Mechanizing the Minimization of Deterministic Generalized Büchi Automata

Souheib Baarir[1,2] Alexandre Duret-Lutz[3]

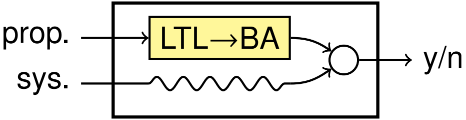
[1]Université Paris Ouest Nanterre la Défense, Nanterre, France
[2]Sorbonne Universités, UPMC Univ. Paris 6, UMR 7606, LIP6, Paris, France
souheib.baarir@lip6.fr

[3]LRDE, EPITA, Le Kremlin-Bicêtre, France
adl@lrde.epita.fr

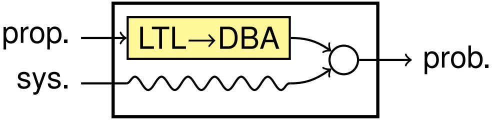
FORTE'14, 3–5 June 2014

Context

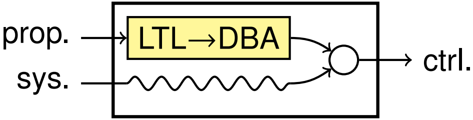
Model checking



Prob. model checking

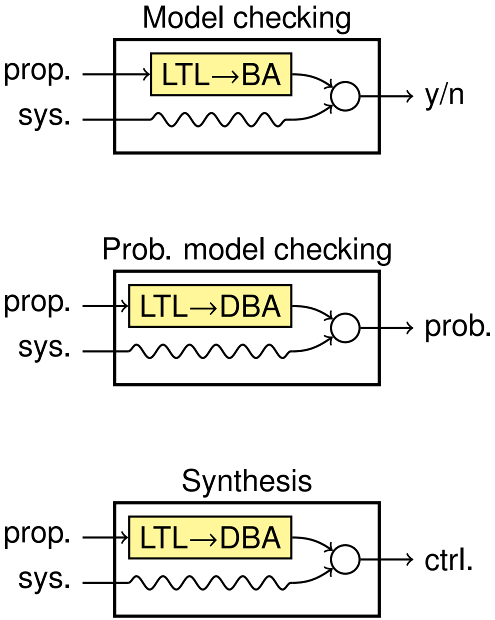


Synthesis



- Büchi Automata are used in many formal methods, but with different requirements.

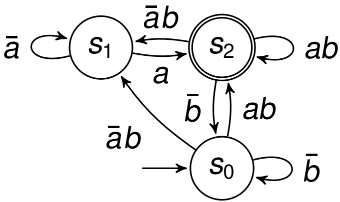
Context



- Büchi Automata are used in many formal methods, but with different requirements.
- Small [D]BA helps
  - Minimization (NP-comp.),
  - Simulation-based algorithms,
  - **generalized acceptance**,
  - **transition-based acceptance**.

Transion-based Generalized Acceptance

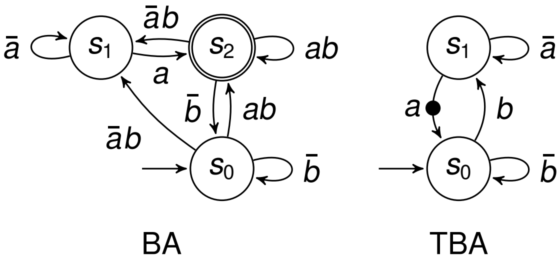
Minimal Büchi automaton for $\mathsf{GF}a \land \mathsf{GF}b$:



BA

Transion-based Generalized Acceptance

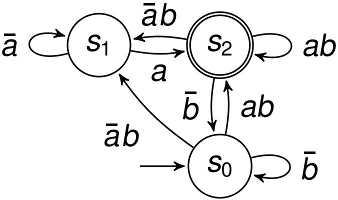
Minimal automata for $\mathsf{G}\mathsf{F}a \wedge \mathsf{G}\mathsf{F}b$:



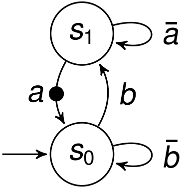
Using Transition-based and Generalized acceptance allows more compact automata.

Transion-based Generalized Acceptance

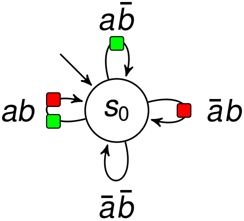
Minimal automata for GF$a \wedge$ GFb:



BA



TBA

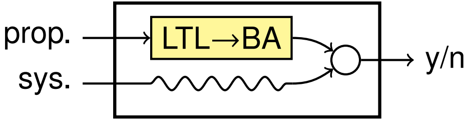


TGBA
with $\mathcal{F} = \{\color{green}\blacksquare, \color{red}\blacksquare\}$

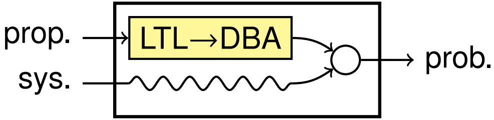
Using Transition-based and Generalized acceptance allows more compact automata.

