Compositional Approach to Suspension and Other Improvements to LTL Translation

Tomáš Babiak[1] Thomas Badie[2] Alexandre Duret-Lutz[2]
Mojmír Křetínský[1] Jan Strejček[1]

[1] Faculty of Informatics, Masaryk University, Brno, Czech Republic

[2] LRDE, EPITA, Le Kremlin-Bicêtre, France

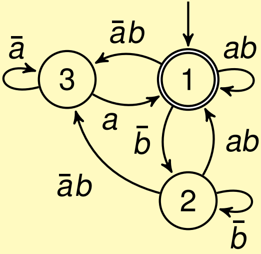
SPIN'13, 8–9 July 2013

From LTL to BA: The Big Picture

LTL form.
φ

$\mathbf{G}\mathbf{F}a \wedge \mathbf{G}\mathbf{F}b$

Büchi automaton



From LTL to BA: The Big Picture

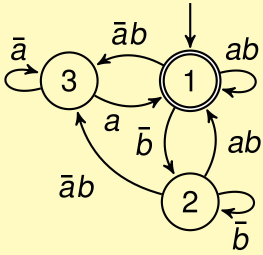
LTL form. φ

LTL rewritings

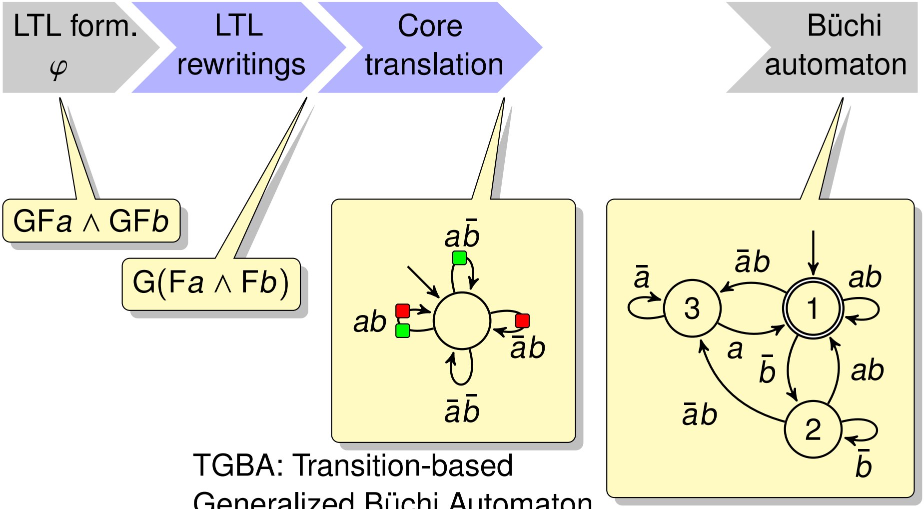
Büchi automaton

$\mathsf{GF}a \wedge \mathsf{GF}b$

$\mathsf{G}(\mathsf{F}a \wedge \mathsf{F}b)$



From LTL to BA: The Big Picture



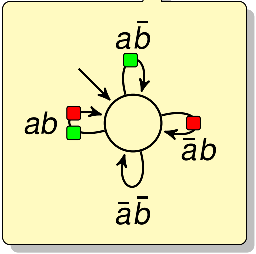
TGBA: Transition-based Generalized Büchi Automaton

From LTL to BA: The Big Picture

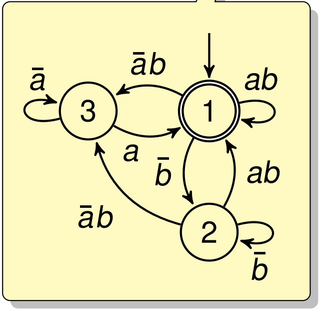
LTL form. φ → LTL rewritings → Core translation → Post-processings → Büchi automaton

$\mathsf{GF}a \wedge \mathsf{GF}b$

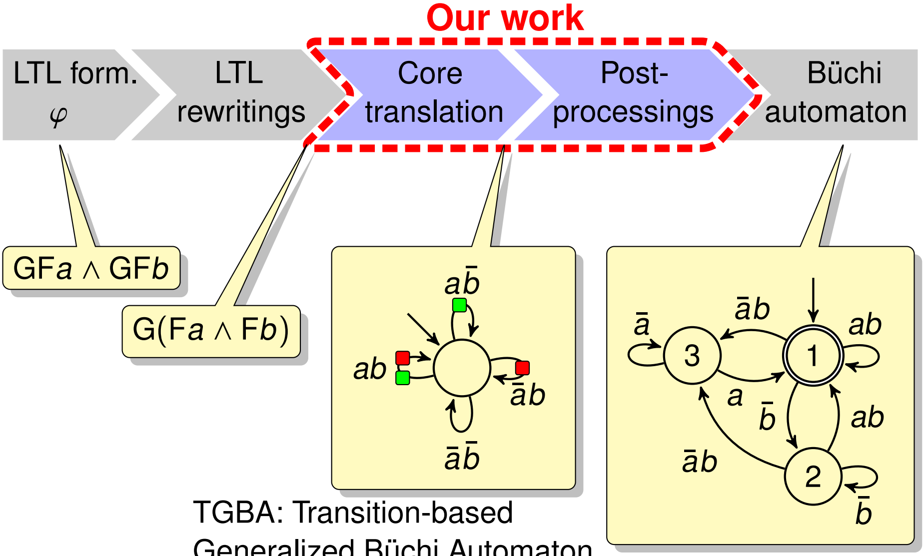
$\mathsf{G}(\mathsf{F}a \wedge \mathsf{F}b)$



TGBA: Transition-based
Generalized Büchi Automaton



From LTL to BA: The Big Picture



TGBA: Transition-based
Generalized Büchi Automaton

From LTL to BA: More Details

- Generic workflow:

φ | Trans. to TGBA | Simplify TGBA | Degen-eralize | Simplify BA | BA

- Dead SCCs removal
- Acceptance simplifications
- Simulation-based reductions

- Simulation-based reductions

From LTL to BA: More Details

- Generic workflow:

