navigation Tool

The navigation tool allows the end-user to access the different level of refinement of the specification. The current version supports a total order of refinement levels: only the most refined level can be further refined.

The navigation tool can be invoked through the menu Hierarchy. It allows the visualization of the next or the former level in the hierarchy. Suitable messages are popped up if the end-user attempts to go beyond the first or the last level. Menu Hierarchy also allows a hierarchy to be saved. It differs from the corresponding command available under menu File because it causes the whole hierarchy information to be save and not only the current net.

Executor

Executor tool facilities are provided by Cabernet under the Executor menu. Once you have drawn the net of the system you are modeling, you can execute it using them.

Menu

Making an executable net	What to do prior to net execution.
Execution options	What execution parameters you can set and how to do it.
Net execution	How to execute a net.

Making an executable net

To execute a net, you must first compile it. Compilation is performed through the Compile option from the Executor menu. This option compiles the specifications coded using C++ and generates the executable code of the net.

To compile nets, you must have a C++ compiler installed and properly set in your **PATH** environment variable (Getting started).

Compilation is necessary only to perform execution using the TER execution mode (Execution options).

Net compilation generates the following files:

A file containing the C++ source for the program that evaluates every predicate and executes every action in the net. This file has the same name as the net with the .cpp extension

If it does not exist, a file Make.mak is created. This is the make file used to generate the executable file. You can create your own make file or modify the one produced by Cabernet, in order to satisfy your compilation needs or preferences. For example, adjusting compiler name (as default, this make file invokes the C++ compiler), path and flags or linking your own libraries.

An executable file with the name of the net and the extension .exe. This file is generated using the make file Make.mak.

A file containing error messages generated during compilation. This file has the same name as the net but with the extension .err .

These files are located in the server directory.

Trying to execute a non compiled net produces error message box.
If this happens, compile the net and retry the execution.

Execution options

There are parameters which modify the way an execution is performed. Options to modify these parameters are located in the Executor menu.

1. Run options: This option from the Executor menu causes a pop up window to be displayed. In such window, you can define the following execution parameters.

a. Stop every firing: Activating this option, tells Cabernet to pause before every firing. When execution is paused, a message is displayed in the Message window indicating it. To resume execution click on the GO button (Net execution).

b. Stop every phase: Activating this option, indicates Cabernet to pause on every phase of the execution process. Phases of the execution process are:

Selection of potential enablings, i.e. of all places in the preset of a transition that satisfy the number of tokens required by the transition. This is the same as in common Petri nets.

Selection of functional enabling, i.e. a subset of the previously described composed by those places which satisfy the transition predicate.

Selection of time enabling, i.e. of another subset of the potential enablings composed by those places which satisfy the time constraints of the transition. Functional and time enabling selections are performed in the same phase.

Chosing enabling, in this phase, the selection of the firing enabling is performed. If there are more than one enabling, the selection is performed according to the status of the User choice option explained below.

Firing of the chosen transition.

Stopping every phase allows to look closely to process evolution. To resume execution click on the GO button.

c.TER execution mode: Activating this mode, tells Cabernet to perform predicate and action evaluation during net execution. Otherwise, this are not considered and the net is executed as a simple Petri net playing the **token game**.

If execution is performed as non TER, Time choice and Execution speed options described below are not used, since they rely on time which is not considered.

d. User choice: This option indicates Cabernet to ask for a user selection every time multiple transitions are potentially enabled. Otherwise, the selection is performed randomly.

e. Firing time: This option refers to the way the firing time for a transition will be selected. There are four options, only one of them (the last one selected) remains active. Names reflect their meaning. Random means selection will be performed randomly from the valid time interval. Choosing Lowest or Greatest means that the firing time selected must be the indicated bound of the time interval. User choice tells Cabernet to ask the user for a firing time value on every firing.

2. Run speed: Selecting this option from the Executor menu, brings a pop up window, this window has a scale control graded from 1 to 10, use it to regulate execution Speed factor.

Speed factor 1 means the slowest execution, 10 means the fastest execution. If the current execution speed does not satisfy your needs. Try different values to find the one that better adjusts to your requirements and system characteristics.

Button OK applies the selected speed factor, button Cancel closes the pop up window with no changes.

3. Number of firings: This option displays a submenu with different numbers. Selecting one of them, sets it as the number of firings to be executed. The first option of the Executor menu, Run **n** firings, displays the current value of this parameter as **n**.

Net execution

Once you have compiled the net (Making an executable net) and defined the execution parameters (Execution options) you can proceed to execute it.