Objective

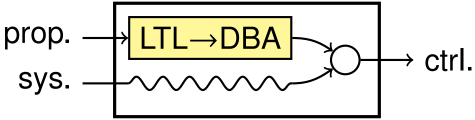
Model checking



Prob. model checking

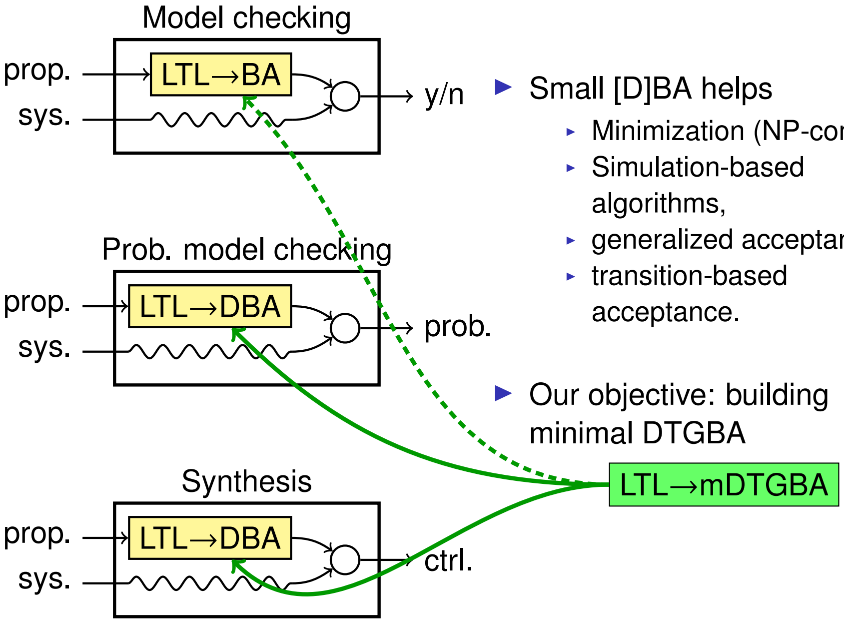


Synthesis



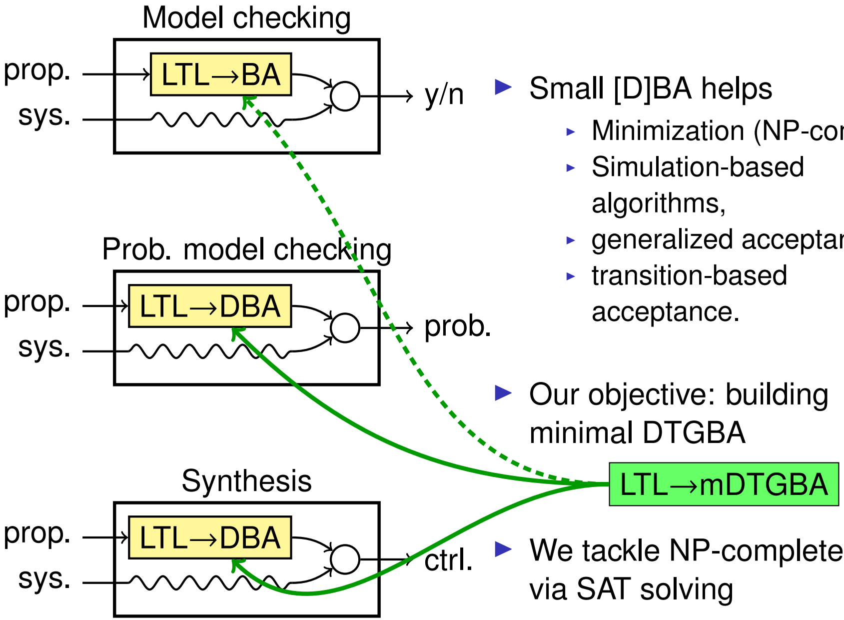
- Small [D]BA helps
  - Minimization (NP-comp.),
  - Simulation-based algorithms,
  - generalized acceptance,
  - transition-based acceptance.
- Our objective: building minimal DTGBA

Objective



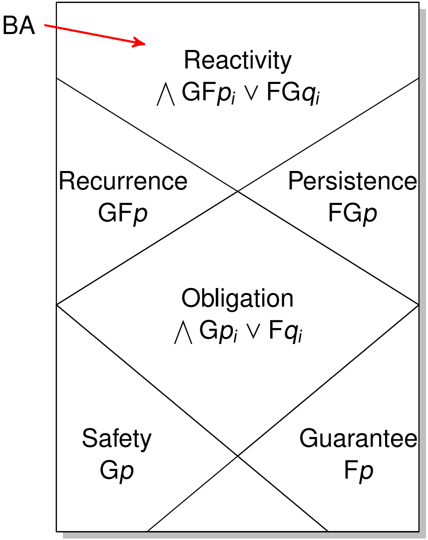
- Small [D]BA helps
  - Minimization (NP-comp.),
  - Simulation-based algorithms,
  - generalized acceptance,
  - transition-based acceptance.
- Our objective: building minimal DTGBA

Objective



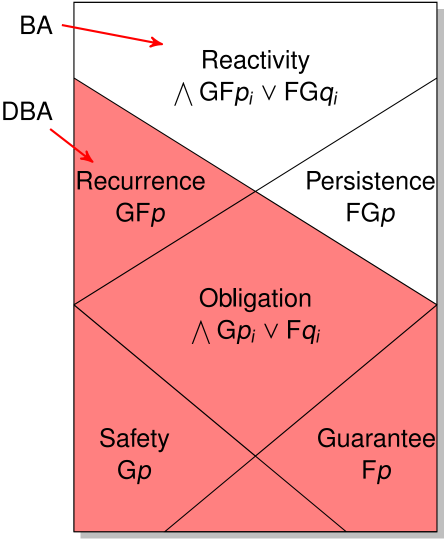
- Small [D]BA helps
  - Minimization (NP-comp.),
  - Simulation-based algorithms,
  - generalized acceptance,
  - transition-based acceptance.
- Our objective: building minimal DTGBA
- We tackle NP-completeness via SAT solving

