Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich
Digital Systems and Design Automation Group

HS 2022

Prof. L. Josipović / J. Xu
based on Prof. L. Thiele's material

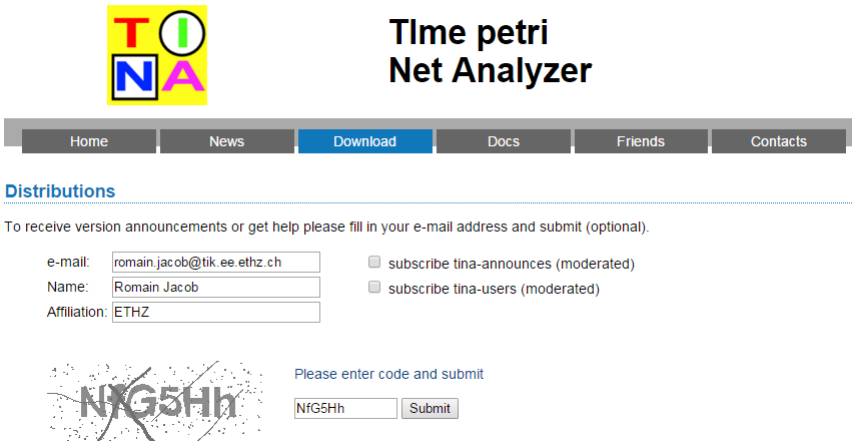
Discrete Event Systems

Short TINA Tutorial v1.1 for Exercise Sheet 13

This document is a short tutorial on the use of the modeling tool TINA (**TI**me petri **N**et **A**nalyzer). TINA is an academic tool which has been developed in the OLC, then VerTICS, research groups of LAAS/CNRS in France.

1 Basic modeling and token game

a) Start by downloading the last version of the tool from: projects.laas.fr/tina/
It is available for most common operating systems.

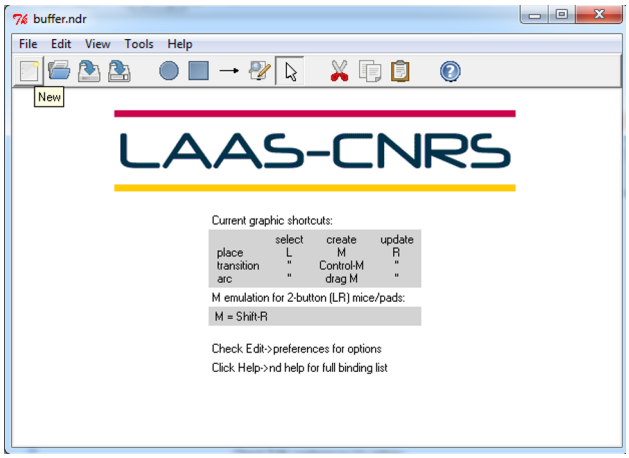


b) Unzip the folder `tina-x.y.z` in your preferred repository.

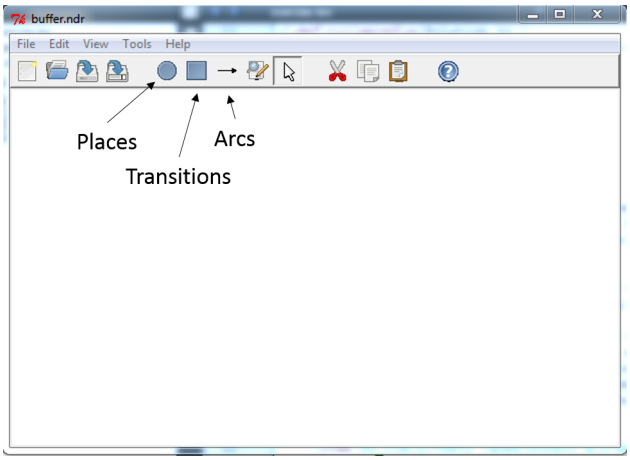
c) In `bin`, open the `nd` application. This is the GUI support for the TINA tool.

d) You should get the following window. Start a new net.

