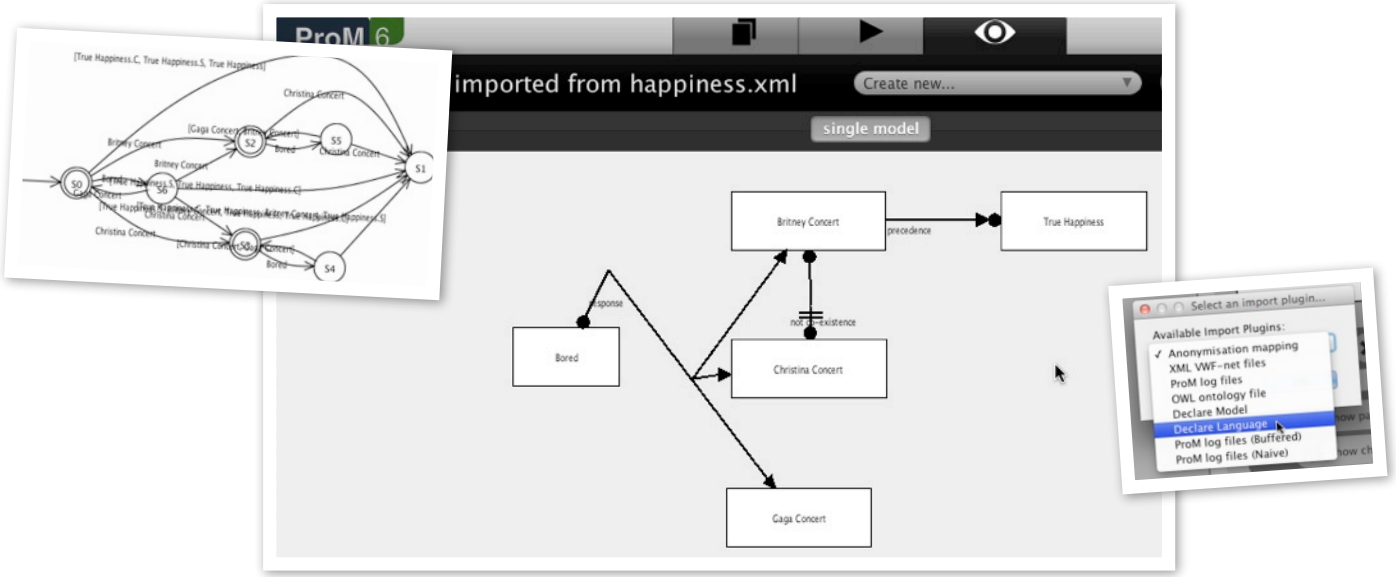


Declare

Michael Westergaard

Tuesday, July 19, 2011

The Declare plug-in makes it possible to load and show Declare models in ProM. The package also contains plug-ins for translating Declare models to other representations, most importantly LTL formulae and corresponding finite automata and several derived types.

Aside from the import facilities, this plug-in is mostly for developers.

Importing and Showing Stuff

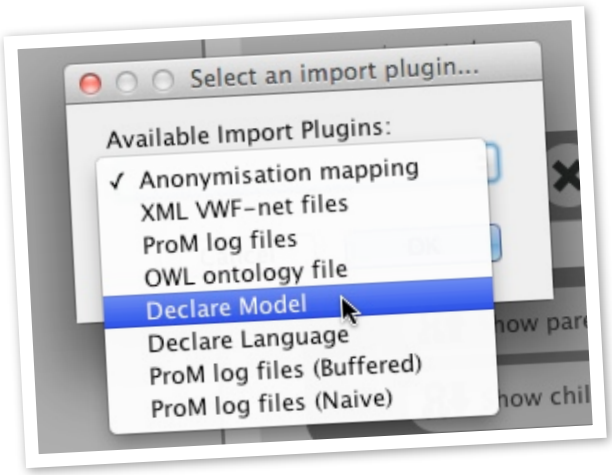
This plug-in can load Declare models and Declare languages. Models are obviously a good thing (or you wouldn't be here) and languages are useful for things constructing Declare models, such as a theoretical Declare miner.