LTL Hierarchy: Determinization & Minimization



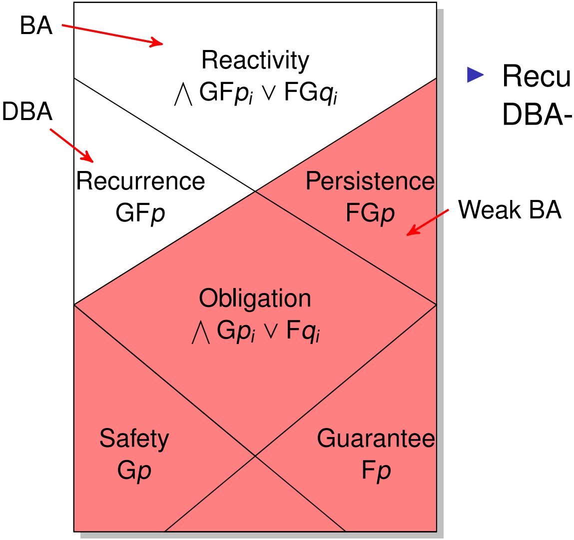
Z. Manna and A. Pnueli. A hierarchy of temporal properties. PODC'90

LTL Hierarchy: Determinization & Minimization



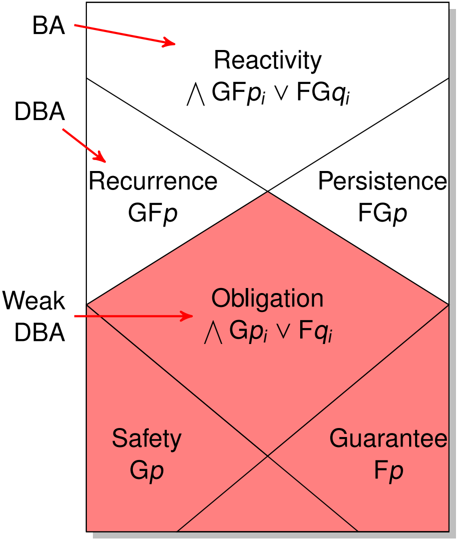
- Recurrence properties are DBA-realizable. (E.g. via Rabin)

LTL Hierarchy: Determinization & Minimization



- Recurrence properties are DBA-realizable. (E.g. via Rabin)

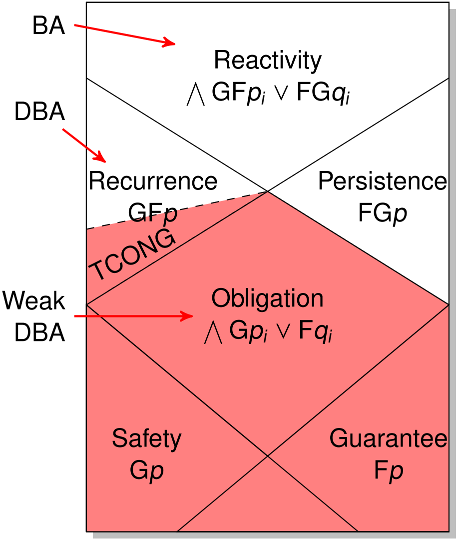
LTL Hierarchy: Determinization & Minimization



- Recurrence properties are DBA-realizable. (E.g. via Rabin)
- WDBA can be minimized in polynomial time.

C. Dax, J. Eisinger, and F. Klaedtke. Mechanizing the powerset construction for restricted classes of ω-automata. ATVA'07

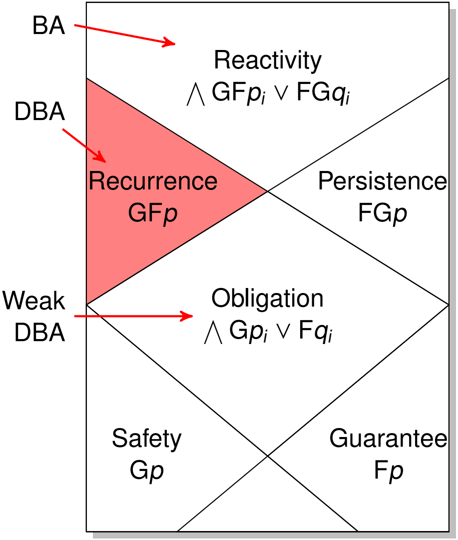
LTL Hierarchy: Determinization & Minimization



- Recurrence properties are DBA-realizable. (E.g. via Rabin)
- WDBA can be minimized in polynomial time.
- Some recurrences (the TCONG class) can always be determininized to DTBA by powerset construction.

C. Dax, J. Eisinger, and F. Klaedtke. Mechanizing the powerset construction for restricted classes of ω-automata. ATVA'07

LTL Hierarchy: Determinization & Minimization

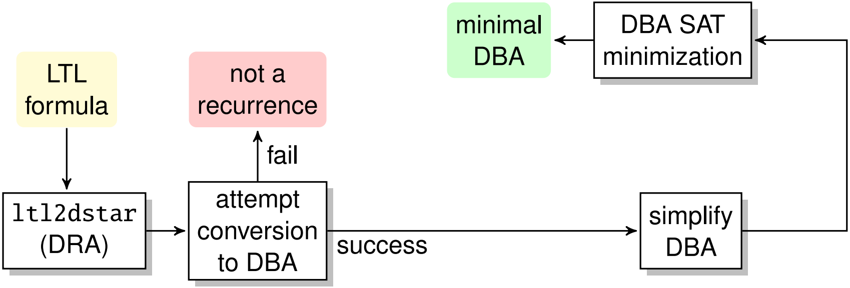


- Recurrence properties are DBA-realizable. (E.g. via Rabin)
- WDBA can be minimized in polynomial time.
- Some recurrences (the TCONG class) can always be determininized to DTBA by powerset construction.
- So far, no technique for:
  - Determinization of TGBA,
  - Minimization of DTGBA.

C. Dax, J. Eisinger, and F. Klaedtke. Mechanizing the powerset construction for restricted classes of ω-automata. ATVA'07

From LTL to Minimal D[T][G]BA

Output: DBA. (Ehlers' setup.)