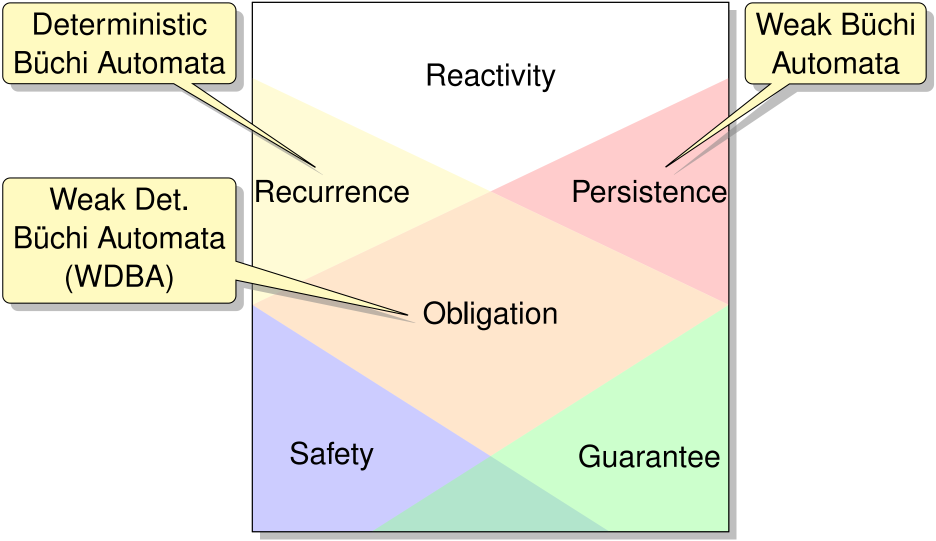
φ | Trans. to TGBA | Simplify TGBA | Degen- eralize | Simplify BA | BA

Simplify TGBA:
- Dead SCCs removal
- Acceptance simplifications
- Simulation-based reductions

Simplify BA:
- Simulation-based reductions

- Obligation properties can be translated better!

Temporal Hierarchy



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

From LTL to BA: More Details

- Generic workflow:

φ → Trans. to TGBA → Simplify TGBA → Degen-eralize → Simplify BA → BA

Simplify TGBA:
- Dead SCCs removal
- Acceptance simplifications
- Simulation-based reductions

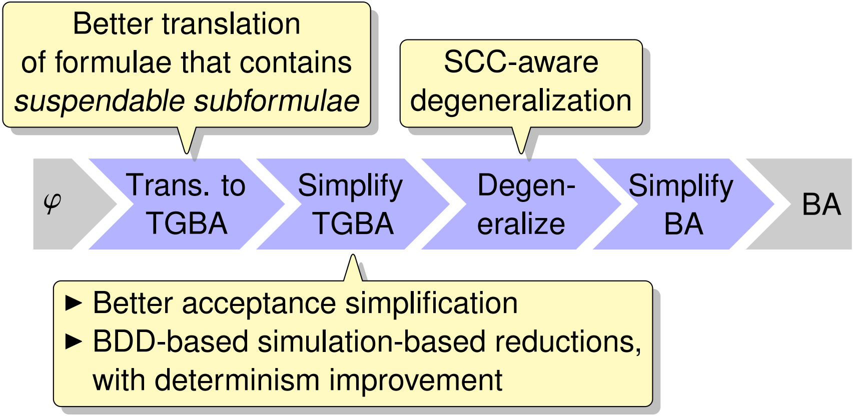
Simplify BA:
- Simulation-based reductions

- Obligation properties can be translated into **minimal** Weak Deterministic Büchi Automata:

φ → Trans. to TGBA → WDBA minimization → (WD)BA

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

Our Contributions



Our Contributions

This talk

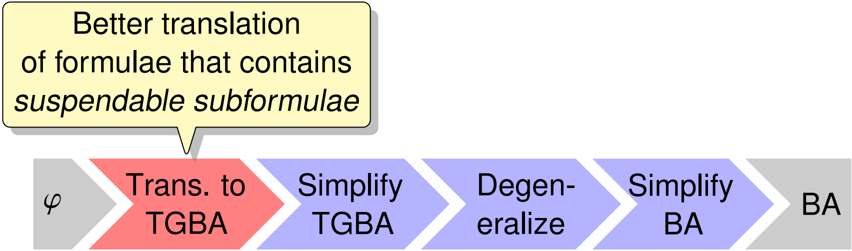
Better translation of formulae that contains *suspendable subformulae*

SCC-aware degeneralization

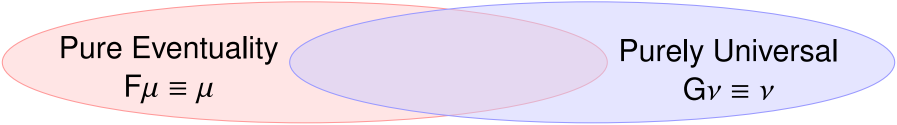
- Better acceptance simplification
- BDD-based simulation-based reductions, with determinism improvement

only in the paper

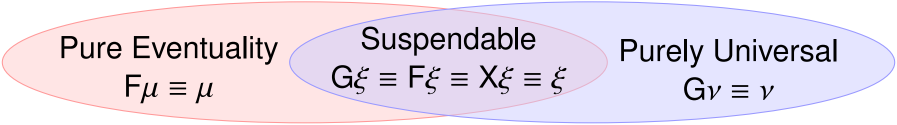
Compositional Suspension



Suspendable Formulae



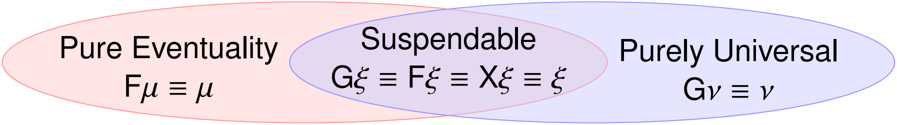
Suspendable Formulae



- **Intuition**: subspendable formulae have one F and one G in each syntactic branch. E.g., all usual fairness constraints:
  - $GF\varphi$
  - $FG\varphi \rightarrow GF\rho$
  - $GF\varphi \rightarrow GF\rho$

