

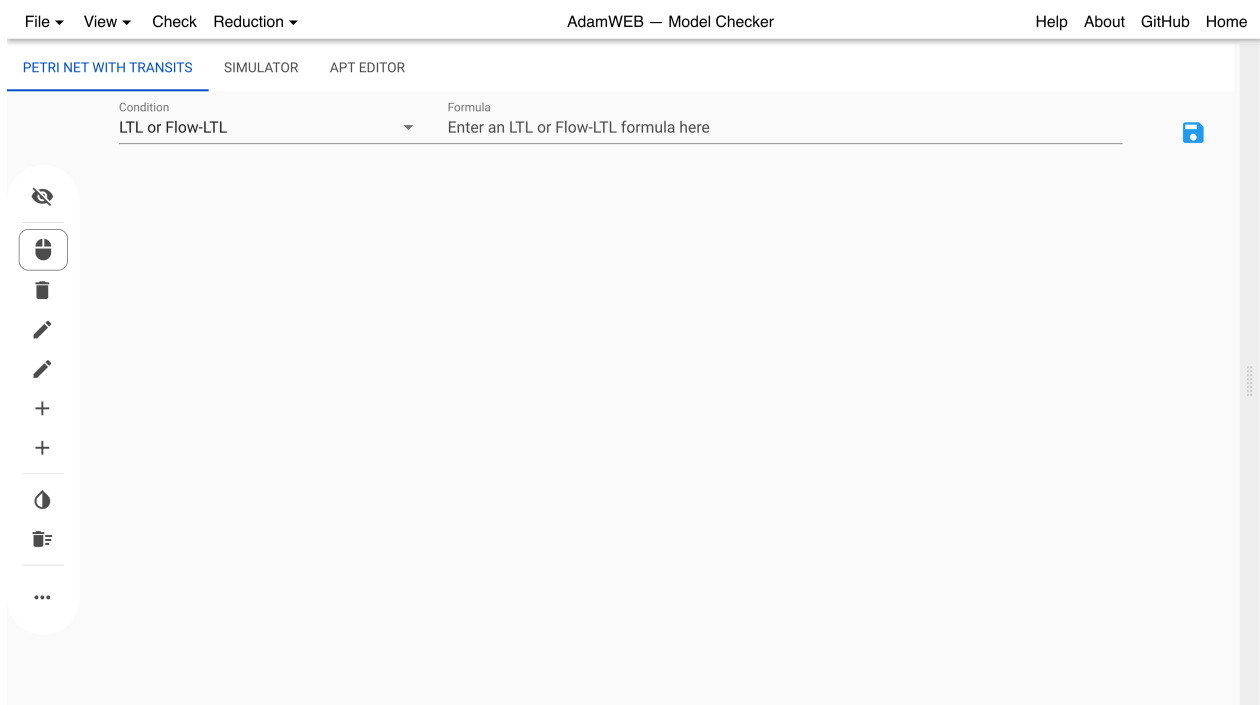
User Guide to the Web Interface (Model Checking)

With this user guide, we give an overview of some common workflows of the web interface for AdamMC.

- [General Items](#)
- [Create a Petri net with transits](#)
- [Simulate a Petri net with transits](#)
- [Syntax of Flow-LTL](#)
- [Model checking Petri nets with transits](#)
- [Reduction from Petri nets with transits and Flow-LTL to Petri nets and LTL](#)
- [Text Editor](#)

General Items:

When opening the web interface for the model checking approach in your browser, you get the following picture:



At the top of the screen, there is a menu bar with the following items:

- **File**
 - **New Petri net with transits** - clears the old Petri net with transits and creates a new one.
 - **Load APT from file** - loads a file from your disk in the APT format (see [here](#) for a format description (Section 4.1)).
 - **Save APT to file** - saves the current Petri net with transits to your disk in the APT format (see [here](#) for a format description (Section 4.1)).
 - **Load example** - loads one of the provided example Petri nets with transits.
- **View**
 - **Log Window** - shows a logging window with debugging information for advanced users.
 - **Job Queue** - shows a panel with the recent jobs and results of the user and the possibility to load the results back into the interface, or to delete or cancel them. Note that only the text and not the colored layer is clickable. You can

exchange your unique identifier of the browser to show others your job list, results, and problems.