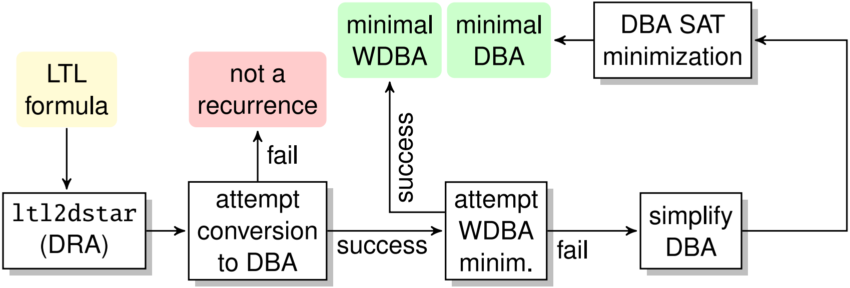


R. Ehlers. Minimising DBA precisely using SAT solving. SAT'10

S. C. Krishnan et al. Deterministic ω-automata vis-a-vis DBA. ISAAC'94

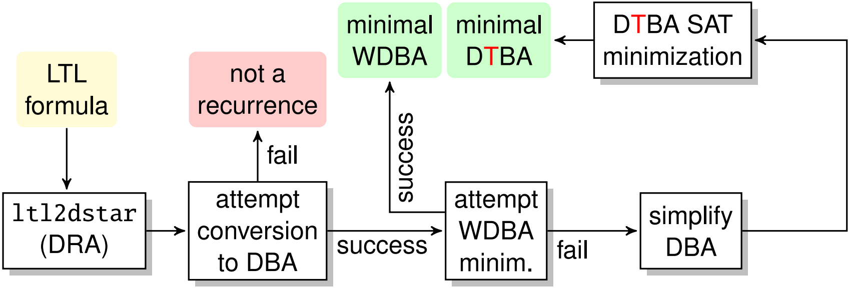
From LTL to Minimal D[T][G]BA

Output: DBA.



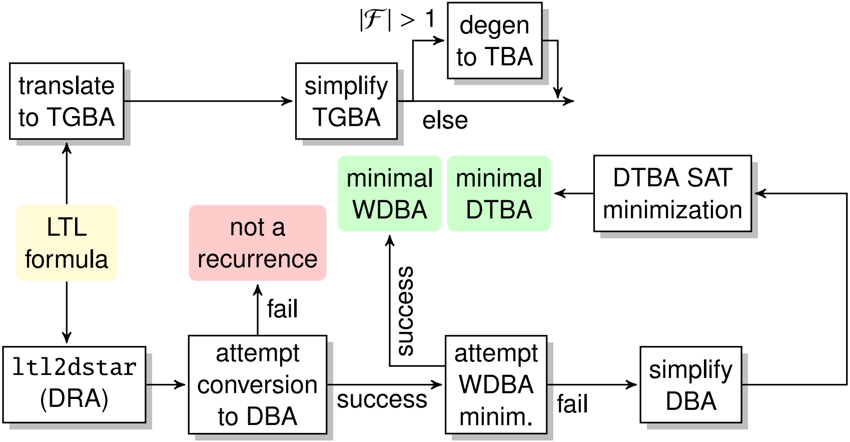
From LTL to Minimal D[T][G]BA

Output: DTBA.



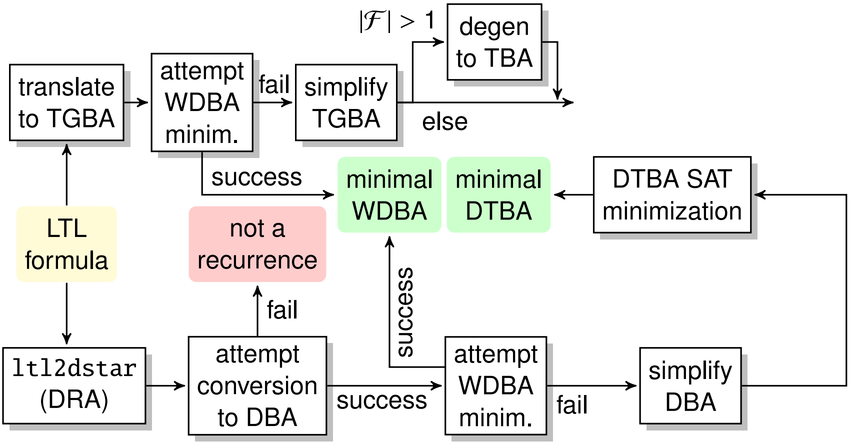
From LTL to Minimal D[T][G]BA

Output: DTBA.



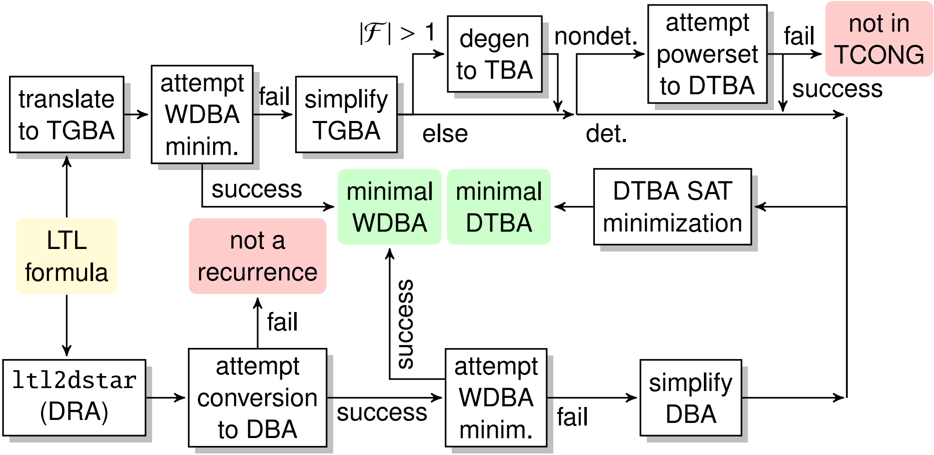
From LTL to Minimal D[T][G]BA

Output: DTBA.