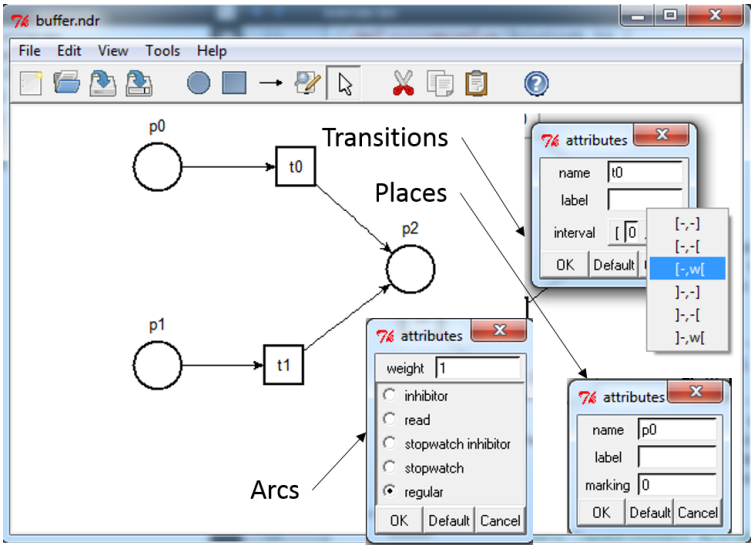
Beware! When saving your nets, you need to manually write the file extension `.ndr` after your file name. Otherwise you won't be able to open the file from `nd` afterwards.



e) There are three main tools to build up you net: places, transitions and arcs. On click, you can position places and transitions. For arcs, click on the starting place or transition and drag on to the destination transition or place.



f) By doing so, you can easily construct a small net. Attributes for all elements (e.g., marking of places, weight of arcs, etc) can be accessed with a right-click on the element.



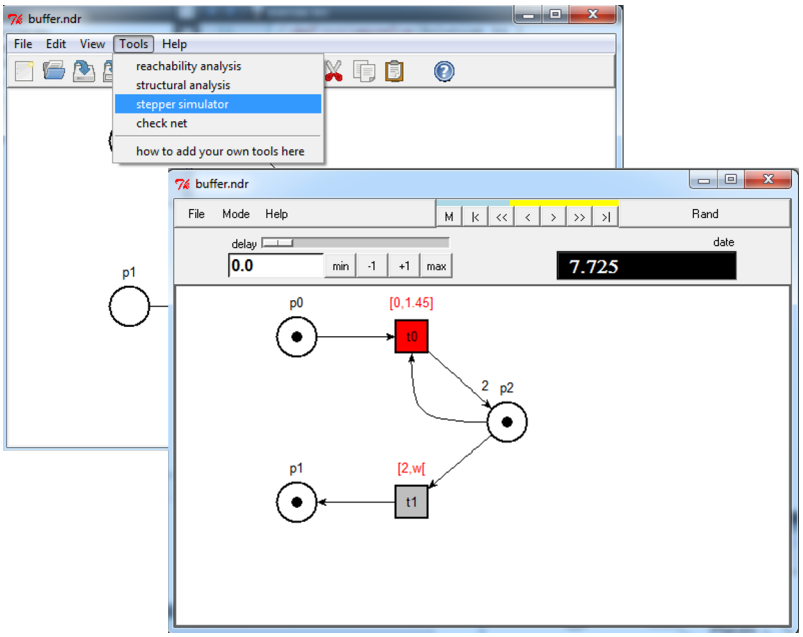
- To create inhibitor arcs, create an arc from a place to a transition, then right-click that arc to open its attributes, and choose "inhibitor". The weight of an arc (i.e., how many tokens are removed ore added by the transition) can also be set in that attribute window.
- For transitions, there is an attribute called `interval`. This is because TINA models time Time Petri net, hence we need to describe the delay associated to transitions. You can associate any timed behavior to the transition (as long as the lower bound is not larger than the upper-bound...), depending on the type of interval you choose.

  Note. w means $+\infty$.

  The interval $[0, w[$ represent the "non-timed behavior" of Petri nets, i.e., a transition can fire as soon as it is enabled, but it can also fire at any further point in time. In other words, nothing is forcing the transition to fire.

  If you want your net to follow the "earliest firing rule" (i.e., transitions fire as soon as they are enabled), you just change transitions interval to $[0, 0]$. If you want a time transition with exact delay e.g. 5, set the interval to $[5, 5]$. This yields the transition to fire exactly 5 time units after it has been enabled (assuming the transition is still enabled by the marking at that point in time).