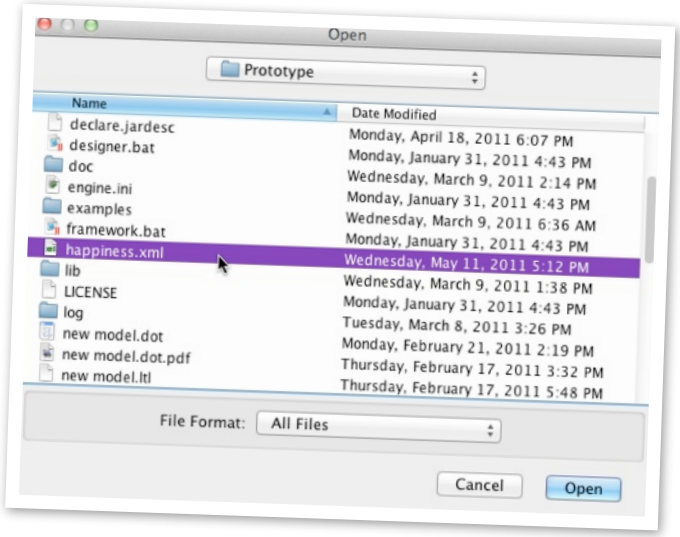
Importing Declare Models

Importing a Declare model is as easy as 1, 2, 3. Except it has one more step:

1. Find the Import button that's in the ProM Workspace. Feel free to click on it and proceed to the next step. You can also not click on it and instead go do boring stuff.
2. Select the file you want to open. Your file is probably less awesome than the one in my screen shot, but you just have to make do. Then give that Open button some clicky action.