To execute the current net, select the Run **n** firings option from the Executor menu, **n** corresponds to the number of firings selected in the Number of firings option (Execution options).

The GO button in the toolbar restarts an execution stopped by Cabernet, e.g. when Stop every firing or Stop every phase options are active.
The Stop button, cancels net execution.

Depending on execution parameters settings, different messages may appear during net execution. The most important are:

Choose a firing time between ... This message is prompted in a pop up window, as answer you must give a value in the specified range. Click
OK to apply the given value. This message is prompted when Firing time choice is set to User choice.

choosing enabling ... This message is displayed in the Message window (General). When there are more than one potentially enabled transition and the User choice option in Run options is active, this message asks for the selection of a transition to be fired.
Transitions are listed in a pop up window. To select a transition from the list click on it with the mouse left button.

click GO to start next execution cycle This is displayed in the Message window on every execution pause. This message is prompted when Stop every firing or Stop every phase options are active.

CURRENT TIME: Message displayed in the canvas upper left corner (General) to show time evolution during net execution.

EXECUTION INTERRUPTED: This message is prompted in the Message window after user interruption of the execution.

Last firing time: Is displayed in the Message window after normal execution end.

number of last cycle executed: Message is prompted at the end of an execution, followed by the number of the last execution cycle.

In addition to these messages, appropriated ones are prompted according to current execution status, including error messages.

Editor

The Cabernet graphical editor allows to edit TER nets. TER nets can be drawn on the canvas window using the Edit menu or the buttons on the left handside. From now on, this set of buttons will be referred as the *Toolbar*.

Menu

Create a place	creating places
Create a transition	creating transitions
Create an arc	creating arcs
Add a token	creating tokens
Select	selecting objects
Move	moving objects
Cut , copy and paste	cut, copy, paste objects
Delete	delete objects
Modify	modify objects
Gridding	using a grid to align objects
Zoom	zooming the view

Create a place

To create a place, you must first activate the Place button in the toolbar, this is the third one in the row, its icon is a circle with an arrow going into it and another going out from it.

If this button is the active one, whenever the mouse pointer enters the canvas, the pointer will take the shape of a pencil, thus meaning it is posible to draw something. In this case, it is possible to draw a place.

When the place has just been created, its properties are set to default values as follows:

Place Name value is composed by the letter P and a consecutive number, names are **P1**, for the first place created, **P2** for the second and so on. The name appears on place's upper left side.

Place Subnet has no value.

Place Size is **100**.

Place Color is set to **1**.

Place shape is set to the default value, initialy this is set to **2**, which corresponds to a circle, but you can change it using the option Symbols from the Option menu.

Type is set to **timed_NKvoid**.

To modify this settings, you must follow the procedure described in section Modify. In that section you will also find information on the meaning of this properties.

Upon its creation, as it must be expected, the place has no tokens associated to it. You can define tokens using the procedure described in section Add a Token .

Create a transition

To create a transition, you must first activate the Transition button in the toolbar, this button has the icon of a rectangle with two arrows, one going into it and another going out from it.

While this button remains active, when the mouse pointer enters the canvas it will take the shape of a pencil, thus meaning it is posible to draw something. In this case, it is possible to draw a transition.

Initially, when the transition has just been created, its properties are set to default values as follows:

Transition Name value is composed by the letter T and a consecutive number, i.e. names are **T1**, for the first place created, **T2** for the second and so on. The name appears on transition's upper left side.
Transition Subnet has no value.
Transition Size is **100** .
Transition Semantics is set to **Strong**.
Transition Color is set to **1**.
Transition shape is set to the default value, initialy this is set to **1**, which corresponds to a horizontal rectangle (not filled). You can change it using the option Symbols from the Option menu. You can create vertical transitons (field Transition Shape set to **0**) if you keep the **SFT** key pressed when you create the transition.
Transition Predicate is set to **TRUE**.
Transition Action has no action defined, this is indicated by a ";".
Static minimum and maximum times are both set to **enab**, being this value the enabling time of the transition.

To modify this settings, you must follow the procedure described in section Modify. This section also offers more information on the meaning of this values.

Create an arc

Before creating an arc, you must activate the arc button in the toolbar, this is the fifth one on the row, its icon is a rounded arrow. If this button is the active one, whenever the mouse pointer enters the canvas the pointer will take the shape of a pencil, thus meaning it is posible to draw something. In this case, it is possible to draw an arc.