Your jobs run/running/queued on the server

APT	Job type	Options	Time started	Time finished	Status	Action	Queue position	Delete
.name "SDN-Heanet"	Model Checking Result	{formula:"A F pOut"}	14:05:59	-	RUNNING	Cancel	In progress	Delete
.name "SDN-Arpanet196912"	Model Checking Result	{formula:"A F sw000"}	14:05:45	14:05:47	FALSE	Load	-	Delete
.name "SDN-Arpanet196912"	Model Checking Result	{formula:"A F pOut"}	14:05:24	14:05:26	TRUE	Load	-	Delete

More info

The jobs listed here are stored in-memory on the server and will disappear if the server is restarted.
You will also lose access to them if you clear the "local storage" of your browser. That's because you can only see jobs that correspond to a randomly generated unique ID that is stored in your local storage.
Your unique ID is 249911eb-41ca-4972-a5ec-b4ef79c4021f.
If you use multiple browsers, you can share one unique ID between them in order to have the same list of jobs appear in all of your browsers.

Other Browser UUID

USE OTHER UUID

- **Show right panel** - expands or collapses the right panel which is shown after a result is obtained. This can also be done with the slider, which can be used to customize the sizes of the panel.
- **Show physics controls** - adds the following slider to the bottom of the screen.



Here, the behavior of the physics control for the nodes of the visualized objects, i.e., the Petri net with transits or the constructed Petri net, can be customized. When the nodes are unfreezed (see [here](#)), the nodes can freely move in the panel. To minimize overlapping, the *Repulsion Strength*, the *Link strength*, and the *Gravity strength* can be modified.

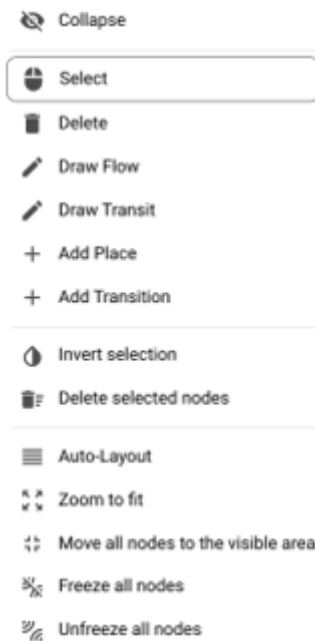
- **Show node labels instead of IDs** - Most of the nodes of the constructed Petri net for the reduction methods for checking Petri nets with transits against Flow-LTL have a correspondence to the input Petri net with transits. With this button, you can toggle between showing the names of corresponding nodes or the original names as labels. See [here](#).
- **Check** - starts the model checking procedure and afterward opens a tab on the right showing the answer if the input Petri net with transit satisfies the Flow-LTL formula formula. If it is not satisfied a counterexample is given. See [here](#).
- **Reduction**
 - **Petri net** - creates the constructed *Petri net* for the reduction method from the given input Petri net with transits and Flow-LTL formula and shows it on the right. See [here](#).
 - **LTL formula** - creates the constructed *LTL formula* for the reduction method from the given input Petri net with transits and Flow-LTL formula and shows it on the right. See [here](#).

The items to the right give you the following features:

- **Help** - opens a help dialog with some shortcuts and the syntax of Flow-LTL formulas.
- **About** - opens a dialog with some information about the web interface.
- **GitHub** - opens the source code for the web interface on GitHub.
- **Home** - leads you back to the index page to choose between the model checking and the distributed synthesis approach.