T. Babiak, M. Křetínský, V. Řehák, and J. Strejček. LTL to Büchi automata translation: Fast and more deterministic. *TACAS'12*

Suspendable Formulae

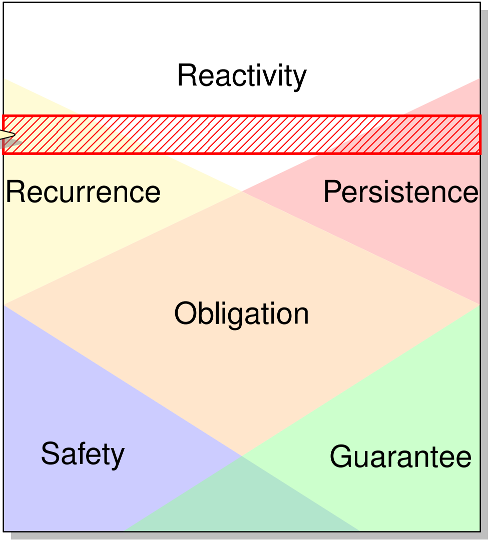


- **Intuition**: subspendable formulae have one F and one G in each syntactic branch. E.g., all usual fairness constraints:
  - $\mathsf{GF}\varphi$
  - $\mathsf{FG}\varphi \rightarrow \mathsf{GF}\rho$
  - $\mathsf{GF}\varphi \rightarrow \mathsf{GF}\rho$
- **Key property**: a suspendable formula either holds at all steps of an execution, or it holds at none.
- **Consequence**: its verification can be “suspended” by any finite number of steps.

T. Babiak, M. Křetínský, V. Řehák, and J. Strejček. LTL to Büchi automata translation: Fast and more deterministic. *TACAS’12*

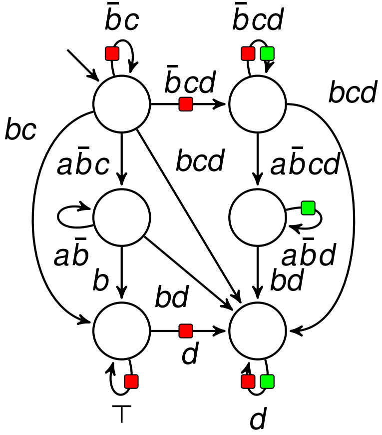
Temporal Hierarchy

Formulae with suspendable subformulae

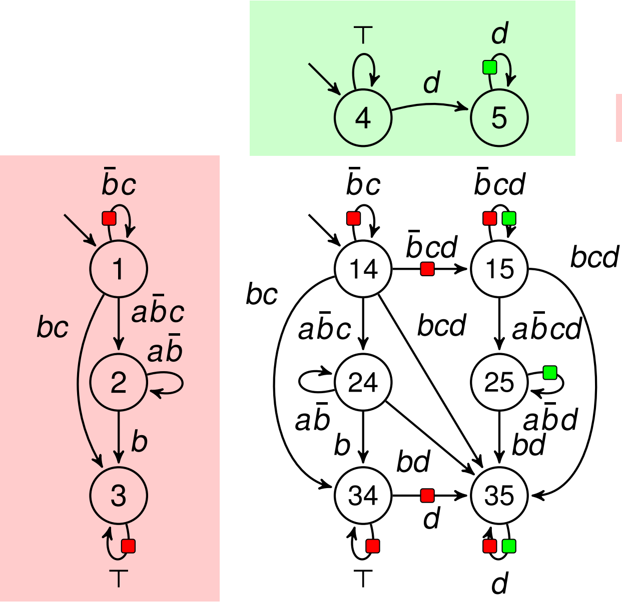


Using Suspension During Translation (Intuition)

$((a \,\mathsf{U}\, b)\,\mathsf{R}\, c) \;\wedge\; \mathsf{FG}d$

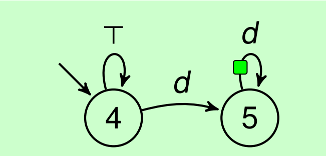


Using Suspension During Translation (Intuition)



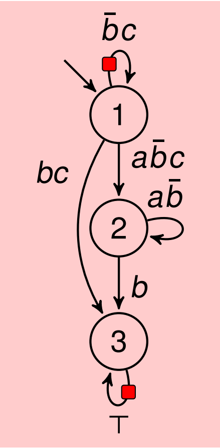
$((a \mathbin{\mathsf{U}} b) \mathbin{\mathsf{R}} c) \land \mathsf{FG}d$

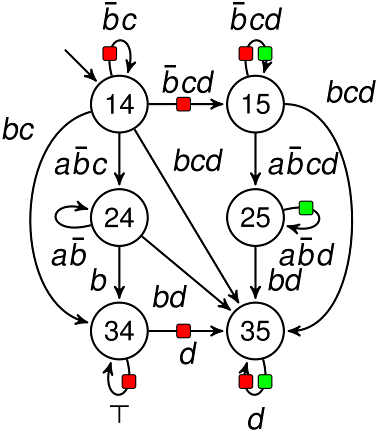
Using Suspension During Translation (Intuition)



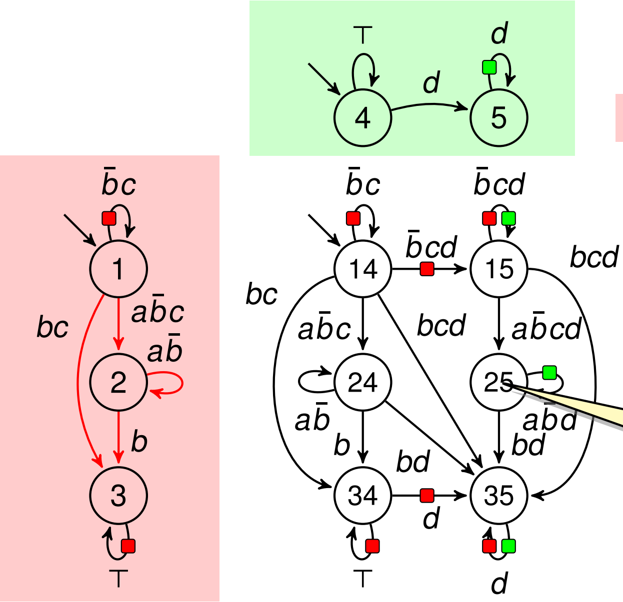
$((a \,\mathsf{U}\, b)\,\mathsf{R}\, c) \wedge \mathsf{FG}d$

Suspendable!