3. You now receive a dialog prompting you for the file type. The smart choice is Declare Model.



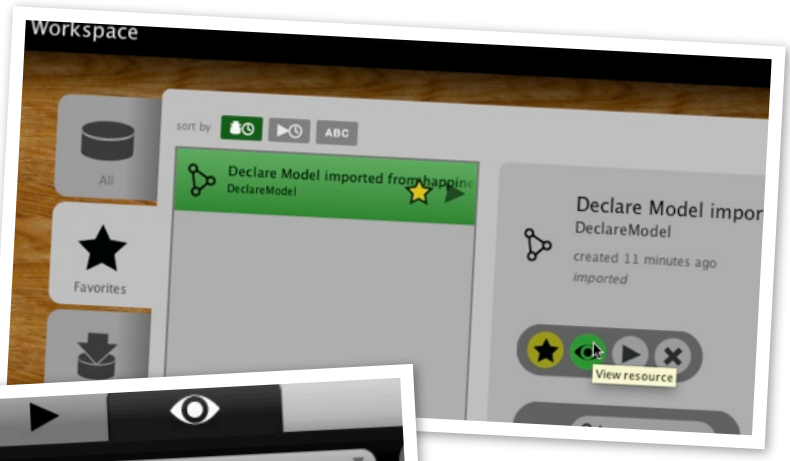


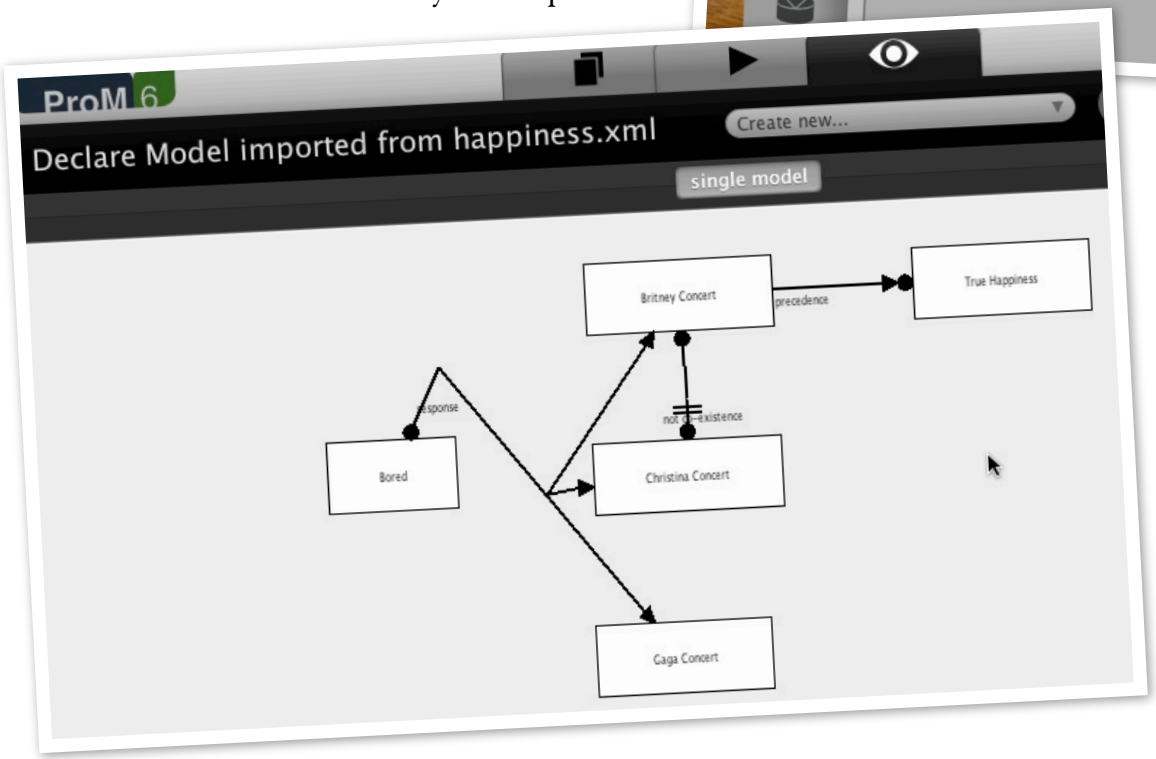
4. I lied about step 4.

Viewing Declare Models

If you create your Declare model using another plug-in, it may be shown to you automatically, curtesy of the fairy automatically showing stuff inside ProM. Otherwise, you may follow this delicate recipe:

1. Select your favorite Declare model in the Workspace. Alternatively, select the one you want to display.
2. With the mouse cursor, bang the View Resource button. It is recognized by not really being a button and not saying View Resource, but instead having an eye icon and a tool tip.
3. You now see your Declare model – specially rendered in 2D just for you! Doesn't that make you feel special?





Importing Declare Languages

To import a Declare language, we do the exact same in every detail as when importing a Declare model, except we make some slight changes here and there:

1. Find the Import button that's in the ProM Workspace. Feel free to click on it and proceed to the next step. You can also not click on it and instead go do boring stuff.
2. Select the file you want to open. Your file is probably less awesome than the one in my screen shot, but you just have to make do. Then give that Open button some clicky action.
3. You now receive a dialog prompting you for the file type. The smart choice is Declare Language.

You cannot visualize the language. That would just be silly, wouldn't it?

Translating Stuff to Other But Similar Stuff

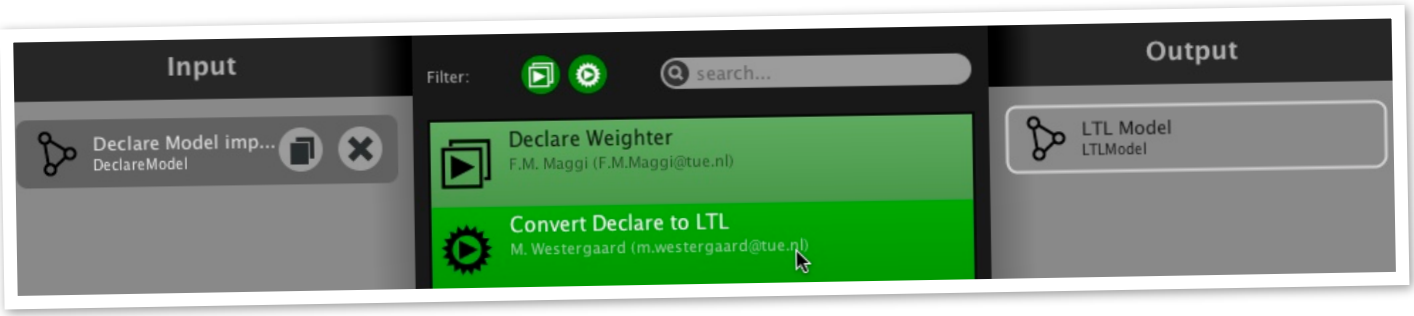
Declare models can, in one or more steps, be translated to other things. Zero steps, if we consider translation to be reflexive.