Create a Petri Net with Transits:

To model a Petri net with transits, the menu bar on the left is used:



Here, the following features are available:

- **Collapse** - collapses the menu bar to make more space for the actual drawing panel, or expands it again.
- **Select** - changes to the mode where a single node can be selected by clicking the node or several nodes can be selected by holding the ctrl-key while clicking the next node. Clicking and holding the left mouse button in a free area creates a rectangle which selects all nodes in the rectangle.
- **Delete** - changes to the mode where clicking a node or an arc deletes the node or the arc.
- **Draw Flow** - changes to the mode to draw the control flows (the black arcs) between the nodes. Clicking and holding the left mouse button on a node allows you to move the visualized arc to the desired successor node. Note that only arcs between places and transitions are allowed.
- **Draw Transit** - changes to the mode to draw transits (the colored and/or labeled arcs) between the nodes. First, click the place where the transit should start, second the transition used for the transit, and third as many successor places as you want to connect with the data flow. Hitting *enter* ends the selection. For creating initial transits, the first node can already be the transition. Note that a control flow is added between the nodes when none is existing.
- **Add Place** - changes to the mode where each click creates a new place at this position.
- **Add Transition** - changes to the mode where each click creates a new transition at this position.
- **Invert selection** - inverts the current selection.
- **Delete selected nodes** - deselects all currently selected nodes.
- **Auto-Layout** - creates a new random automatic layout of the nodes.
- **Zoom to fit** - zooms into or out of the panel such that all nodes are visible.
- **Move all nodes to the visible area** - changes the position of the invisible nodes of the current cutout of the panel such that they fit into the current cutout.
- **Freeze all nodes** - deactivates the physics control and the movability of all nodes.
- **Unfreeze all nodes** - reactivates the physics control, deletes the current coordinates of all nodes, and reactivates the movability of all nodes.

Drag and drop can be used to move the **nodes** (when clicking the nodes) and the **panel** when clicking into the free space and holding the *shift*-key. Dragging a node with a pressed *ctrl*-key **snaps** the node **to a grid**. **Zoom in and out** can be done with the mouse wheel. **Enabled transitions** are visualized with an asterisk *.

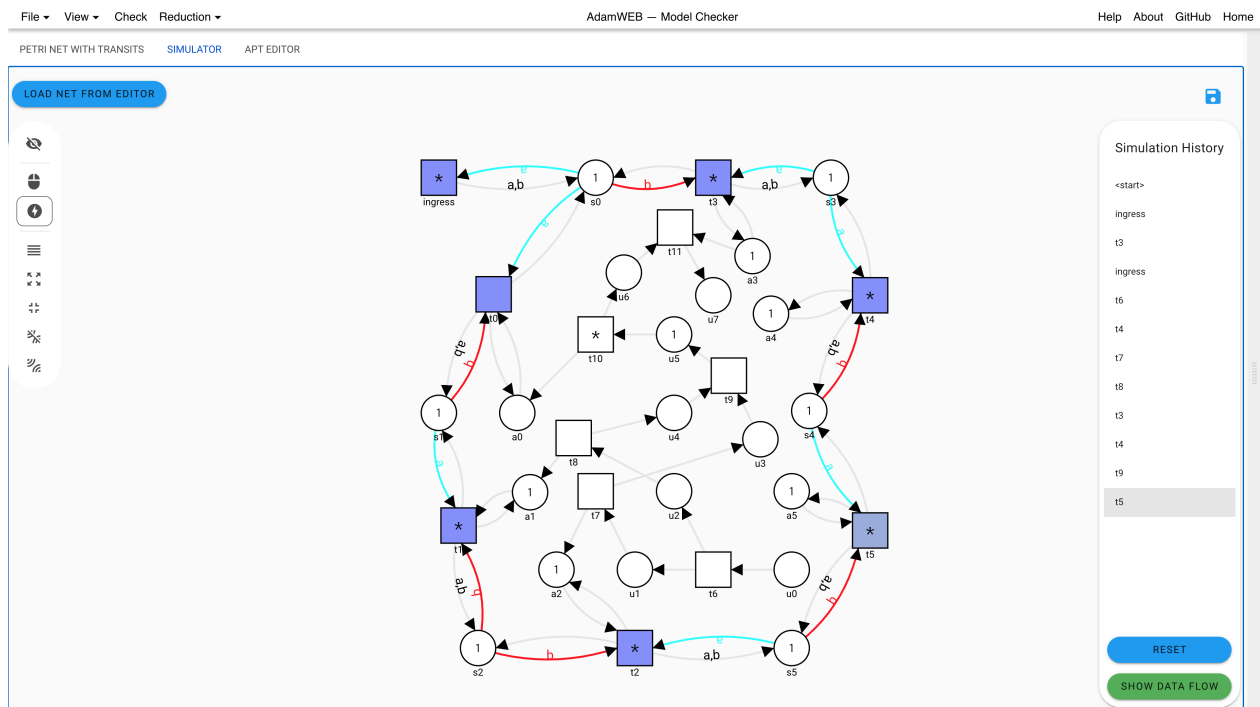
Clicking a node with the **right mouse button** opens a context menu for the node which shows the name in the first line and the label in square brackets in the second. The options for the nodes are:

- Place
 - **Delete** - deletes the node.

- **Rename** - allows to enter a new name.
- **Set initial token** - allows to enter a number of initial tokens. Note that the current approach only allows for model checking safe, i.e., 1-bounded Petri nets with transits.
- **Toggle is special** - allows to mark this place as special. This is used to automatically generate the Flow-LTL formulas for a Büchi, reachability, and safety condition (only visible if the Büchi, Reachability, or Safety condition is selected in the drop down menu left to the formula field).
- Transition
 - **Delete** - deletes the node.
 - **Rename** - allows to enter a new name.
 - **Set weak fair** - marks this transition as *weak* fair (only visible if not already weak fair).
 - **Set strong fair** - marks this transition as *strong* fair (only visible if not already strong fair).
 - **Remove fairness** - removes the fairness constraint from this transition (only visible if the transition is marked as weak or strong fair).
- Flow
 - **Delete Flow** - deletes the flow.
 - **Set inhibitor arc** - marks the flow as an inhibitor arc (only visible for ingoing arcs from transitions, which are not already inhibitor arcs). Inhibitor arcs are visualized by a circle at the end of the arc instead of the arrow head.
 - **Set not inhibitor arc** - removes the inhibitor marker of a flow (only visible for inhibitor arcs).

Simulate a Petri Net with Transits:

Clicking on the **SIMULATOR** tab allows to fire enabled transitions (indicated by the asterisk *) in the Petri net with transits:



When the **thunderbolt** item of the left menu bar is chosen, transitions are clickable and a visual feedback (flashing green or red) is provided whether the transition has fired or not. On the right, the list of fired transition is remembered in the **Simulating History** panel. By clicking the transitions in the history, the Petri net with transits is set back or forth to the corresponding state. Note that many examples have transitions which take tokens but also put them back to the same place. This may look like nothing has changed. The button **RESET** removes all transitions from the history. The button **SHOW DATA FLOW** generates a PDF showing all data flow trees corresponding to the given firing sequence. Note that real tree behavior is only achieved if at least one transition with branching behavior transiting an existing data flow occurs in the firing sequence.

The other items of the **left menu** belong to the layout of the nodes. See [here](#) for the explanations. This layout does not change anything for the input Petri net with transits. The simulated net stays in the state even if the tab is hidden. It only changes when

loading a new net by the **LOAD NET FROM EDITOR** button or when loading a counterexamples (see [here](#)).

Syntax of Flow-LTL:

Flow-LTL consists of two different kind of formulas. The *run formulas* for the control part and the *flow formulas* for the data part. In both, standard LTL can be used with the following operators:

LTL	Syntax
true	TRUE
false	FALSE
atoms	place and transition ids
Negation	NEG or !
Next	X
Conjunction	AND
Disjunction	OR
Implication	IMP or ->
Bimplication	BIMP or <->
Until	U
Weak Until	W
Release	R

Binary operators have to be in brackets, unary operators not, e.g., $(\neg \phi_0 \rightarrow (\phi_1 \text{ AND } G F \phi_2))$. Note that the ids for the places and transitions are more restricted for the formula as for the web interface. For the restrictions please see the detailed syntax below.

The start of a **flow formula** is indicated by the operator 'A'.