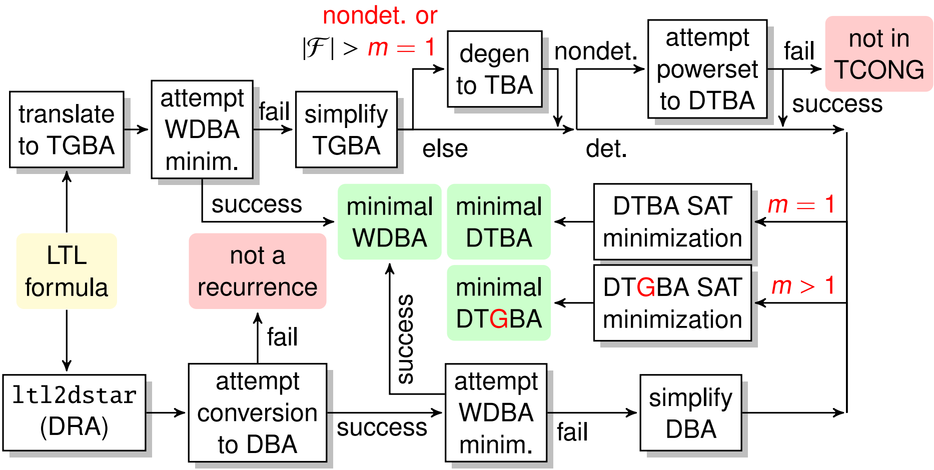
From LTL to Minimal D[T][G]BA

Output: DTBA.



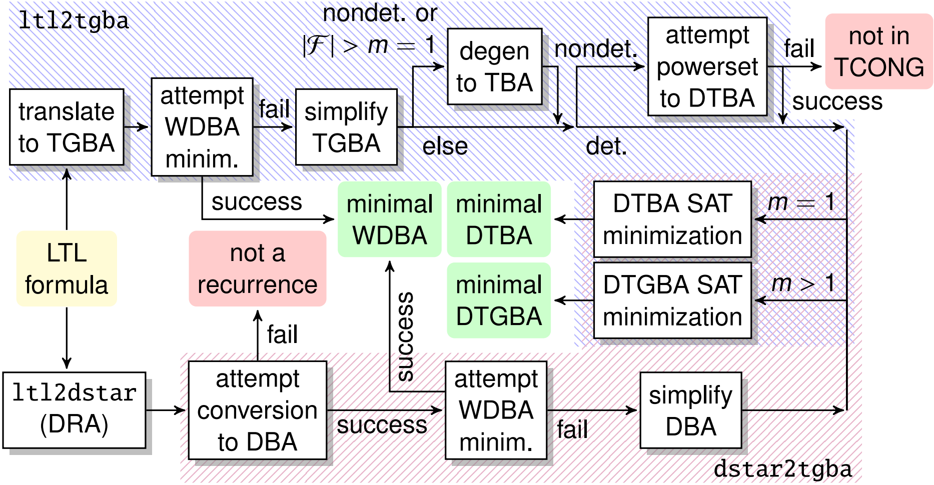
From LTL to Minimal D[T][G]BA

Output: DTGBA ($m > 1$) or DTBA ($m = 1$).



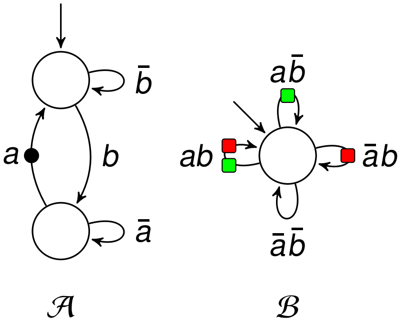
From LTL to Minimal D[T][G]BA

Output: DTGBA ($m > 1$) or DTBA ($m = 1$). Our setup.



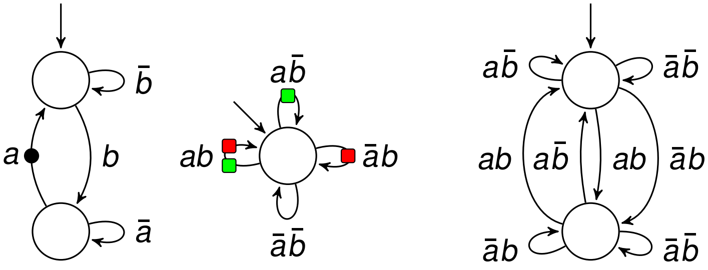
SAT-based Minimization

Equivalence Check of Two DTGBA



Equivalence Check of Two DTGBA

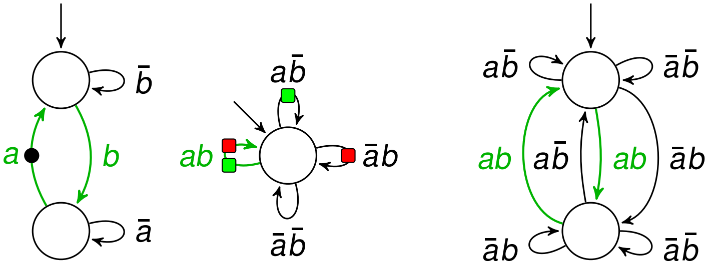
Two **complete** DTGBA $\mathcal{A}$ and $\mathcal{B}$ are equivalent iff:
for each elementary cycle c of $\mathcal{A} \otimes \mathcal{B}$,
$c_{|A}$ is accepting $\iff$ $c_{|B}$ is accepting.



