Integrated Net Analyser INA V2.1 Release Note

In the sequel we list the differences between the versions 1.7 and 2.1 of the Integrated Net Analyser.

1. Facts

Facts are transition nodes that are supposed never to fire (dead transitions). Facts have been used in INA to restrict the computation of the reachability graph: a state in which a fact was enabled was not developed further (and called a 'bad' state).

Facts are no longer supported in INA.

But the possibility to restrict the computation of the reachability graph is kept by using a 'bad' predicate: If the predicate is satisfied by a state, this state will not be developed further.

2. Self-modifying nets

In a self-modifying net an arc is inscribed by a formal sum of places. The weight of the arc then is the corresponding sum of the current marking. In INA until version 1.7 we had a rudimentary implementation of self-modification in which the number inscribed to an arc was interpreted as the number of a place, and the weight of the arc then was the marking of that place. Since this implementation does not meet the users' demands we omit it.

Self-modifying nets (VALKs rule) are no longer supported.

3. Place-timed nets

In a place-timed net, for any place p there is assigned a duration d and any token born at place p then has to wait there for d units of time until it may be consumed. Hence, the state of place-timed net contains for any place p and any time $t \leq d$ the number of tokens which have to wait for t units of time at place p, i.e. the states are matrix-valued. To handle and to store matrices costs a lot of space and time; on the other hand, our users prefer to use time restrictions on transitions or arcs rather than places:

Place-timed nets are no longer supported.

Safe place-timed nets can be easily simulated by arc-timed nets. If 'the' token at place p has to wait there for d units of time, associate the interval $[d, oo]$ with every post-arc of p. This has the same effect as long as there is no second token that sets back the clock at place p.

4. Editing

In the editor we have the new function *insert a run place*. This function is of practical interest e.g. for Timed nets, if you want only singleton steps to be fired. The run place is marked with one token and loops around any transition (colour), therefore it kills the net under the safe transition rule.

You also have the possibility to *add a loop* to the net. The loop consists of a single place marked with one token having arcs to and from a single transition. If you associate a time constraint to this loop you can get (maybe only some of) the transient states. A state is transient, if no transition is enabled but by the flow of time the state changes to a state where some transition is enabled. Transient states are not included in the reachability graph (to keep it small). Note, that the initial state may be transient.

5. Firing
Using by-hand simulation of a net with time constraints not only the actual state and the enabled transition resp. steps are displayed but also the system time for the actual state.

6. Analysis
Analysis has been reorganized to make it more effective. This concerns mainly the reachability analysis.

The file STATES.GRA *is no longer supported.*

Until INA 1.7 during the generation of a reachability or coverability graph information on the nodes of the graph except markings and clock positions had been written to that file and analysis was split into two parts: First one had to decide all questions on reachability, then, after reading the file, the questions concerning liveness, paths and so on. The advantage was that after the first part one could switch off the computer and do the second part the next day, the disadvantage the time needed to write and read the file. We decided in favor of the time.

7. Structural Boundedness
In the analysis menu you can now choose the function *decide structural boundedness.* The algorithm then decides whether the (underlying) net is covered by semi-positive place-sub-invariants without computing them. A net is covered iff it is bounded under any initial marking.

8. Non-reachability Test
The analysis menu now offers the function *non-reachability test of a partial marking using the state equation.* The algorithm decides whether the given partial marking is not reachable without computing a part of the state space using the state equation. A partial marking is not reachable if the state equation has no solution.

9. Predicates
In reachability analysis and model checking we work with (state) predicates. A predicate is a disjunction of conjunctions of negated or non-negated atoms (disjunctive normal form). An atom is an assertion of the form

$low \leq value \leq upp$ (with $0 \leq low \leq upp \leq oo$)

For marking atoms, *value* is the token load of a certain place; in the case of a time atom, *value* is interpreted as the clock position of a certain transition or (in arc-timed nets) of a certain place. So the atom

m_1: *low* =1, *upp*=2;

is satisfied by all states where the place p_1 holds one or two tokens. Predicates are used as 'bad' predicates during the generation of the reachability graph, for the computation of minimal paths and in the CTL-model checker.

10. Minimal Paths
Computation of minimal paths has been extended. In the analysis menu you can now choose the new function *compute a minimal path from the initial state to satisfy a predicate.*

Until INA 1.7 you only can compute minimal paths to partial markings. Given a predicate INA now computes a minimal path to a state, that satisfies this predicate. You can choose minimal length, minimal time and minimal value.

11. Reachability analysis

Using the function *compute a reachability graph* you have to set only a few options concerning the type of reduction to be applied. Then the graph is computed faster than by INA 1.7. Not only the number of states generated but also the number of arcs is written to the session report. After that you will reach the graph analysis menu where the following new functions are offered:

convert a set of states to a predicate

The predicate defined will be satisfied by exactly the given states.

define an enabledness predicate

The predicate defined will be satisfied by exactly those states which enable at least one of the given transitions.

check a CTL-formula

You have to specify a formula of the Computation Tree Logic (using the temporal quantifiers $EX, AX, EF, AF, EG, AG, EU, AU, EB, AB$ and predicates defined before). Then it is checked whether the formula holds at the initial state.

compute strongly connected components

The strongly connected components (not only the terminal ones) and their reachability relation are computed.

inspect a result file

If you need you can display the session report or any other file.

The other analysis functions known from INA 1.7 are maintained with one exception: to *draw the graph as a tree* is no longer supported.

Remark: Please, be sure that none of the files OPTIONS.INA or COMMAND.INA created by an earlier version of INA is present in the directory that you start INA 2.1 from!