For a **Flow-LTL formula** ϕ we allow the syntax

```
 $\phi = \text{LTL} \mid (\phi \text{ AND } \phi) \mid (\phi \text{ OR } \phi) \mid (\text{LTL} \rightarrow \phi) \mid A \text{ LTL}$ 
```

The following picture shows the parser's grammar for Flow-LTL:

```

flowLTL          = runFormula EOF

runFormula
  flowFormula    = ltl | '(' ltl rimp runFormula ')' | runBinary |
runBinary        = '(' runFormula rbin runFormula ')'
flowFormula      = forallFlows ltl

ltl
ltlUnary         = ltlUnary | ltlBinary | tt | ff | atom
ltlUnary         = unaryOp ltl
ltlBinary        = '(' ltl binaryOp ltl ')'

atom             = ID | INT

// LTL
unaryOp          = (ltlFinally | globally | next | neg)
binaryOp         = (and | or | imp | bimp | until | weak | release )
ltlFinally       = 'F'
globally         = 'G'
next             = 'X'
neg              = 'NEG' | '!'
and              = 'AND'
or               = 'OR'
imp              = 'IMP' | '->'
bimp             = 'BIMP' | '<=>'
until            = 'U'
weak             = 'W'
release          = 'R'

// FlowFormula
forallFlows      = 'A'

// RunFormula
rbin             = rand | ror
rand             = 'AND'
ror             = 'OR'
rimp            = 'IMP' | '->'

tt              = 'TRUE'
ff              = 'FALSE'

INT             = '0'..'9'+
ID              = ('a'..'z' | 'A'..'Z' | '_' ) ( 'a'..'z' | 'A'..'Z' | '0'..'9' | '_'
)*


COMMENT         = ( '/' '/' ~ ( '\n' | '\r' ) * | '/' '*' ( . ) * ? '*' '/' )
WS              = ( ' ' | '\n' | '\r' | '\t' )

```

Model Checking Petri Nets with Transits:

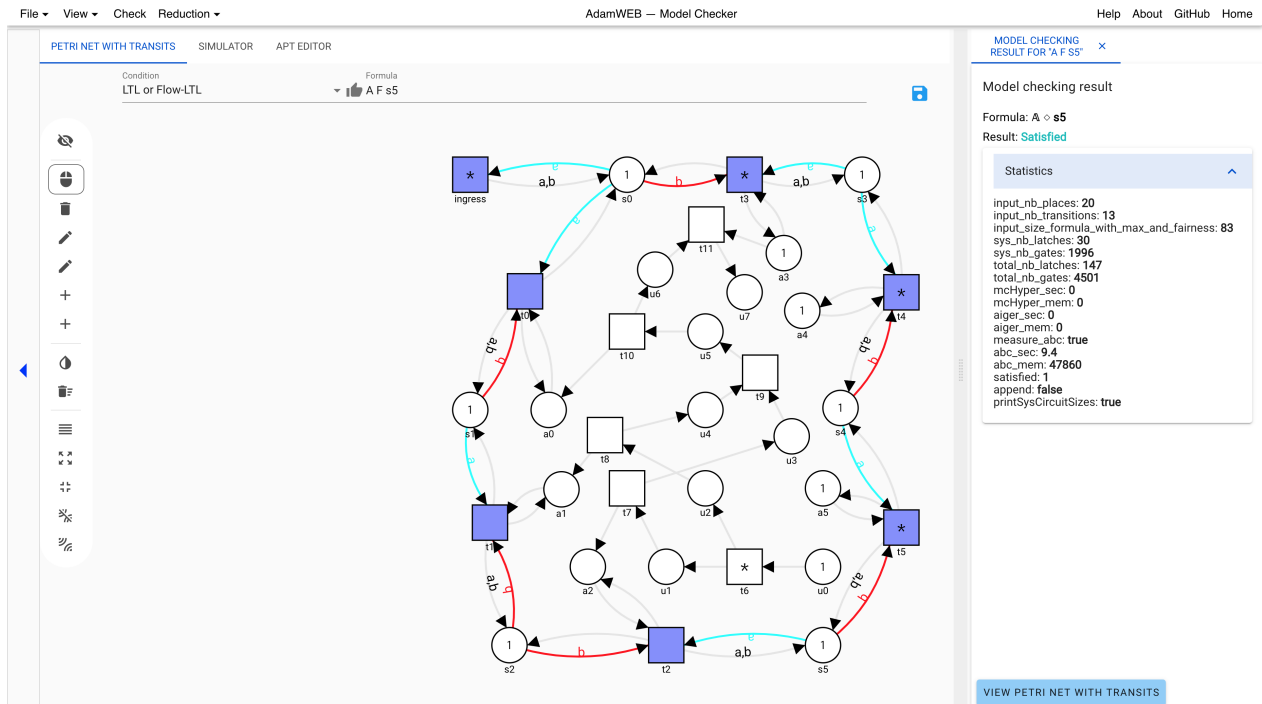
To model check a Petri net with transits against Flow-LTL, a Flow-LTL formula has to be available in the input field:

Condition
LTL or Flow-LTL

Formula
▼  A F s5

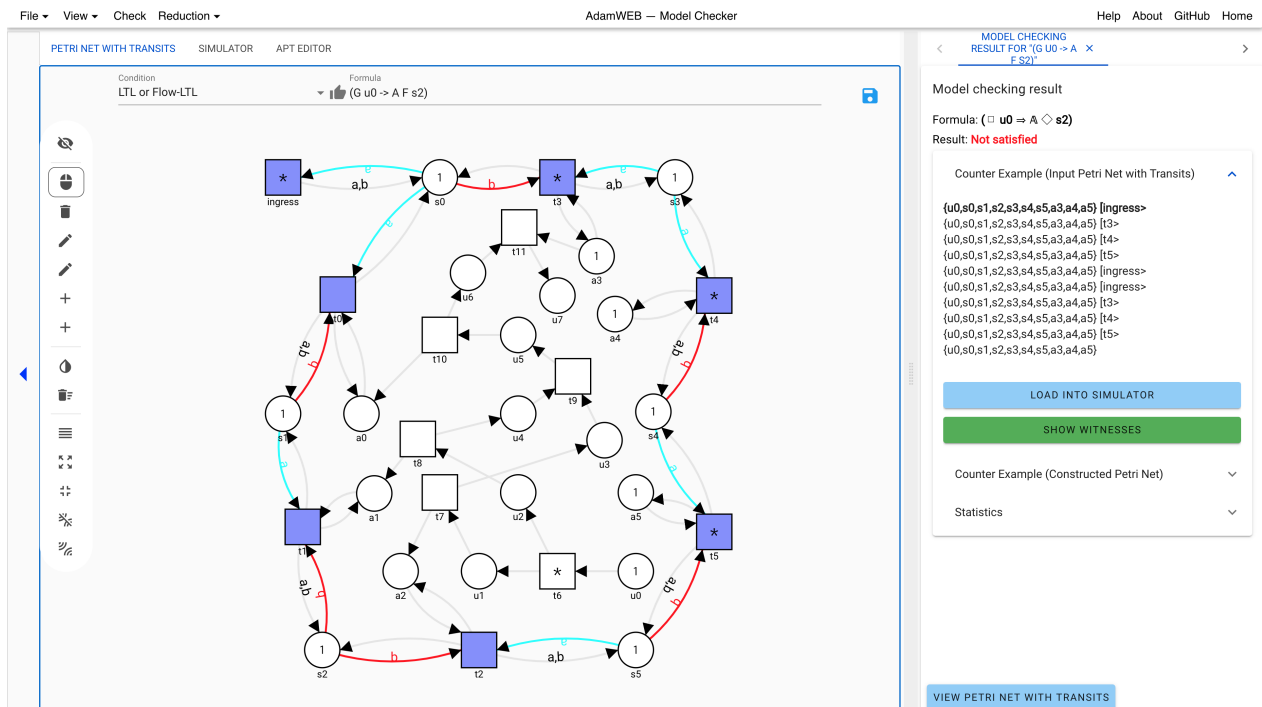
With the drop down menu, you can select to either input a **Flow-LTL formula**, or to select **Büchi**, **Reachability**, or **Safety** as condition. Selecting **Büchi**, **Reachability**, or **Safety** uses the as special toggled places to generate a Flow-LTL formula. The syntax for the Flow-LTL formula can be seen [here](#). Note that the fairness assumptions of the transitions are automatically added while checking the formula and the interleaving maximality assumption is added. Therefore, only runs are considered which satisfy that whenever a transition is enabled, some transition fires.

