Automata-Theoretic Model-Checking for LTL

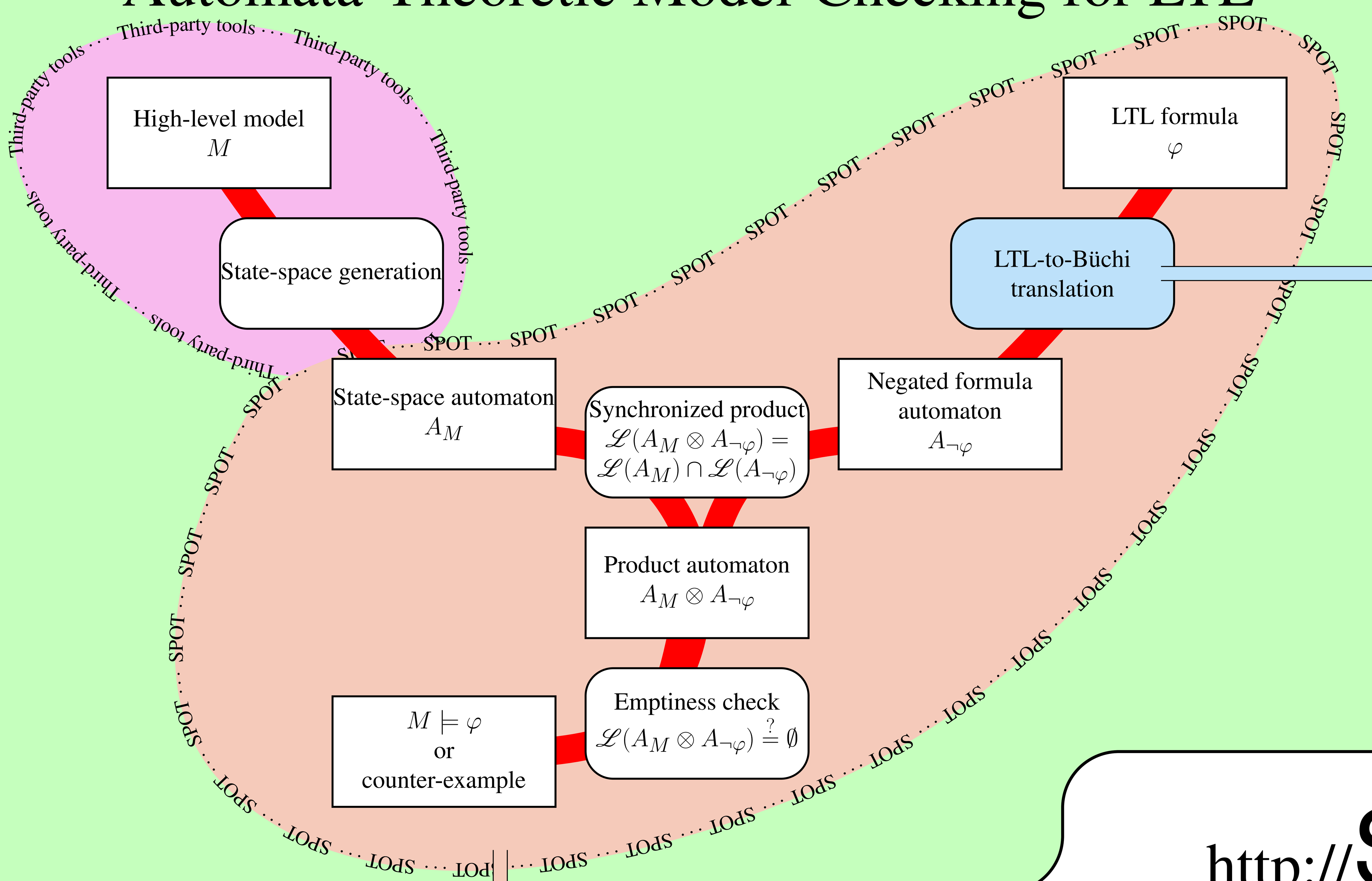


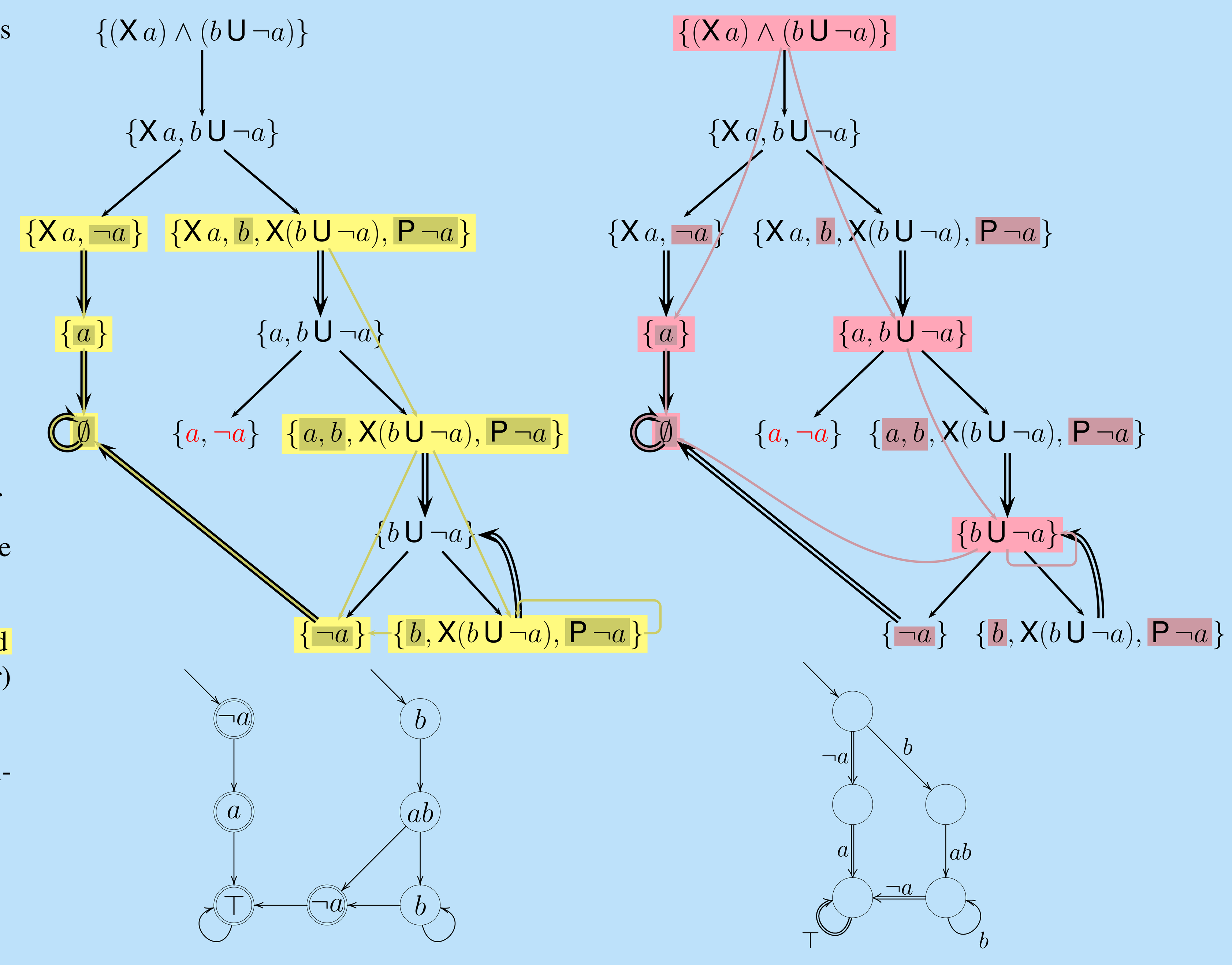
Tableau Methods for LTL: States vs. Transitions

1. Develop a satisfaction tree using tableau rules until no more rule can be applied.

formula	1st child	2nd child
$\neg\top$	$\{\bot\}$	
$\neg\bot$	$\{\top\}$	
$\neg\neg f$	$\{f\}$	
$f \wedge g$	$\{f, g\}$	
$f \vee g$	$\{f\}$	$\{g\}$
$\neg(f \wedge g)$	$\{\neg f\}$	$\{\neg g\}$
$\neg(f \vee g)$	$\{\neg f, \neg g\}$	
$\neg \mathsf{X} f$	$\{\mathsf{X} \neg f\}$	
$f \mathsf{U} g$	$\{g\}$	$\{f, \mathsf{X}(f \mathsf{U} g), \mathsf{P}\, g\}$
$\neg(f \mathsf{U} g)$	$\{\neg f, \neg g\}$	$\{\neg g, \mathsf{X} \neg(f \mathsf{U} g)\}$

$\mathsf{P}\, g$ promises that g will be fulfilled eventually.