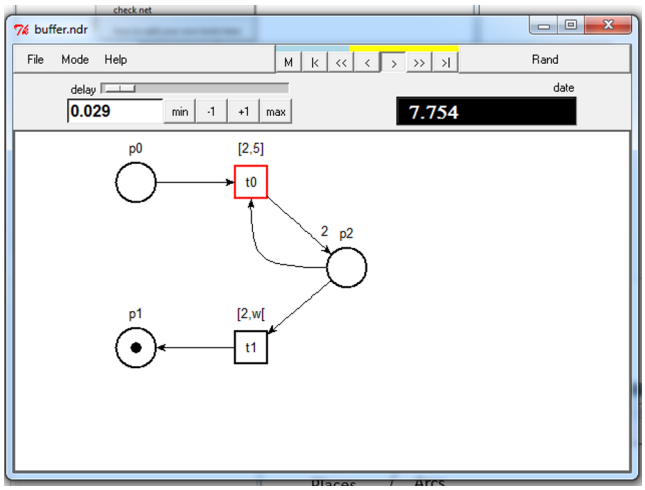
g) From the GUI, you can access directly a random simulator, which is very useful to test the behavior of your net. You can access it via `Tools/stepper simulator`.



By clicking `Rand`, you trigger a random evolution of the net. Once stopped (or when you reach a deadlock), you can go back and forth in the trace using the lecture bar:

A transition in red is an enabled transition that can fire (i.e., the lower-bound of the remaining delay has reached zero). A transition in grey is enabled by the current marking but cannot be fired yet (due to the time delays).

On clicking the next or previous button < >, the fired transition is shown red-boxed.

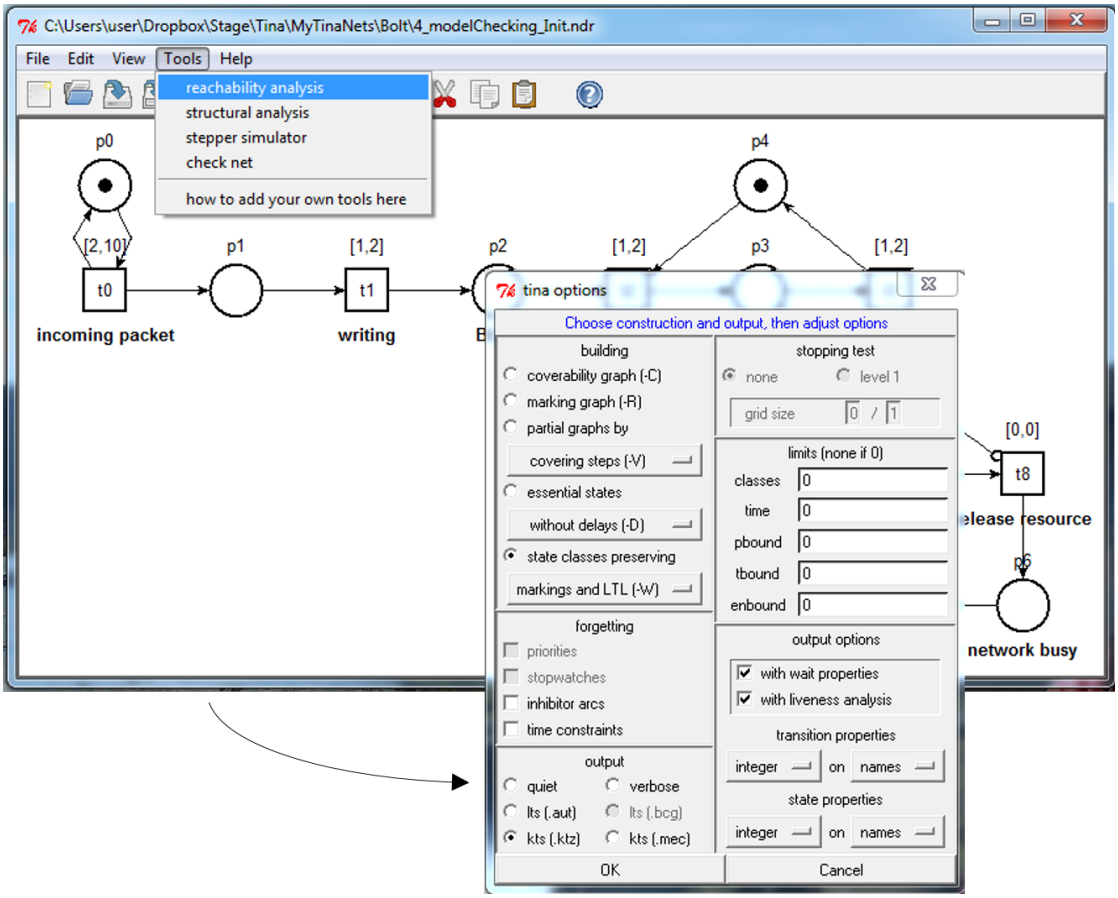


You can go back to the editor via `File/return to editor`, or using `Ctrl-q`.