To create an arc, you must first select the initial node (place or transition) and the ending node, nodes selection is made by clicking on them. Arcs can only go from transitions to places or from places to transitions, as it should be expected.

Once you have clicked on a node, no arc is drawn until you click on another valid node, e.g. if you click on transition **T1**, then on another transition **T2** and finally on a place **P1**, an arc is drawn from **T1** to **P1**. In this case, clicking on **T2** (which is not a valid node) had no effect.

Following the procedure described above, arcs are created as straight lines. However, you may want to draw curved arcs. To create curved arcs (with max 4 vertices), click on the first node, then click on the canvas in the place you want to position the curve. Then, click in the final node, this will create a curved arc.

When the arc has just been drawn, its properties are set to default values as follows:

Arc Name value is composed by the letter A and a consecutive number, i.e. names are **A1**, for the first place created, **A2** for the second and so on. Arc names are not displayed.
Arc Color is set by default to **1**.

To modify this settings, you must follow the procedure described in section Modify.

Add a token

To add a token, you must first activate the Token button, this one is the sixth on the row, its icon is a circle with a smaller filled circle inside and an arrow going into it.

If this button is the active one, whenever the mouse pointer enters the canvas the pointer will take the shape of filled circle, thus meaning it is posible to draw something. In this case, it is possible to add (draw) a token.

Tokens are created into places. The first time a token is created into a place, the symbol representing tokens appears within that place, this symbol is a small filled circle. After the first one, whenever a token is created, the symbol in the container place remains the same and a number appears above of it, this number is the number of tokens contained in such place.

Upon their creation, tokens have their properties set to default values as follows:

Environment Name, this value is composed by the letters Tk and a consecutive number, i.e. names are **Tk1**, for the first token created, **Tk2** for the second one and so on.
Environment Time is set to **0**.
Symbolic Time is given the value **T0**.
Environment Value is empty.

To modify this values you must follow the procedure described in section Modify.

Select

To select an object the Select button must be activated. This is the one on the top of the toolbar. Its icon is an arrow. Selected objects are highlighted. Once you have selected the object, you can perform the following operations on it: Modify, Copy, Cut, Paste or just Move it .

To select a single object in the canvas, you can do one of the following:

Once activated the Select button, position the mouse pointer on the object you want to select and click with the mouse left button.
To select an object without activating the Select button, click on it with the mouse right button while the **Ctrl** key is pressed.

To select groups of objects, you have to follow one of the procedures described below:

1. While the Select button is active, position the mouse pointer near (not on) the first object you want to select. Then, while you keep the left mouse button pressed, move the mouse, at this moment a square appears. Drag over the desired objects, so they all are surrounded by the square. Release the mouse button, now all these objects are selected.

2. If the Select button is active, click the left mouse button while the mouse pointer is on the first object you want to select, then while you keep the **SFT** key pressed, click on every object you want to select.

3. As noticed, the previous methods require the Select button to be active. To select multiple objects without activating the Select button, follow the same procedures using the right mouse button and **Ctrl** key instead of the left one. This is some sort of *shortcut*.

When there is a group of selected objects; Clicking on one of them with the left mouse button if the Select button is active, or with the right one and **Ctrl** key without activating the Select button, while you keep the **SFT** key pressed, causes the object to be de-selected.

Move

There are various ways to move an object in the editing area. These are described below:

To use the Move button you must first acitivate it, this is the one located on the upper side of the toolbar. Its icon is a hand. When this button is active and the mouse pointer enters the editing area, the pointer takes the shape of a hand.

1. With the Move button active. Position the mouse pointer on the object you want to move then press the left mouse button and drag it to the new position, while you keep the mouse button pressed. Release the mouse button and the object appears on the indicated position. These method is applicable only to move single objects in the editing area.

2. Another way to move a single object is the following: in this case it does not matter which button is active in the toolbar. Position the mouse pointer on the object you want to move, then press the right mouse button and, while keeping this button pressed, drag to the new position of the object. Release the mouse button and the object will appear on the indicated position.
   When multiple objects are selected, any of the two methods mentioned above affects only the object on which they are applied directly. The rest of the objects remain as they were before the operation.

3. To move sets of objects, first select them and then apply any of the two methods mentioned above while you keep the **SFT** key pressed.

Cut copy and paste

When an object (or a set of them) is Cutted, they are placed in memory so if you wish to reinsert (paste) it (them) in the canvas you can do it. If you want to delete an object permanently from your current work, select it and click on the Delete button, this button is the one with the scissors icon. Before doing so, be sure you will not want to recover the deleted object(s), because you will not be able to do so.