$\mathcal{A}$ $\mathcal{B}$ $\mathcal{A} \otimes \mathcal{B}$
(acceptance marks omitted)

Equivalence Check of Two DTGBA

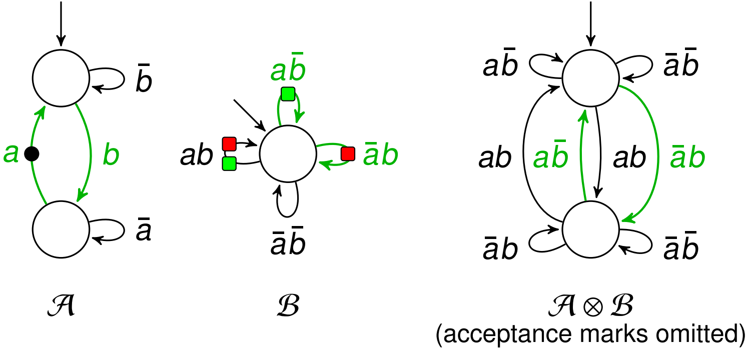
Two **complete** DTGBA $\mathcal{A}$ and $\mathcal{B}$ are equivalent iff:
for each elementary cycle c of $\mathcal{A} \otimes \mathcal{B}$,
$c_{|A}$ is accepting $\iff$ $c_{|B}$ is accepting.



$\mathcal{A}$ $\mathcal{B}$ $\mathcal{A} \otimes \mathcal{B}$
(acceptance marks omitted)

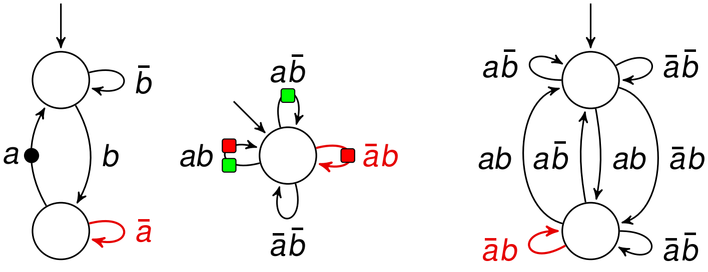
Equivalence Check of Two DTGBA

Two **complete** DTGBA $\mathcal{A}$ and $\mathcal{B}$ are equivalent iff:
for each elementary cycle c of $\mathcal{A} \otimes \mathcal{B}$,
$c_{|A}$ is accepting $\iff$ $c_{|B}$ is accepting.



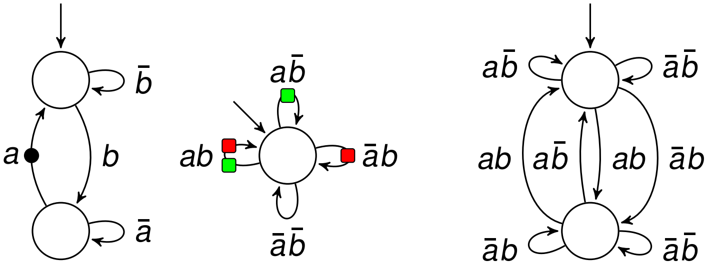
Equivalence Check of Two DTGBA

Two **complete** DTGBA $\mathcal{A}$ and $\mathcal{B}$ are equivalent iff:
for each elementary cycle c of $\mathcal{A} \otimes \mathcal{B}$,
$c_{|A}$ is accepting $\iff$ $c_{|B}$ is accepting.



$\mathcal{A}$ $\mathcal{B}$ $\mathcal{A} \otimes \mathcal{B}$
(acceptance marks omitted)

Two **complete** DTGBA $\mathcal{A}$ and $\mathcal{B}$ are equivalent iff:
for each elementary cycle c of $\mathcal{A} \otimes \mathcal{B}$,
$c_{|A}$ is accepting $\iff$ $c_{|B}$ is accepting.



$\mathcal{A}$ $\mathcal{B}$ $\mathcal{A} \otimes \mathcal{B}$
(acceptance marks omitted)

Now, given a reference $\mathcal{A}$, does a smaller equivalent $\mathcal{B}$ exist?

SAT-Based Synthesis of Equivalent DTGBA

We look for an automaton $\mathcal{B}$ equivalent to $\mathcal{A}$, but with $|\mathcal{A}| - 1$ states and m acceptance sets.

1. Encode as a SAT problem:
   - Some Boolean variables represent all possible transitions in $\mathcal{B}$.
   - More Boolean variables represent all possible cycles in the product $\mathcal{A} \otimes \mathcal{B}$.
   - Constraints ensure that transitions in the product are letter-compatible, and the *elementary cycle acceptance* condition is fulfilled.
2. Run a SAT solver:
   - If the problem is UNSAT, then a smaller DTGBA does not exist.
   - Otherwise the solution contains an encoding of $\mathcal{B}$.

SAT-Based Synthesis of Equivalent DTGBA

We look for an automaton $\mathcal{B}$ equivalent to $\mathcal{A}$, but with $|\mathcal{A}| - 1$ states and m acceptance sets.

1. Encode as a SAT problem:
   - Some Boolean variables represent all possible transitions in $\mathcal{B}$.
   - More Boolean variables represent all possible cycles in the product $\mathcal{A} \otimes \mathcal{B}$.
   - Constraints ensure that transitions in the product are letter-compatible, and the *elementary cycle acceptance* condition is fulfilled.

Differs from Ehlers' approach in the support for **generalized acceptance**, and some **SCC-based** encoding **optimizations**.

2. Run a SAT solver:
   - If the problem is UNSAT, then a smaller DTGBA does not exist.
   - Otherwise the solution contains an encoding of $\mathcal{B}$.