We shall assume that readers can start a plug-in. This is done by:

1. Select the source in the Workspace.
2. Click the Use resource button, which much like the view resource is known by being a non-buttonny thing with a triangle.
3. Select the appropriate plug-in. Green ones have enough inputs to be executed as is, yellow ones need more inputs and invisible ones are not compatible with your selected input.
4. Click Start to win the lottery xor starting the selected plug-in.

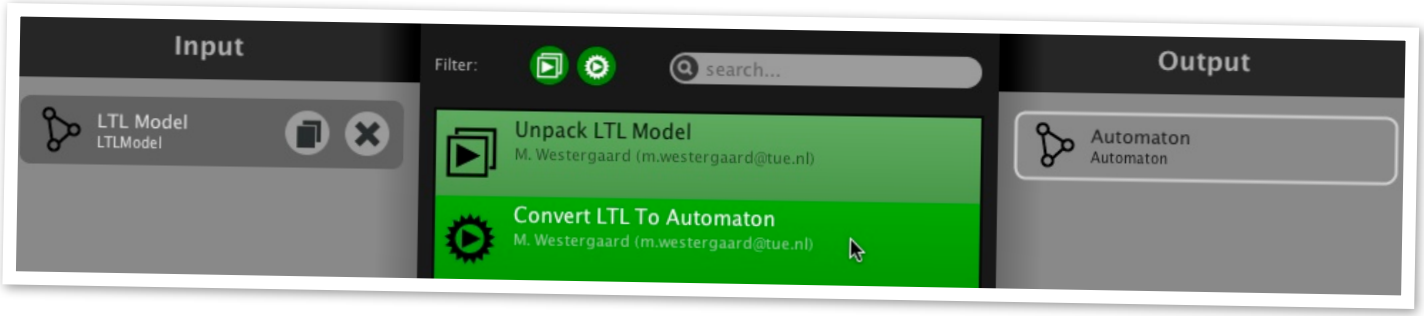
Declare model to LTL model

Use a Declare model as input and get a LTL model. LTL models are an abstract representation of an LTL expression, i.e., it contains a representation of a LTL formula with same behavior as the given Declare model.



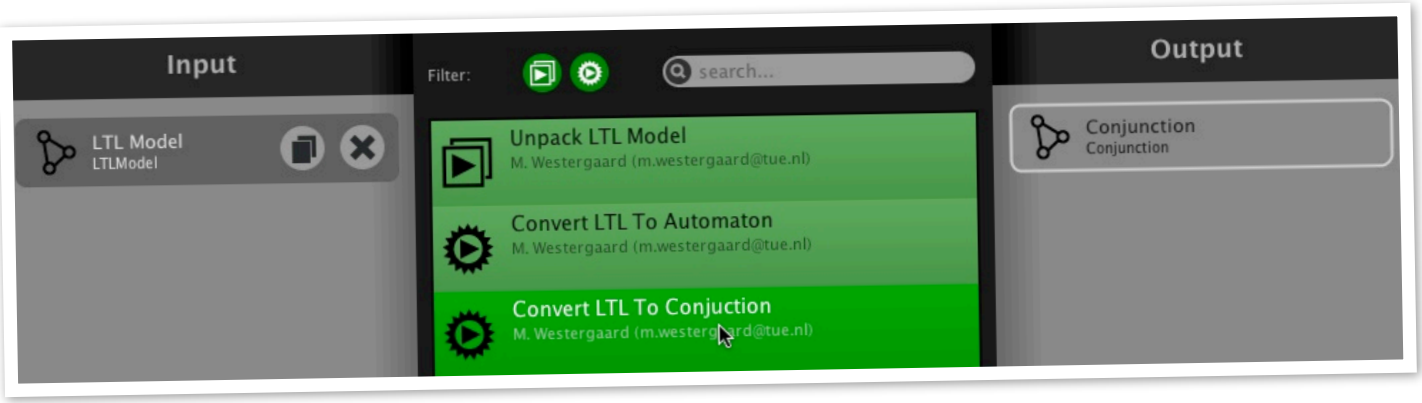
LTL model to Automaton

Use a LTL model as input and get a Automaton as output. An automaton is a finite automaton which can be used for executing the semantics of the given LTL model.