2. For each leaf of the tree, develop the X formulae ($\Longrightarrow$) recursively, identifying common nodes.
3. Use subtree leaves to construct a state-based Büchi automaton, roots to construct a (smaller) transition-based Büchi automaton.
4. Complement each promise ($\mathsf{P}\, g$) to define generalized acceptance sets.



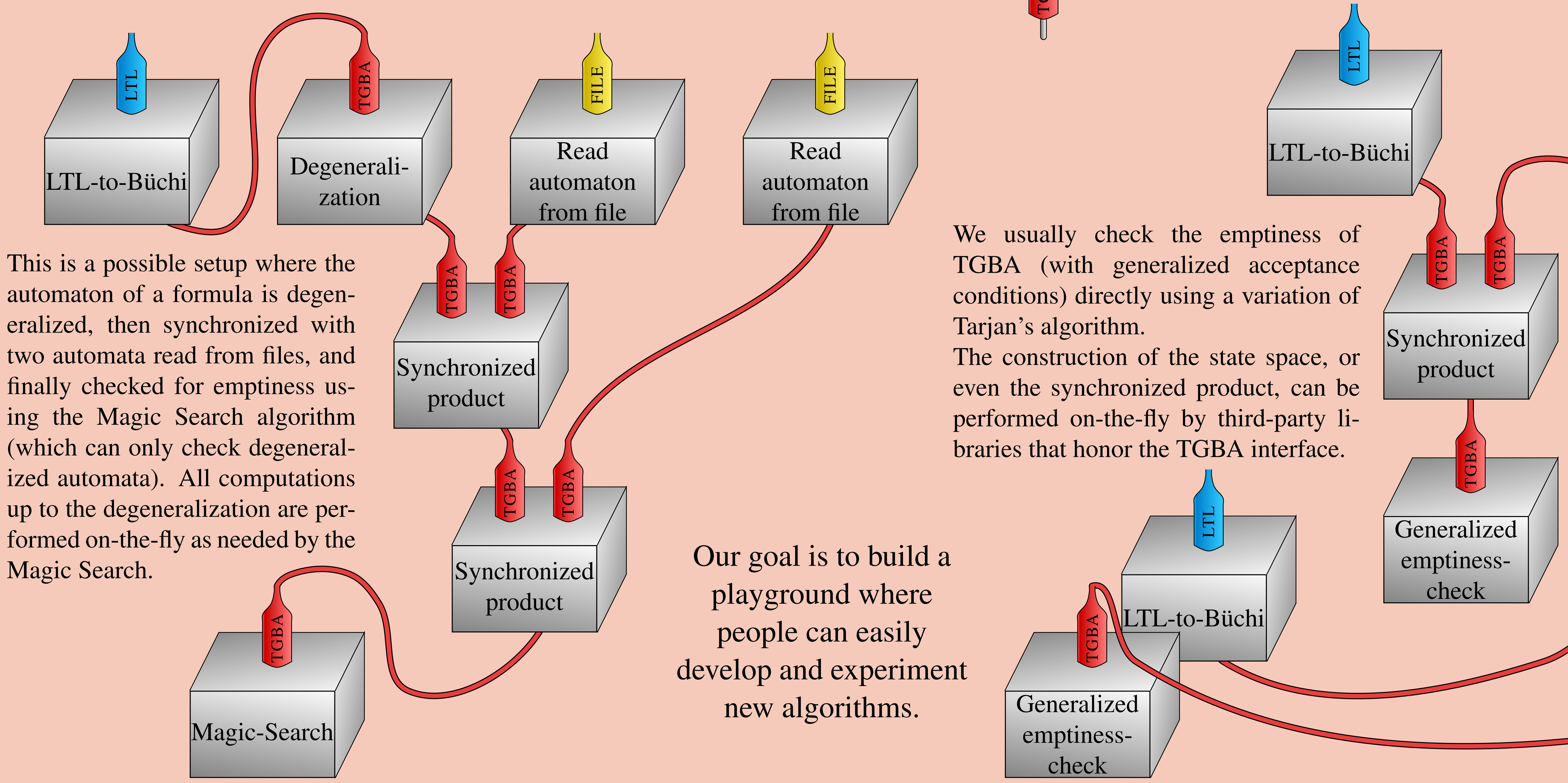
We handle Transition-based Generalized Büchi Automata (TGBA) in SPOT because they can be used to construct shorter automata from LTL formulae. Furthermore any state-based automaton can be represented as a TGBA without growth in size (the converse is false).