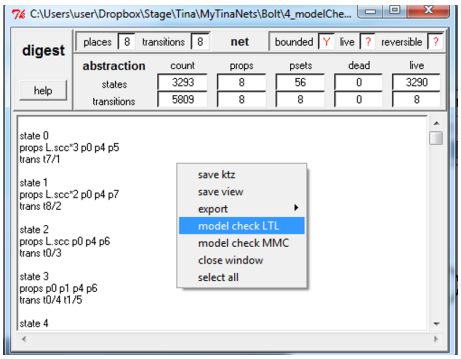
2 LTL model-checking with TINA

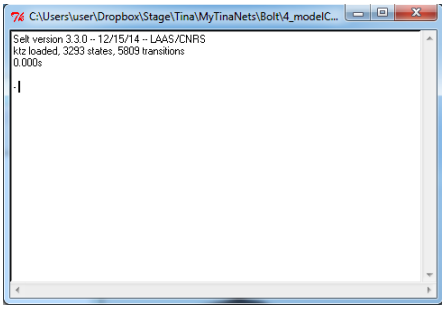
In order to use the model-checker in TINA, proceed in the following steps:

a) First you need to compute the reachable sets of states. To do that, open the module `Tools/reachability analysis`. You should not have to change any parameters, just click OK.



b) A new window open with some information about the system you don't really care about. Just click anywhere within the window and select `model check LTL`. This opens the interface of the model-checker, `selt`, in a third window.





c) LTL is a different logic than CTL. For our simple purposes here, we can say the difference is that the E quantifier doesn't exist, the A quantifier is always implied. Furthermore, G writes $\Box$, F writes $\Diamond$, and X writes $\circ$. This is summarized in the table below:

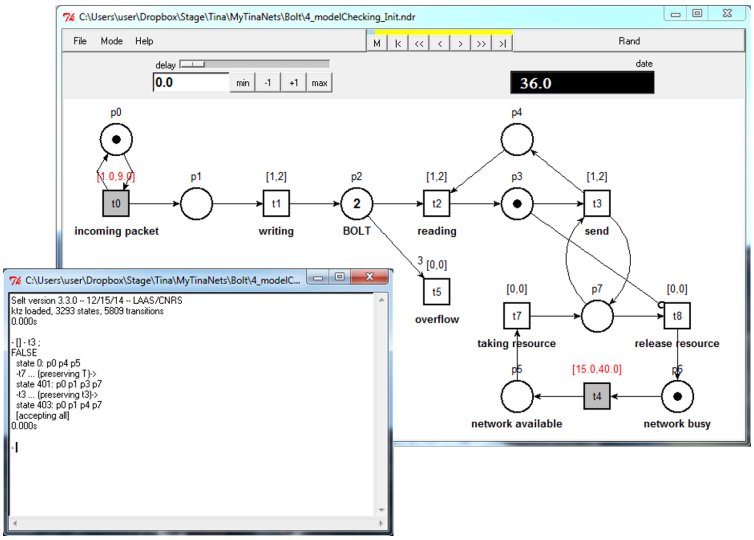
CTL	AF a	AG a	EF a	AX a	...
LTL	$\Diamond$ a	$\Box$ a	$\neg$ AG $\neg$ a	$\circ$ a	...
in TINA	`<> a ;`	`[] a ;`	`- [] - a ;`	`() a ;`	...

In the `selt` window, you can type in LTL queries using elements name from your Petri net. For instance, you can ask if there is always a token in place p0 with the query "`[] p0 ;`". There are a few important things to note:

- Queries are terminated by `;`
- In order to avoid parsing problems, separate each element with a space.
- Every time you modify your net, you need to recompute the set of reachable states (procedure from **a)**), before checking a new query.

The complete semantics of `selt` queries are available here:
projects.laas.fr/tina/manuals/selt.html

d) Finally, once you have `selt` open, you can go back to `nd` and launch the `stepper simulator`. Then, if you enter a new query into `selt`, a verification trace will be automatically loaded into the stepper history, when applicable (some queries do not generate counter-examples).



Note that this is nice from a design perspective but it can be a big workload for your computer if the trace is really long. Avoid using the stepper if your computer has a hard time...