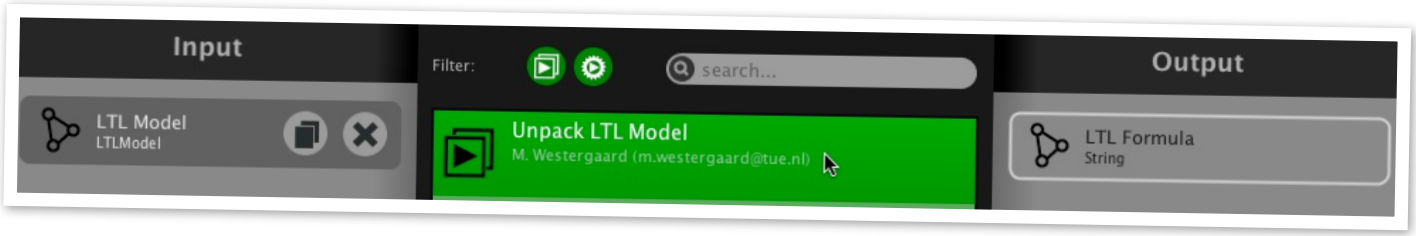
LTL model to Conjunction

Use a LTL model as input and get a Conjunction. Conjunctions are a more efficient representation of LTL formula with many conjunctions, which facilitates more efficient translation to automata. Conjunctions may be used internally by other translations.



Unpack LTL formula

Use a LTL model as input and get a string representation of the formula in the syntax used by Declare. Useful for debugging.



LTL formula to Automaton

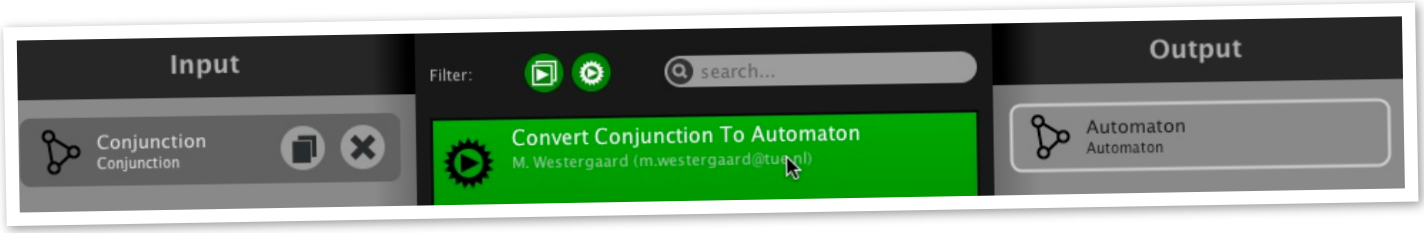
Same as converting an LTL model to an automaton, except the input instead is just a string representation of the formula.

LTL formula to Conjunction

Same as converting an LTL model to a conjunction, except the input instead is just a string representation of the formula.