Using Suspension During Translation (Intuition)

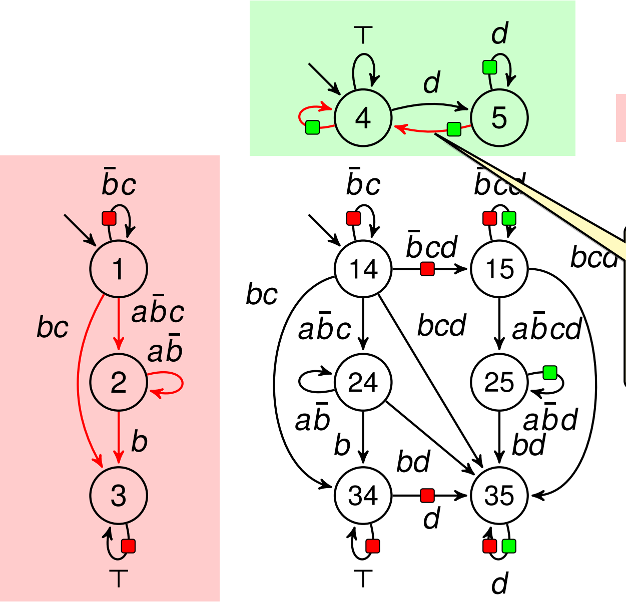


$((a \,\mathsf{U}\, b) \,\mathsf{R}\, c) \wedge \mathsf{FG}d$

Suspendable!

Pointless!
No need to check for $\mathsf{FG}d$ while $((a \,\mathsf{U}\, b) \,\mathsf{R}\, c)$ is not in an accepting SCC.

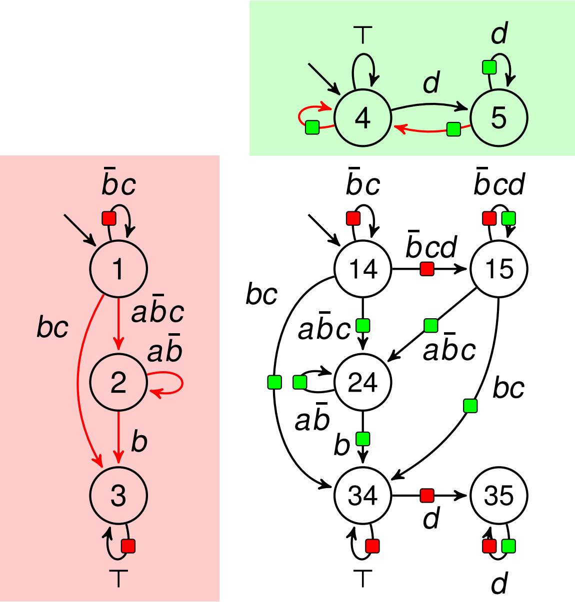
Using Suspension During Translation (Intuition)



$((a\,\mathsf{U}\,b)\,\mathsf{R}\,c) \wedge \mathsf{FG}d$

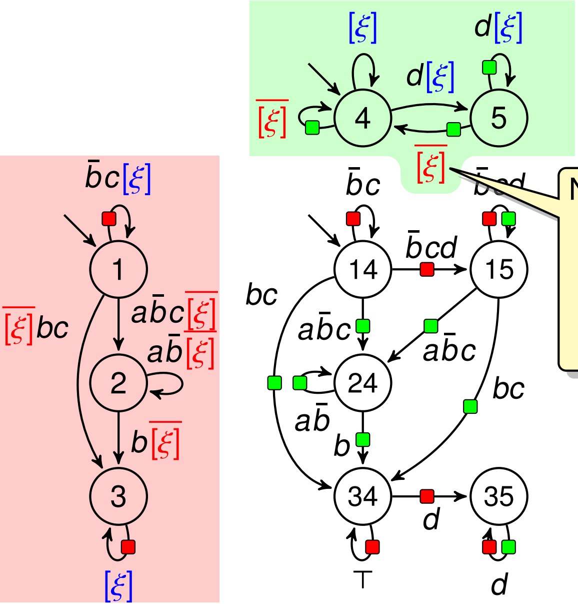
Reset transitions to be synchronized with transitions out of accepting SCCs.

Using Suspension During Translation (Intuition)



$((a \,\mathsf{U}\, b)\,\mathsf{R}\, c) \wedge \mathsf{FG}d$

Using Suspension During Translation (Intuition)



$((a \,\mathsf{U}\, b) \,\mathsf{R}\, c) \wedge \mathsf{FG}d$

New atomic proposition so that our special synchronization can be implemented as a synchronous product.