Clicking the **Check** item of the top menu bar creates a new tab on the right and starts the model checking procedure. For a **satisfying** Flow-LTL formula, the result looks like this:



The slider splitting the left and the right panel can be used to customize the panel sizes. The result and the checked formula is given, and some statistically insights of this model checking run can be shown.

In the negative case, i.e., when the formula is **not satisfied**, the picture looks like this:



Again the formula and the result are printed. In addition, a **counterexample** as a firing sequence which violates the formula is given. When clicking **LOAD INTO SIMULATOR**, this firing sequence is given to the simulator (see [here](#)). With **SHOW WITNESSES**, you can create a PDF showing the witnesses regarding all flow subformulas for this counterexample. This PDF shows the firing sequence of the control and connected to that the violating flow chain for each flow subformula:

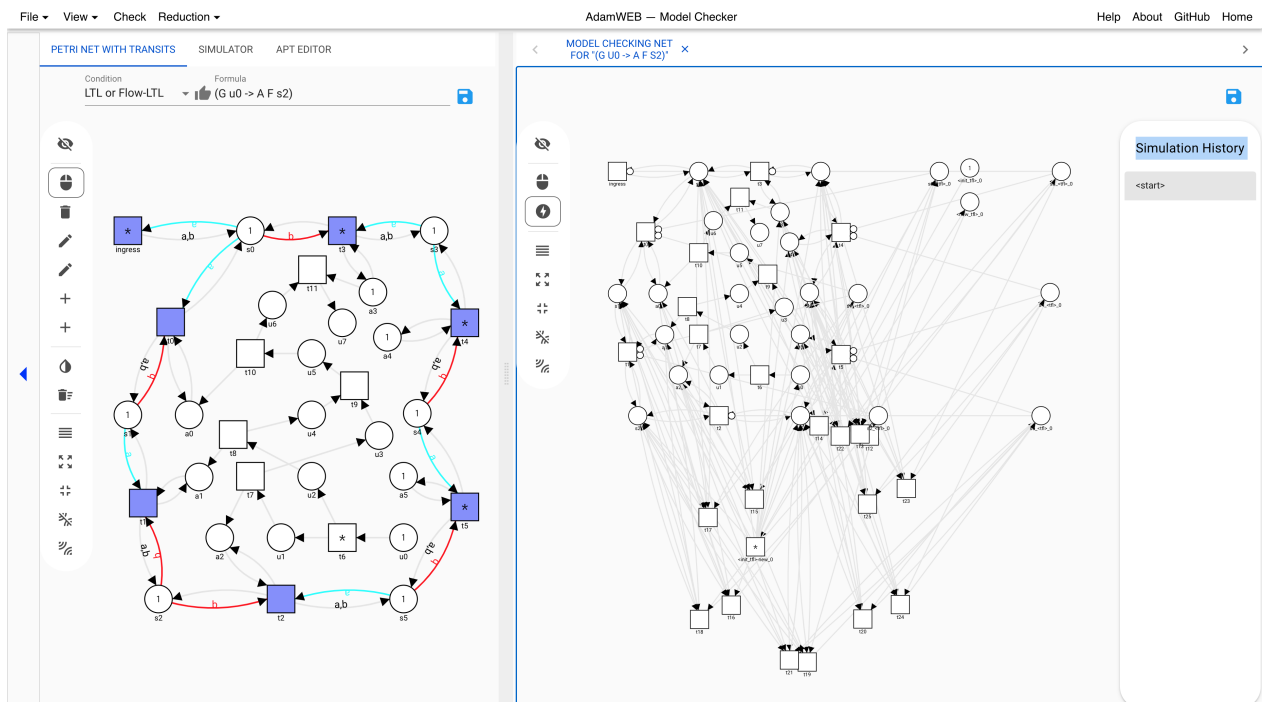


The model checking procedure for model checking Petri nets with transits against Flow-LTL is based on a reduction to model checking Petri net against LTL (see [here](#)). A **counterexample** for this **constructed Petri net** is also given and can be loaded into the simulator with the corresponding Petri net. The technical details regarding the latches of the circuit can be seen when clicking on the counterexample for the constructed Petri net.

Reduction from Petri nets with transits and Flow-LTL to Petri nets and LTL:

In the background, the problem of model checking Petri nets with transits against Flow-LTL is reduced to the problem of model checking Petri nets against LTL. The corresponding constructed parts can be shown with the **Reduction** item of the main menu bar.

The constructed Petri net is shown in the right panel when hitting the subitem **Petri Net** from the main menu bar:



Here, we have the same features as for the simulation tab (see [here](#)) apart from showing the data flow, since a standard Petri net does not have data flows. Clicking the **floppy disk** symbol in the top right corner of the panel allows to save the net in SVG, PNML, or APT file format.