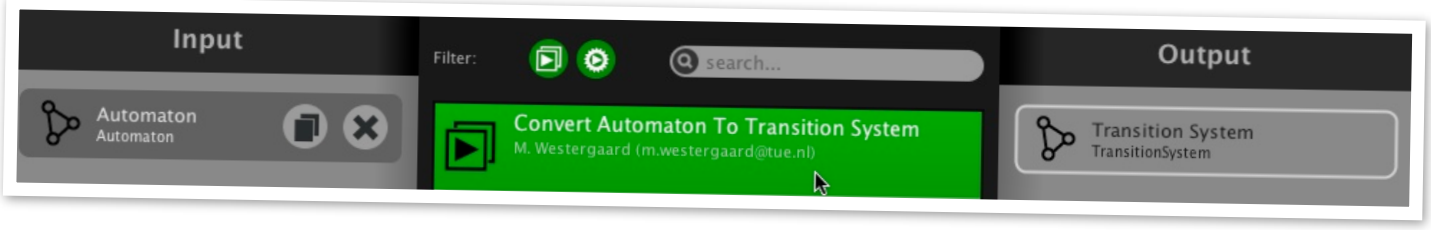
Conjunction to Automaton

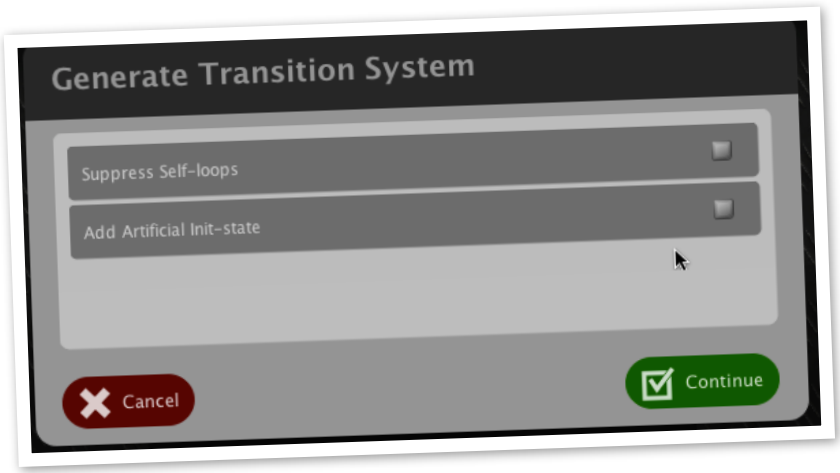
Translate a conjunction to a corresponding automaton.



Automaton to Transition System

A transition system can be used to visualize an automaton or as input for various mining or replay algorithms in ProM. This translation is slightly more involved than the others, as it requires

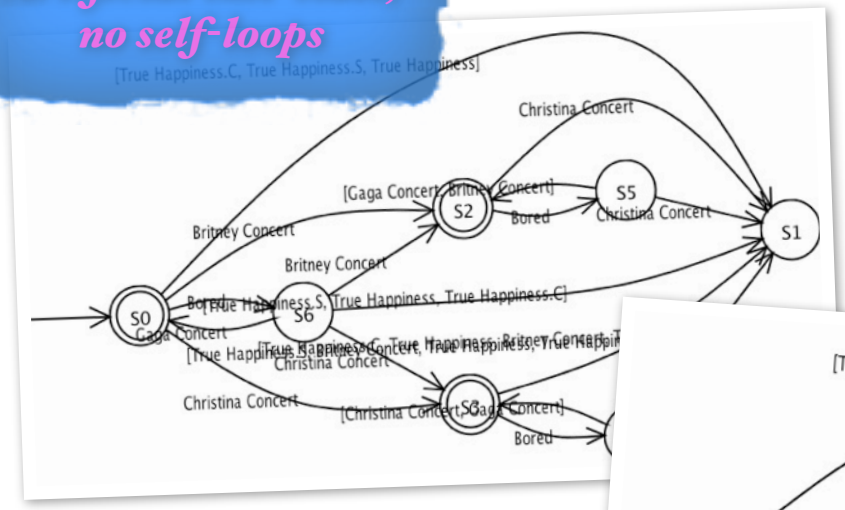




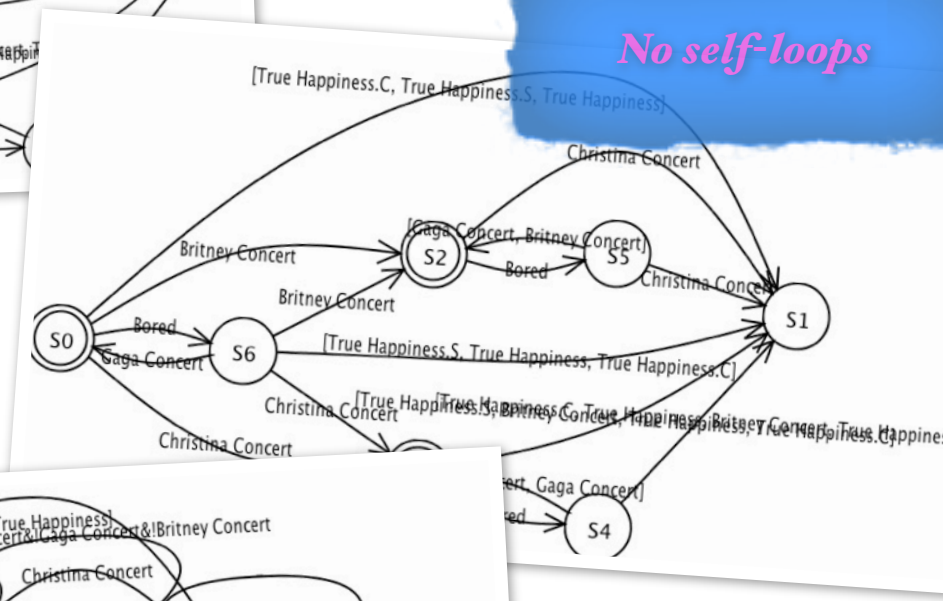
some parameters depending on whether you want to use the result for visualization or for algorithmic purposes. For visualization, adding an artificial init-state is useful as it makes it easier to see the first state of the automaton, and suppressing self-loops also makes the result more human readable, as they can be inferred from the other arcs in most cases. Both of these should be switched off when using the model algorithmically.