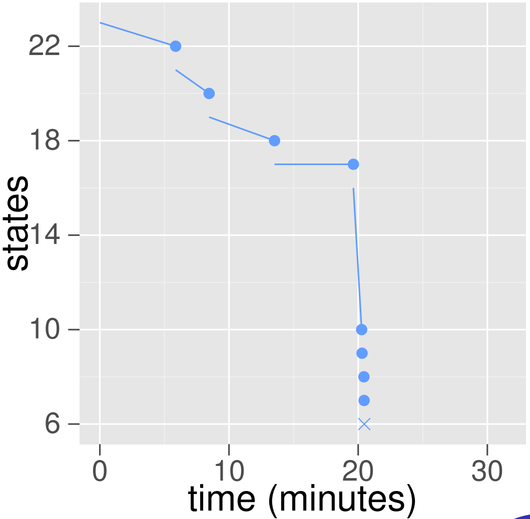
R. Ehlers. Minimising DBA precisely using SAT solving. SAT'10

Minimization by Iterative Synthesis

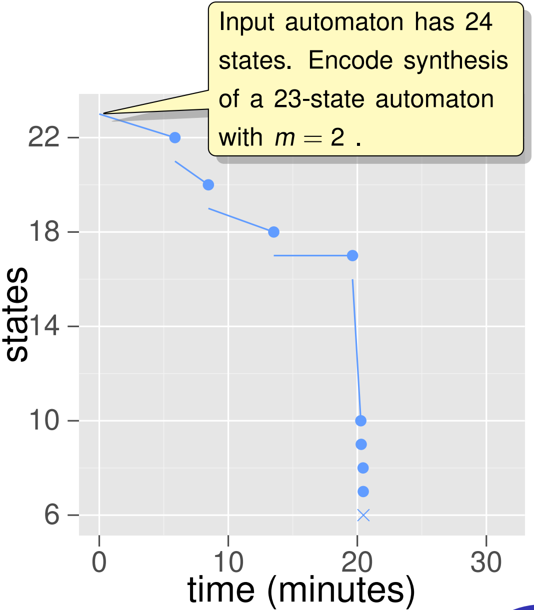
$\textsc{Minimize}(\mathcal{A}, m = \mathcal{A}.\text{nb\_acc}())$:
repeat:
$n \leftarrow \mathcal{A}.\text{nb\_states}()$
$\mathcal{B} \leftarrow \textsc{Synthesize}(\mathcal{A}, n{-}1, m)$
if $\mathcal{B}$ **does not exists**:
return $\mathcal{A}$
$\mathcal{A} \leftarrow \mathcal{B}$

Minimization by Iterative Synthesis

$\textsc{Minimize}(\mathcal{A}, m = \mathcal{A}.\text{nb\_acc}())$:
repeat:
  $n \leftarrow \mathcal{A}.\text{nb\_states}()$
  $\mathcal{B} \leftarrow \textsc{Synthesize}(\mathcal{A}, n{-}1, m)$
  if $\mathcal{B}$ **does not exists**:
    return $\mathcal{A}$
  $\mathcal{A} \leftarrow \mathcal{B}$



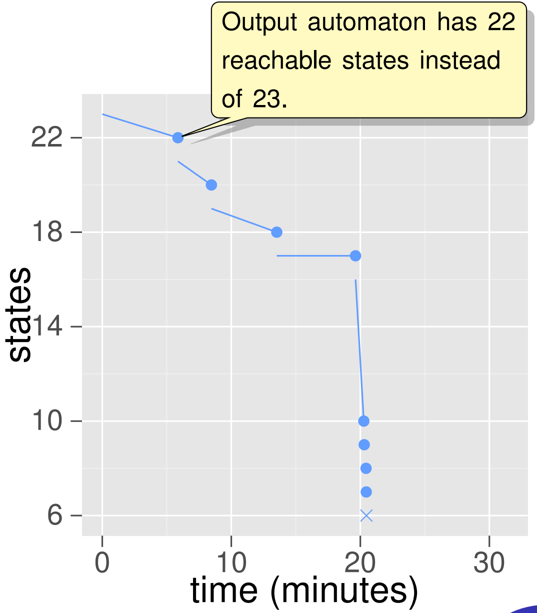
Minimization by Iterative Synthesis



$\textsc{Minimize}(\mathcal{A}, m = \mathcal{A}.\text{nb\_acc}())$:
  repeat:
    $n \leftarrow \mathcal{A}.\text{nb\_states}()$
    $\mathcal{B} \leftarrow \textsc{Synthesize}(\mathcal{A}, n{-}1, m)$
    if $\mathcal{B}$ **does not exists**:
      return $\mathcal{A}$
    $\mathcal{A} \leftarrow \mathcal{B}$

Minimization by Iterative Synthesis

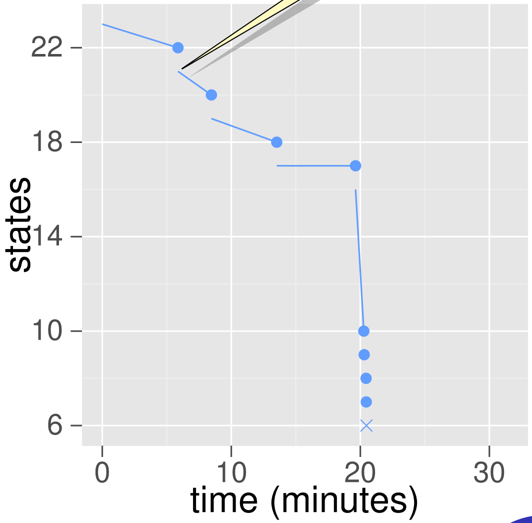
$\textsc{Minimize}(\mathcal{A}, m = \mathcal{A}.\mathrm{nb\_acc}())$:
repeat:
$n \leftarrow \mathcal{A}.\mathrm{nb\_states}()$
$\mathcal{B} \leftarrow \textsc{Synthesize}(\mathcal{A}, n{-}1, m)$
if $\mathcal{B}$ **does not exists**:
return $\mathcal{A}$
$\mathcal{A} \leftarrow \mathcal{B}$