http://SPOT.lip6.fr
A Model-Checking Library

Alexandre.Duret-Lutz@lip6.fr

A Library of Reusable Bricks

Uses a *Transition-based Generalized Büchi Automata* interface supporting on-the-fly computations: TGBA.

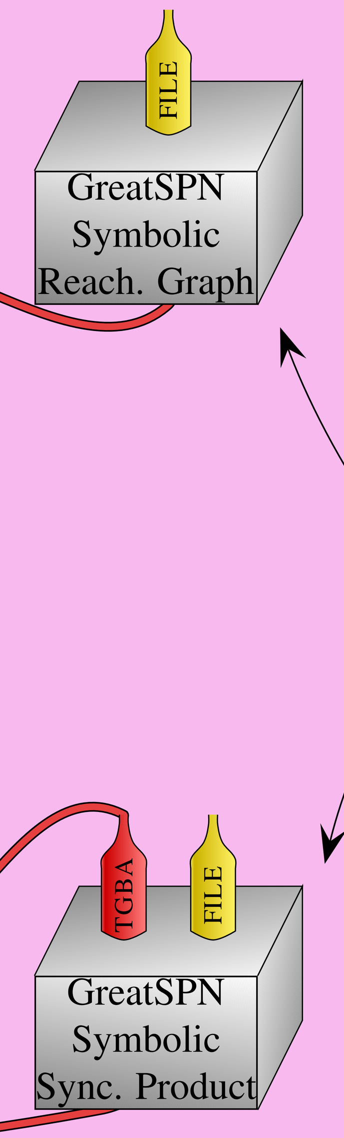


This is a possible setup where the automaton of a formula is degeneralized, then synchronized with two automata read from files, and finally checked for emptiness using the Magic Search algorithm (which can only check degeneralized automata). All computations up to the degeneralization are performed on-the-fly as needed by the Magic Search.

We usually check the emptiness of TGBA (with generalized acceptance conditions) directly using a variation of Tarjan's algorithm.
The construction of the state space, or even the synchronized product, can be performed on-the-fly by third-party libraries that honor the TGBA interface.

Our goal is to build a playground where people can easily develop and experiment new algorithms.

Interfacing Third-Party Tools



We have kept the state-space generation outside SPOT to be independent of the high-level modeling formalism (Petri net, Promela, etc.).

To model-check some specification, you should find a tool that can read your modeling formalism and generate its state space. Then equip this tool so it can produce TGBA (preferably on-the-fly), and connect it to SPOT.

We currently have interfaces for several flavors of GreatSPN[1] (*University of Turin*), which inputs well-formed Petri nets.

- In the symbolic reachability graph, **global** symmetries of the Petri net are exploited to "fold" the accessibility graph and reduce the state space.
- The symbolic synchronized product is more involved: symmetries are computed **locally**, for the transitions being synchronized. The interface is here a synchronized product driven by the formula automaton.

With the authors of Quasar[2] (Cedric/CNAM) we are looking how to interface their tool with SPOT. Quasar analyses Ada programs and can perform structural reductions, as well as automatic abstractions according to the properties to be verified.

[1] http://www.di.unito.it/~greatspn/
[2] http://quasar.cnam.fr/

UNIVERSITE PIERRE & MARIE CURIE
LA SCIENCE A PARIS
LIP6
CNRS CENTRE NATIONAL DE LA RECHERCHE SCIENTIFIQUE