The constructed LTL formula is shown in the right panel when clicking the subitem **LTL Formula** from the main menu bar:

The screenshot shows the AdamWEB Model Checker interface. The main panel displays a Petri net diagram with places (circles) and transitions (squares). The net is labeled with various identifiers like 'ingress', 's0', 't3', 'a3', 't11', 'u6', 'u7', 'a3', 's4', 't10', 'u5', 't9', 's4', 't8', 'a1', 't7', 'u2', 'u3', 's4', 't5', 's2', 't2', 'u0', 't6', 'u0', 't5', 's5'. The right panel shows the LTL formula: $X (\neg \text{init_tfl} \rightarrow_0 \text{OR} (G u0 \text{IMP} (G \neg \text{new_tfl} \rightarrow_0 \text{OR} (\neg \text{new_tfl} \rightarrow_0 U (\neg \text{new_tfl} \rightarrow_0 \text{AND} F s2_tfl \rightarrow_0))))))$.

Text Editor:

A text editor is provided to change and also edit the input Petri net with transits:

The screenshot shows the AdamWEB Model Checker interface with the APT Editor tab selected. The text editor contains the following content:

```
.name "SDN-handbuildExample"
.description "The motivating example from the TACAS21 paper and the update from configuration [s0fwdTos3, s3fwdTos4, s4fwdTos5] to configuration [s0fwdTos1, s1fwdTos2, s2fwdTos5].
    Satisfiable formula:  A F s5
    Unsatisfiable formula: (G u0 -> A F s2)"
.type LPN
.options
.condition="LTL"

.places
a0[yCoord=390.0, xCoord=510.0]
a1[yCoord=511.51, xCoord=529.6]
a2[yCoord=630.0, xCoord=570.0]
a3[yCoord=150.0, xCoord=870.0]
a4[yCoord=238.23, xCoord=898.26]
a5[yCoord=510.0, xCoord=930.0]
s0[yCoord=30.0, xCoord=630.0]
s1[yCoord=390.0, xCoord=390.0]
s2[yCoord=750.0, xCoord=450.0]
s3[yCoord=30.0, xCoord=990.0]
s4[yCoord=386.98, xCoord=956.83]
s5[yCoord=750.0, xCoord=930.0]
u0[yCoord=630.0, xCoord=930.0]
u1[yCoord=630.0, xCoord=690.0]
u2[yCoord=510.0, xCoord=750.0]
u3[yCoord=430.4, xCoord=882.06]
u4[yCoord=390.0, xCoord=750.0]
u5[yCoord=270.0, xCoord=750.0]
u6[yCoord=175.31, xCoord=673.11]
u7[yCoord=210.0, xCoord=810.0]

.transitions
```

For the format please refer to [here](#) (Section 4.1).