Minimization by Iterative Synthesis

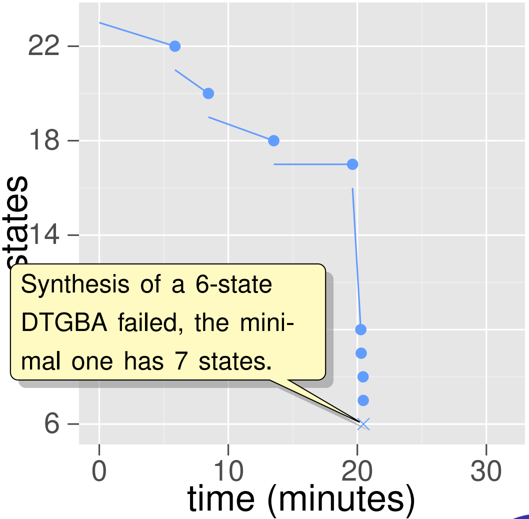
$\textsc{Minimize}(\mathcal{A}, m = \mathcal{A}.\text{nb\_acc}())$:
repeat:
  $n \leftarrow \mathcal{A}.\text{nb\_states}()$
  $\mathcal{B} \leftarrow \textsc{Synthesize}(\mathcal{A}, n{-}1, m)$
  if $\mathcal{B}$ **does not exists**:
    return $\mathcal{A}$
  $\mathcal{A} \leftarrow \mathcal{B}$

Encode synthesis of a 21-state automaton. Etc.



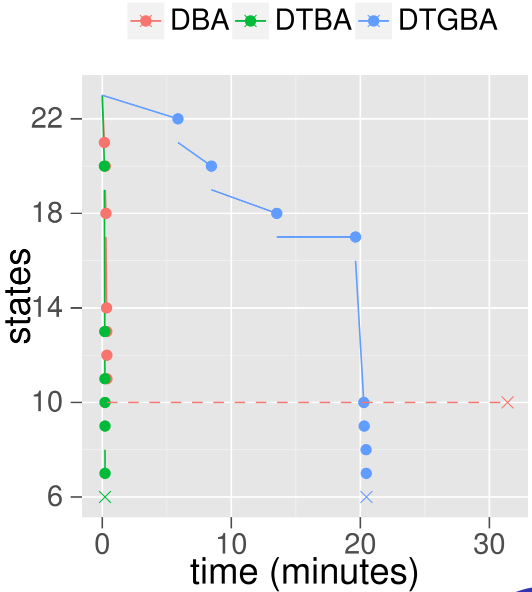
Minimization by Iterative Synthesis

$\textsc{Minimize}(\mathcal{A}, m = \mathcal{A}.\text{nb\_acc}())$:
repeat:
  $n \leftarrow \mathcal{A}.\text{nb\_states}()$
  $\mathcal{B} \leftarrow \textsc{Synthesize}(\mathcal{A}, n{-}1, m)$
  if $\mathcal{B}$ **does not exists**:
    return $\mathcal{A}$
  $\mathcal{A} \leftarrow \mathcal{B}$



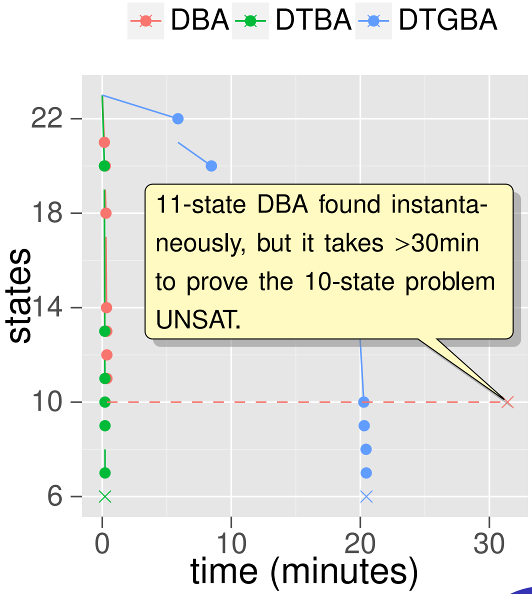
Minimization by Iterative Synthesis

MINIMIZE($\mathcal{A}$, $m = \mathcal{A}$.nb_acc()):
repeat:
$n \leftarrow \mathcal{A}$.nb_states()
$\mathcal{B} \leftarrow$ SYNTHESIZE($\mathcal{A}$, $n-1$, m)
if $\mathcal{B}$ **does not exists**:
return $\mathcal{A}$
$\mathcal{A} \leftarrow \mathcal{B}$



Minimization by Iterative Synthesis

$\textsc{Minimize}(\mathcal{A}, m = \mathcal{A}.\mathrm{nb\_acc}())$:
  repeat:
    $n \leftarrow \mathcal{A}.\mathrm{nb\_states}()$
    $\mathcal{B} \leftarrow \textsc{Synthesize}(\mathcal{A}, n{-}1, m)$
    if $\mathcal{B}$ **does not exists**:
      return $\mathcal{A}$
    $\mathcal{A} \leftarrow \mathcal{B}$



Contributions

- We extended Ehlers' approach with:
  - support generalized and transition-based acceptance,
  - SCC-based optimizations of the encoding (not discussed here)
- We integrated this minimization procedure in a more general framework supporting different determinization procedures, and a faster minimization procedure for weak automata.
  - Our tool is integrated in Spot 1.2.3, available from `http://spot.lip6.fr/`
  - Instructions for building minimal D[T][G]BA are at `http://spot.lip6.fr/userdoc/satmin.html`
- We ran a large benchmark exploring the effects of this minimization on many DTGBA generated from LTL formulas.

Future Work

- Comparing the minimal automata computed in our benchmark with automata produced by LTL→TGBA or LTL→BA translators suggests that these tools could be improved in many cases.
- We can create minimal DTGBA with m acceptance conditions, but it is not clear how to select the right m.
- We believe the technique can easily be extended to deal with Rabin or Streett acceptance.