Our Compositional Approach to Suspension

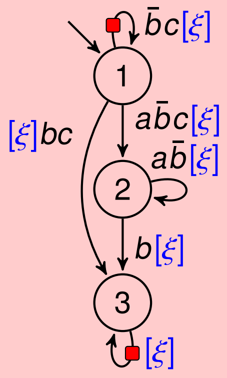
Given an LTL formula φ: $\boxed{((a \,\mathsf{U}\, b) \,\mathsf{R}\, c) \wedge \mathsf{FG}d}$

Our Compositional Approach to Suspension

Given an LTL formula φ: $\boxed{((a\,\mathsf{U}\,b)\,\mathsf{R}\,c) \wedge \mathsf{FG}d}$

1. Rewrite all (maximal) suspendable subformulae ξ_i of φ as $\mathsf{G}[\xi_i]$. Call this φ'.
   $\boxed{\varphi' = ((a\,\mathsf{U}\,b)\,\mathsf{R}\,c) \wedge \mathsf{G}[\xi]}$ $\boxed{\xi = \mathsf{FG}d}$

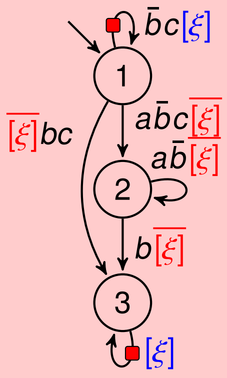
Our Compositional Approach to Suspension



Given an LTL formula φ: $\boxed{((a \,\mathsf{U}\, b) \,\mathsf{R}\, c) \wedge \mathsf{FG}d}$

1. Rewrite all (maximal) suspendable subformulae ξ_i of φ as $\mathsf{G}[\xi_i]$. Call this φ'.
   $\boxed{\varphi' = ((a \,\mathsf{U}\, b) \,\mathsf{R}\, c) \wedge \mathsf{G}[\xi]}$ $\boxed{\xi = \mathsf{FG}d}$
2. Translate φ' as a TGBA $A_{\varphi'}$

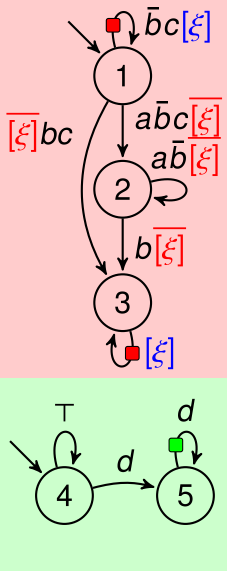
Our Compositional Approach to Suspension



Given an LTL formula φ: $\boxed{((a \,\mathsf{U}\, b) \,\mathsf{R}\, c) \wedge \mathsf{FG}d}$

1. Rewrite all (maximal) suspendable subformulae ξ_i of φ as $\mathsf{G}[\xi_i]$. Call this φ'.
   $\boxed{\varphi' = ((a \,\mathsf{U}\, b) \,\mathsf{R}\, c) \wedge \mathsf{G}[\xi]}$ $\boxed{\xi = \mathsf{FG}d}$
2. Translate φ' as a TGBA $A_{\varphi'}$
3. Remove $[\xi_i]$ from all transitions that are not in accepting SCCs.
4. Add $\overline{[\xi_i]}$ to transitions that do not have $[\xi_i]$.

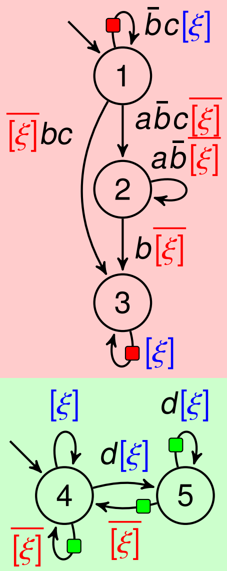
Our Compositional Approach to Suspension



Given an LTL formula φ: $\boxed{((a \,\mathsf{U}\, b) \,\mathsf{R}\, c) \land \mathsf{FG}d}$

1. Rewrite all (maximal) suspendable subformulae ξ_i of φ as $\mathsf{G}[\xi_i]$. Call this φ'.
   $\boxed{\varphi' = ((a \,\mathsf{U}\, b) \,\mathsf{R}\, c) \land \mathsf{G}[\xi]}$ $\boxed{\xi = \mathsf{FG}d}$
2. Translate φ' as a TGBA $A_{\varphi'}$
3. Remove $[\xi_i]$ from all transitions that are not in accepting SCCs.
4. Add $\overline{[\xi_i]}$ to transitions that do not have $[\xi_i]$.
5. Translate each ξ_i into A_{ξ_i}

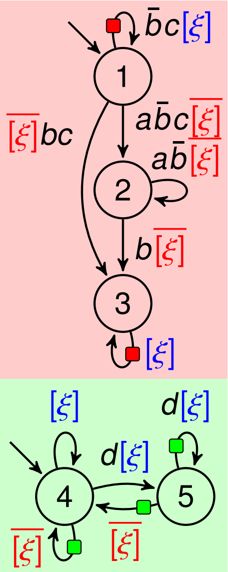
Our Compositional Approach to Suspension



Given an LTL formula φ: $\boxed{((a\,\mathsf{U}\,b)\,\mathsf{R}\,c) \wedge \mathsf{FG}d}$

1. Rewrite all (maximal) suspendable subformulae ξ_i of φ as $\mathsf{G}[\xi_i]$. Call this φ'.
   $\boxed{\varphi' = ((a\,\mathsf{U}\,b)\,\mathsf{R}\,c) \wedge \mathsf{G}[\xi]}$ $\boxed{\xi = \mathsf{FG}d}$
2. Translate φ' as a TGBA $A_{\varphi'}$
3. Remove $[\xi_i]$ from all transitions that are not in accepting SCCs.
4. Add $\overline{[\xi_i]}$ to transitions that do not have $[\xi_i]$.
5. Translate each ξ_i into A_{ξ_i}
6. Add $[\xi_i]$ labels and reset transitions to each A_{ξ_i}.

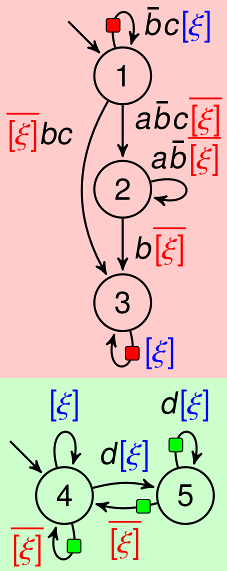
Our Compositional Approach to Suspension



Given an LTL formula φ: $\boxed{((a \,\mathsf{U}\, b) \,\mathsf{R}\, c) \wedge \mathsf{FG}d}$

1. Rewrite all (maximal) suspendable subformulae ξ_i of φ as $\mathsf{G}[\xi_i]$. Call this φ'.
   $\boxed{\varphi' = ((a \,\mathsf{U}\, b) \,\mathsf{R}\, c) \wedge \mathsf{G}[\xi]}$ $\boxed{\xi = \mathsf{FG}d}$
2. Translate φ' as a TGBA $A_{\varphi'}$
3. Remove $[\xi_i]$ from all transitions that are not in accepting SCCs.
4. Add $\overline{[\xi_i]}$ to transitions that do not have $[\xi_i]$.
5. Translate each ξ_i into A_{ξ_i}
6. Add $[\xi_i]$ labels and reset transitions to each A_{ξ_i}.
7. Build the product of all these automata. Strip $[\xi_i]$ and $\overline{[\xi_i]}$ from the result.