Translating the model on page 2 yields the 3 results below, depending on the used options (we can also make one with artificial init-state and suppressed self-loops).

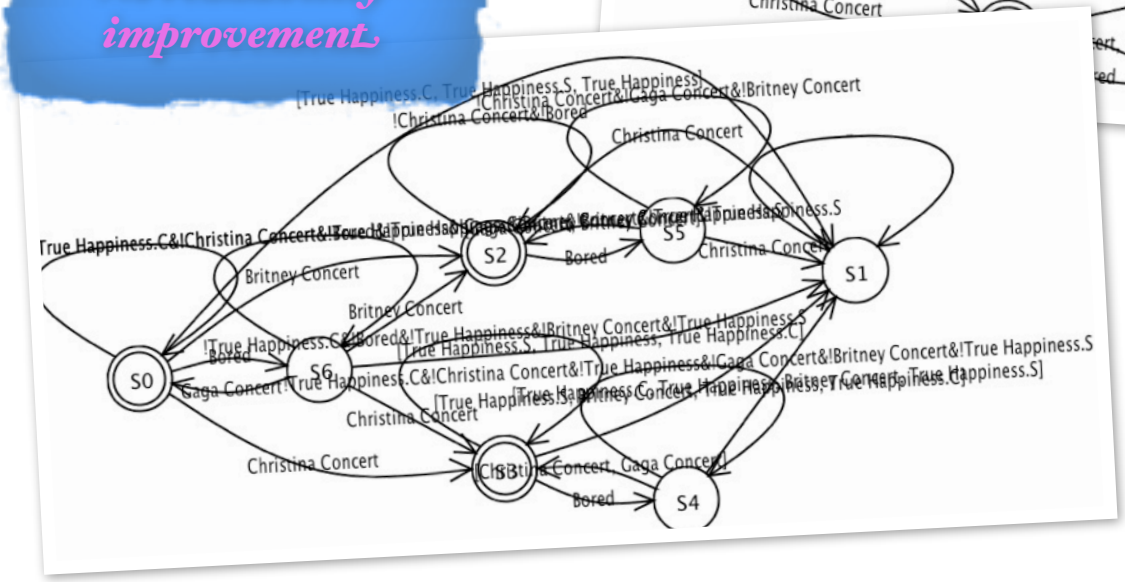
Artificial init-state, no self-loops

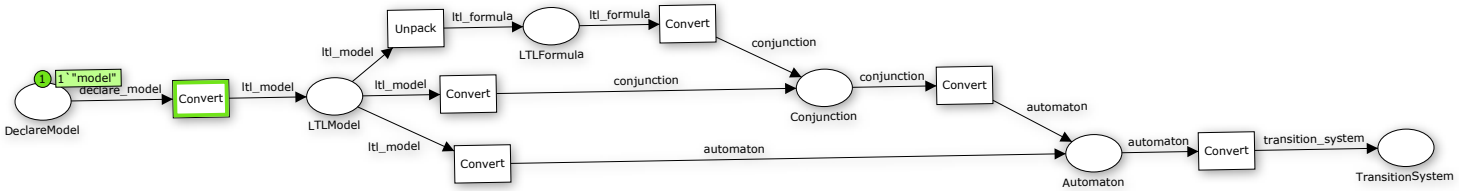


No self-loops



No readability improvement





At the top of this page, you find a CPN model illustrating all possible conversions. At the bottom of the page, you see one of my desktop backgrounds, as Apple Pages is obscuring it.