To cut an object or a set of objects in the editing area, first select the object(s) to be cutted, and then do one of the following: Select the option Cut from the Edit menu .

To copy an object or a set of objects in the editing area, first select the object(s) to be copied, and then do one of the following: Select the option Copy from the Edit menu. The object(s) will be copied in the Windows Clipboard too.

To paste an object or set of objects that has been previously cutted, select the Paste option from the Edit menu.

Delete

To delete an object or a set of objects in the editing area, first select the object(s) to be deleted, and then select the option Delete from the Edit menu .

See also:

Cut, copy and paste

Modify

There are different ways to modify an object. They are described in the following paragraphs. When you want to modify an object, you will follow one of the procedures described below.

1. To modify an object in the editing area, first select it. Then, click on the Modify button, this is the tenth in the toolbar.

2. Another option is to select the object and then choose the option Modify from the menu Edit.

3. If the Select button is active, double click on the object with the left mouse button.

Each kind of object has a different set of properties and a different dialog box to modify them. These are described in the following subsections:

Menu

Places	modifying places
Transitions	modifying transitions
Arc	modifying arcs
Tokens	modifying tokens

Places

When you ask to modify a place, the following data appears in a pop up window where it can be edited, following its name we give a brief description of what each place property means.

Place Name: This is the name given to the current place. Place names syntax is the same as for C++ identifiers.
Place Size: This value specifies the size of the icon that represents the current place.
Place Color: This value specifies the color asigned to the place. Using colors makes possible to remark an important place or group of them.
Place Shape: This value defines the shape of the icon representing the current place. Allowed values are from 0 to 9 each representing a different shape. Default values correspond to standard shapes, i.e. circles for places, rectangles for transitions and arrows for arcs.
Place Type: In this field the type of the objects (tokens) the place can contain is specified. Each type must be defined as a C++ class. New types can be defined using the option Modify net from the edit menu.
Tokens (button): This button allows to modify tokens contained in the current place .

Transitions

When you ask to modify a transition, its properties and their values aredisplayed in a pop up window where they can be edited. This properties and their meaning are described below.

Transition name: This is the name given to the current transition. Transition names syntax is the same as for C++ identifiers.
Semantics: This can take only one of two values, **Weak** and **Strong**. Which refer to the time semantics of the transition.
Transition Size: This value specifies the size of the icon that represents the current transition.
Transition Color: This value specifies the color asigned to the transition, using colors makes possible to remark an important transition or group of them.
Trans. Shape: This value defines the shape of the icon representing the current transition. Allowed values are from 0 to 9, each one representing a different shape.
Transition Predicate: In this place is where the predicate which must be satisfied to enable the transition is written.
Transition Action: In this place, the actions to be taken upon transition's firing must be specified.
Static min. time: Is the lower bound for the time interval within which the transition must (or may, depending on its time semantics) fire.
Static max. time: Is the upper bound for the time interval within which the transition must (or may, depending on its time semantics) fire.

Arcs

When you ask to modify an arc, its properties and their values are displayed in a pop up window where they can be edited. Arcs have only two editable properties:

Arc Name: This is the name given to the current arc, its syntax is the same as for C++ identifiers.
Arc Color: This value specifies the color asigned to the current arc, using colors makes possible to remark an important arc or group of them.

Tokens

In the window where Places properties can be edited, you will find a button named **tokens**, clicking on it, displays the window where you can modify the properties of the tokens contained in such place. The list of these tokens is displayed at the left side of the window. Clicking on the name of one of them shows its current values and allows you to modify them.

A brief description of these properties is given below.

Environment Name: This value is the name given to the token.
Environment Time: This represents the current environment time for the selected token. This time is used by the Executor.
Symbolic Time: Describes the symbolic time assigned at the current moment to the token. This name is used by the Analyzer.
Environment Value: Displays the current value of the token within the environment.

Gridding

The grid is a tool which serves as a guide that helps you to align and order objects in the canvas.

The grid can be activated and deactivated using the option Grid On/Off from the Option menu. Upon grid activation menu-item, there is another menu-item Grid spacing to estabilish the distance between two grid point.

Zoom

Through the zoom options, you can magnify or reduce the net currently displayed in the canvas in order to see the whole net or to look at the details of a part of it.

The zoom buttons are the ones located at the bottom of the toolbar. The last one on the row reduces (zooms out) the contents of the canvas, the other one magnifies it (zooms in).