Our Compositional Approach to Suspension



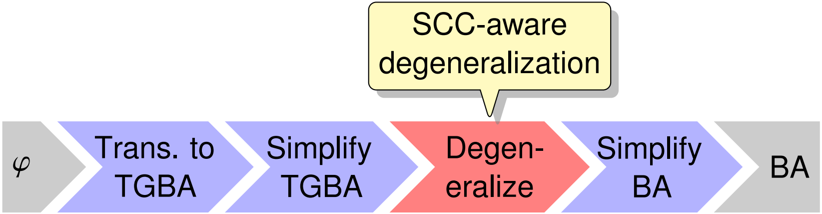
Given an LTL formula φ: $\boxed{((a \mathbin{\mathsf{U}} b) \mathbin{\mathsf{R}} c) \land \mathsf{FG}d}$

1. Rewrite all (maximal) suspendable subformulae ξ_i of φ as $\mathsf{G}[\xi_i]$. Call this φ'.
   $\boxed{\varphi' = ((a \mathbin{\mathsf{U}} b) \mathbin{\mathsf{R}} c) \land \mathsf{G}[\xi]}$ $\boxed{\xi = \mathsf{FG}d}$
2. Translate φ' as a TGBA $A_{\varphi'}$ and simplify it.
3. Remove $[\xi_i]$ from all transitions that are not in accepting SCCs.
4. Add $\overline{[\xi_i]}$ to transitions that do not have $[\xi_i]$.
5. Translate each ξ_i into A_{ξ_i} and simplify them.
6. Add $[\xi_i]$ labels and reset transitions to each A_{ξ_i}.
7. Build the product of all these automata. Strip $[\xi_i]$ and $\overline{[\xi_i]}$ from the result.

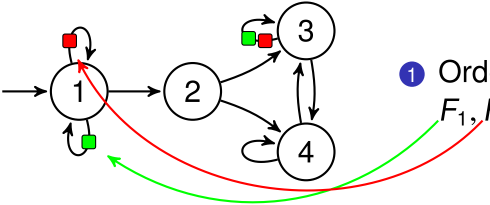
Compositional Suspension Benefits

- Can work on top of any translator.
- Largest reduction obtained when A_{ξ_i} are big, and $A_{\varphi'}$ have a lot of non-accepting SCCs.
- Suspendable formulae include usual fairness constraints.
- Intermediate automata can be simplified independently.
- In particular, φ' could be an obligation and $A_{\varphi'}$ subjected to WDBA-minimization.

SCC-Aware Degeneralization

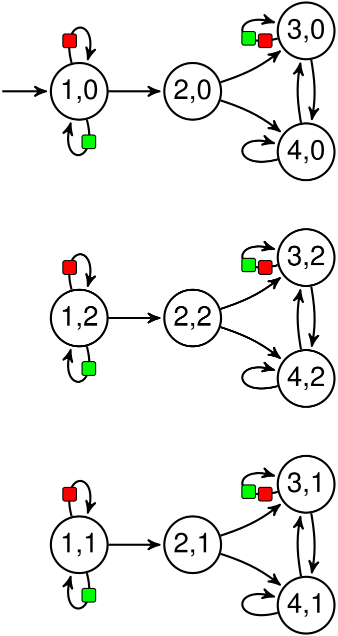


Classical Degeneralization (TGBA $\longrightarrow$ BA)



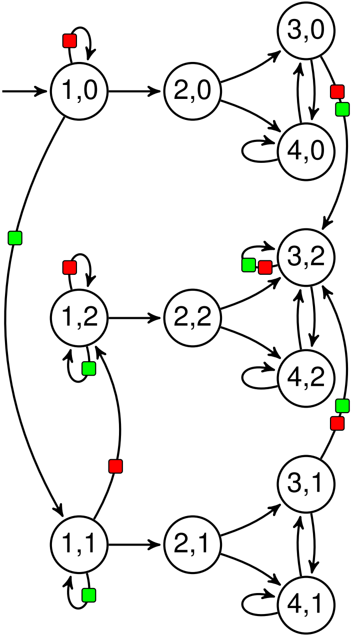
1. Order the m acceptance sets $F_1, F_2 \ldots, F_m$

Classical Degeneralization (TGBA → BA)



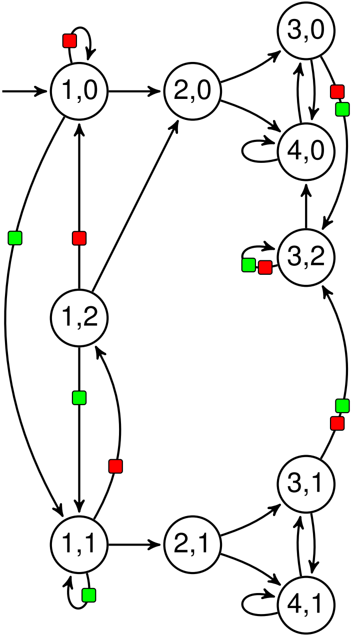
1. Order the m acceptance sets $F_1, F_2 \ldots, F_m$
2. Duplicate $m + 1$ times

Classical Degeneralization (TGBA $\rightarrow$ BA)



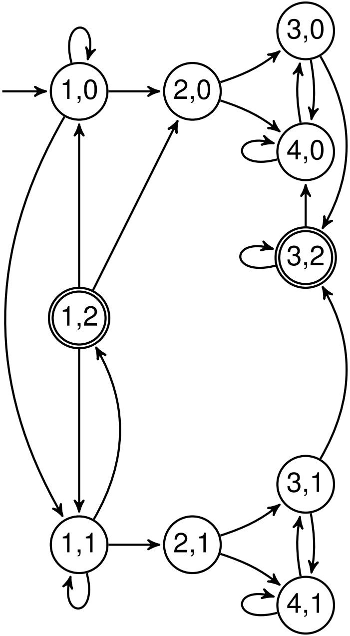
1. Order the m acceptance sets $F_1, F_2 \ldots, F_m$
2. Duplicate $m + 1$ times
3. Level $i < m$ redirects outputs from $F_{i+1} \cap F_{i+2} \cap \ldots \cap F_j$ to level j

Classical Degeneralization (TGBA $\rightarrow$ BA)



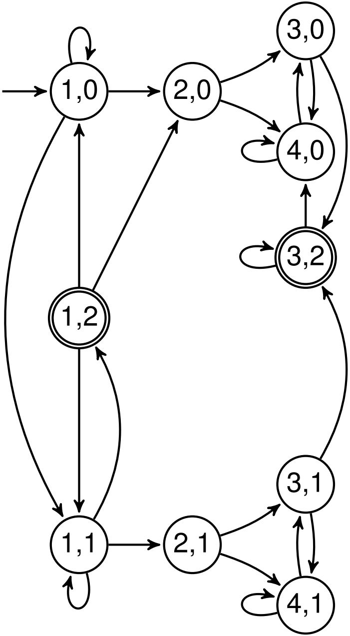
1. Order the m acceptance sets $F_1, F_2 \dots, F_m$
2. Duplicate $m + 1$ times
3. Level $i < m$ redirects outputs from $F_{i+1} \cap F_{i+2} \cap \dots \cap F_j$ to level j
4. Wire level m like level 0.

Classical Degeneralization (TGBA → BA)



1. Order the m acceptance sets $F_1, F_2 \ldots, F_m$
2. Duplicate $m + 1$ times
3. Level $i < m$ redirects outputs from $F_{i+1} \cap F_{i+2} \cap \ldots \cap F_j$ to level j
4. Wire level m like level 0.
5. Mark level m as accepting.

SCC-Aware Degeneralization (TGBA → BA)

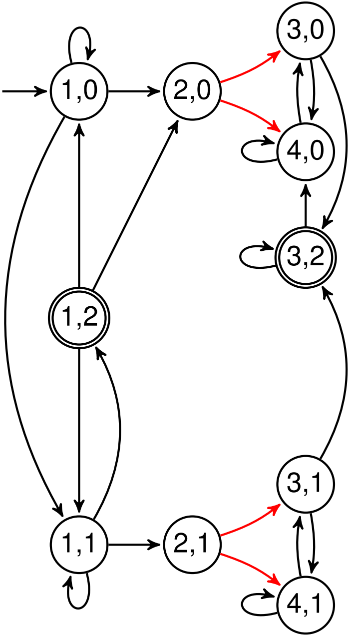


1. Order the m acceptance sets $F_1, F_2 \ldots, F_m$
2. Duplicate $m + 1$ times
3. Level $i < m$ redirects outputs from $F_{i+1} \cap F_{i+2} \cap \ldots \cap F_j$ to level j
4. Wire level m like level 0.
5. Mark level m as accepting.

We suggest two optimizations:

- **Level Caching**:
- **Level Reset**:

SCC-Aware Degeneralization (TGBA → BA)

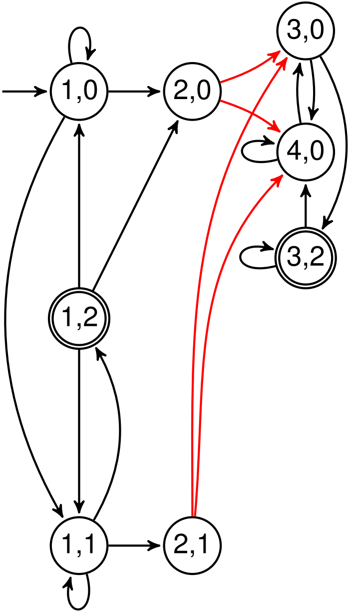


1. Order the m acceptance sets $F_1, F_2 \ldots, F_m$
2. Duplicate $m + 1$ times
3. Level $i < m$ redirects outputs from $F_{i+1} \cap F_{i+2} \cap \ldots \cap F_j$ to level j
4. Wire level m like level 0.
5. Mark level m as accepting.

We suggest two optimizations:

- **Level Caching**: Upon *entering* an accepting SCC (of the TGBA), reuse any existing level.
- **Level Reset**:

SCC-Aware Degeneralization (TGBA → BA)

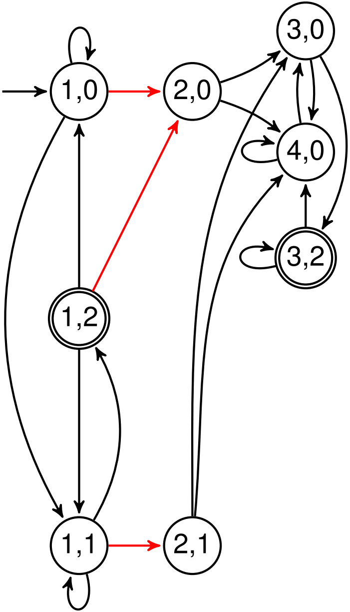


1. Order the m acceptance sets $F_1, F_2 \ldots, F_m$
2. Duplicate $m + 1$ times
3. Level $i < m$ redirects outputs from $F_{i+1} \cap F_{i+2} \cap \ldots \cap F_j$ to level j
4. Wire level m like level 0.
5. Mark level m as accepting.

We suggest two optimizations:

- **Level Caching**: Upon *entering* an accepting SCC (of the TGBA), reuse any existing level.
- **Level Reset**:

SCC-Aware Degeneralization (TGBA $\rightarrow$ BA)

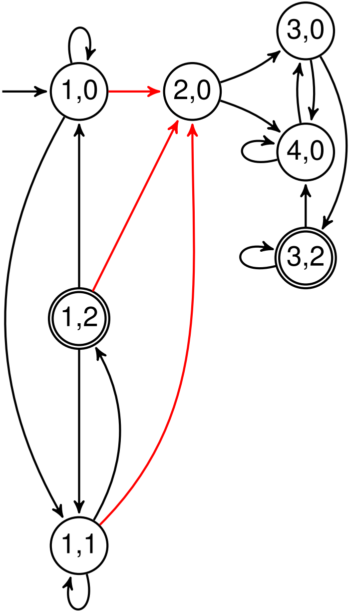


1. Order the m acceptance sets $F_1, F_2 \ldots, F_m$
2. Duplicate $m + 1$ times
3. Level $i < m$ redirects outputs from $F_{i+1} \cap F_{i+2} \cap \ldots \cap F_j$ to level j
4. Wire level m like level 0.
5. Mark level m as accepting.

We suggest two optimizations:

- **Level Caching**: Upon *entering* an accepting SCC (of the TGBA), reuse any existing level.
- **Level Reset**: Upon *leaving* an SCC, reset the level to 0.

SCC-Aware Degeneralization (TGBA → BA)



1. Order the m acceptance sets $F_1, F_2 \ldots, F_m$
2. Duplicate $m + 1$ times
3. Level $i < m$ redirects outputs from $F_{i+1} \cap F_{i+2} \cap \ldots \cap F_j$ to level j
4. Wire level m like level 0.
5. Mark level m as accepting.

We suggest two optimizations:

- **Level Caching**: Upon *entering* an accepting SCC (of the TGBA), reuse any existing level.
- **Level Reset**: Upon *leaving* an SCC, reset the level to 0.

Some Results

100 random formulae of the form $\varphi_i \wedge (\mathsf{GF}a \rightarrow \mathsf{GF}b) \wedge (\mathsf{GF}c \rightarrow \mathsf{GF}d)$

φ → Trans. to TGBA → Simplify TGBA → Degen-eralize → BA

- Dead SCCs removal
- Acceptance simplifications

	states	transitions	time
baseline	8207	3928868	114 s

Some Results

100 random formulae of the form $\varphi_i \wedge (\mathsf{GF}a \rightarrow \mathsf{GF}b) \wedge (\mathsf{GF}c \rightarrow \mathsf{GF}d)$

- Dead SCCs removal
- Acceptance simplifications

	states	transitions	time
baseline	8207	3928868	114 s
+ better acceptance simplification	8083	3876308	151 s

Some Results

100 random formulae of the form $\varphi_i \land (\mathsf{GF}a \rightarrow \mathsf{GF}b) \land (\mathsf{GF}c \rightarrow \mathsf{GF}d)$

φ → Trans. to TGBA → Simplify TGBA → Degen-eralize → BA

- Dead SCCs removal
- Acceptance simplifications
- Simulation-based reductions

	states	transitions	time
baseline	8207	3928868	114 s
+ better acceptance simplification	8083	3876308	151 s
+ simulation	3488	782324	178 s

Some Results

100 random formulae of the form $\varphi_i \wedge (\mathsf{GF}a \rightarrow \mathsf{GF}b) \wedge (\mathsf{GF}c \rightarrow \mathsf{GF}d)$

φ → Trans. to TGBA → Simplify TGBA → Degen-eralize → Simplify BA → BA

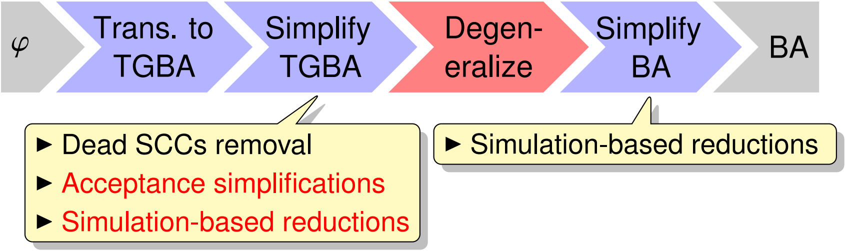
- Dead SCCs removal
- Acceptance simplifications
- Simulation-based reductions

- Simulation-based reductions

	states	transitions	time
baseline	8207	3928868	114 s
+ better acceptance simplification	8083	3876308	151 s
+ simulation	3488	782324	178 s
+ BA simulation	3371	699096	183 s

Some Results

100 random formulae of the form $\varphi_i \wedge (\mathsf{GF}a \rightarrow \mathsf{GF}b) \wedge (\mathsf{GF}c \rightarrow \mathsf{GF}d)$



	states	transitions	time
baseline	8207	3928868	114 s
+ better acceptance simplification	8083	3876308	151 s
+ simulation	3488	782324	178 s
+ BA simulation	3371	699096	183 s
+ better degeneralization	3259	727416	181 s

Some Results

100 random formulae of the form $\varphi_i \land (\mathsf{GF}a \rightarrow \mathsf{GF}b) \land (\mathsf{GF}c \rightarrow \mathsf{GF}d)$

φ | Trans. to TGBA | Simplify TGBA | Degen-eralize | Simplify BA | BA

- Dead SCCs removal
- Acceptance simplifications
- Simulation-based reductions

- Simulation-based reductions

	states	transitions	time
baseline	8207	3928868	114 s
+ better acceptance simplification	8083	3876308	151 s
+ simulation	3488	782324	178 s
+ BA simulation	3371	699096	183 s
+ better degeneralization	3259	727416	181 s
+ compositional suspension	3091	668768	53 s

Conclusion

Better translation of formulae that contains *suspendable subformulae*

SCC-aware degeneralization

- Better acceptance simplification
- BDD-based simulation-based reductions, with determinism improvement

- LTL-to-BA translators are already fairly well optimized. We still managed some improvement.
- All these techniques are implemented in Spot 1.1.2.
- Compositional suspension can be tested on-line at `http://spot.lip6.fr/ltl2tgba.html`