A Fresh Look at Linear Temporal Logic

Javier Esparza
Technical University of Munich

Joint work with Jan Křetínský and Salomon Sickert
If all bandersnatches are borogoves and all borogoves are slithy, then all bandersnatches are slithy.
If all bandersnatches are borogoves and all borogoves are slithy, then all bandersnatches are slithy. "True"
Logic

If all $X$ are $Y$ and all $Y$ are $Z$, then all $X$ are $Z$. 
Logic is the subject of identifying true statements in a language of which you only know a few words.
Temporal Logic

Different logics study different language fragments:

- Propositional logic: and, or, not, if ... then

- Temporal logic: propositional logic + today, tomorrow, eventually, never, ...

Studied within mathematical logic since the end of the XIX century.

Clarence Lewis (1883-1964)

Arthur Prior (1914-1969)
Temporal logic in computer science

• Amir Pnueli proposes in 1977 to use temporal logic to reason about computer programs
A worker that succeeds in acquiring a lock will **eventually** release it, assuming its "doResult" call returns.

The `req_close_state` is **always** in `close_enabled` state.

If `artist1` registers for event2 **before** `artist2` does, then **once** dispatcher receives event2 from the ADT, it will **first** send it to `artist1` **and then** to `artist2`.

The OK button on the login window is enabled **as soon as** the application is started and the login window is **first** displayed to the user.

None of the available methods can be called **until** `connect` is called.

Mathew Dwyer, Temporal Specification Patterns, https://matthewbdwyer.github.io/psp/
Linear Temporal Logic (LTL)

- LTL extends propositional logic with temporal operators.
- Syntax:

\[ \varphi ::= \text{true} | \text{false} | p | \neg p | \varphi_1 \land \varphi_2 | \varphi_1 \lor \varphi_2 \\
X\varphi_1 | \varphi_1 U \varphi_2 | \varphi_1 W \varphi_2 \\
\varphi_1 R \varphi_2 | \varphi_1 M \varphi_2 | \text{past operators (Past LTL)} \\
F\varphi ::= \text{true} U \varphi \quad \text{(eventually } \varphi \text{ or finally } \varphi) \\
G\varphi ::= \varphi W \text{false} \quad \text{(always } \varphi \text{ or globally } \varphi) \]
Temporal logic in computer science

A worker that succeeds in acquiring a lock will eventually release it, assuming its "doResult" call returns.

\[ G(\text{call} \_\text{doResult} \rightarrow F \text{retur}n_{\text{doResult}}) \rightarrow G(\text{return}_{\text{lockacq}} \rightarrow F \text{call}_{\text{lockrel}}) \]

If artist1 registers for event before artist2 does, then once dispatcher receives event from the ADT, it will first notify artist1 and then artist2.

\[ G((\text{reg} \_a_1 \land (\neg \text{unreg} \_a_1 U (\text{reg} \_a_2 \land (\neg \text{unreg} \_a_1 \land \neg \text{unreg} \_a_2) U \text{notify})))) \rightarrow F (\text{notify} \land (\neg \text{notify} \_a_2 U \text{notify} \_a_1))) \]
Specifying and verifying reactive systems

Zohar Manna (1939-2018)

Amir Pnueli (1941-2009)

1992

1995
Proof rules for different classes in the hierarchy:

I1. $\Theta \rightarrow \varphi$
I2. $\varphi \rightarrow q$
I3. $\{\varphi\} \vdash T \{\varphi\}$
   $\Box q$

C1. $\Box (p \rightarrow (q \lor \varphi))$
C2. $\{\varphi\} \vdash T \{q \lor \varphi\}$
C3. $\{\varphi\} \tau \{q\}$
C4. $T - \{\tau\} \vdash \Box (\varphi \rightarrow \Diamond (q \lor En(\tau)))$
   $\Box (p \rightarrow \Diamond q)$

B1. $\Box (p \rightarrow (q \lor \varphi))$
B2. $\{\varphi \land (\delta = \alpha)\} \vdash T \{q \lor (\varphi \land (\delta \leq \alpha))\}$
B3. $\Box (\varphi \land (\delta = \alpha) \land r) \rightarrow \Diamond [q \lor (\delta < \alpha)]$
   $\Box ((p \land \Box r) \rightarrow \Diamond q)$

The Safety-Progress Hierarchy

$\phi_i$ and $\psi_i$ are past formulas
The Safety-Progress Hierarchy

Normal form theorem
Every formula is equivalent to a reactivity formula.

\( \phi_i \) and \( \psi_i \) are past formulas
Proving the Normal Form Theorem

The proof [...] is based on many previous results, including [Buc], [MNP], [C], [T] and [GPSS] which, when combined, yield the theorems almost immediately.

Lichtenstein, Pnueli, Zuck: Logic of Programs, 1985
Proving the Normal Form Theorem

Past LTL formula

Proving the Normal Form Theorem


Chapter 4
Proving the Normal Form Theorem

- Past LTL formula
  - Counter-free semi-automaton
    - Star-free regular expression

Krohn-Rhodes Decomposition Theorem

Chapter 4

Proving the Normal Form Theorem

Past LTL formula

Counter-free semi-automaton

Star-free regular expression

Past LTL formula in normal form


Chapter 4

Krohn-Rhodes Decomposition Theorem

Chapter 5
Proving the Normal Form Theorem

Past LTL formula

Counter-free semi-automaton

Star-free expression

Non-elementary blow-up!!!

in normal form

Chapter 4

Krohn-Rhodes Decomposition Theorem

Chapter 5

Proving the Normal Form Theorem

Past LTL formula

Counter-free semi-automaton

Star-free decomposition

Non-elementary blow-up!!!

in normal form

Chapter 4

Krohn-Rhodes Decomposition

Chapter 5

Maybe "only" triple exponential?

Maler, Essays in Memory of Amir Pnueli, 2010

... and the rest is silence.

No further attempts to improve on these bounds, even though there is no lower bound!

How come?
... and the rest is silence.

No further attempts to improve on these bounds, even though there is no lower bound!

An Automata-Theoretic Approach to Automatic Program Verification

Moshe Y. Vardi
CSLI, Ventura Hall, Stanford University, Stanford, CA 94305.

Pierre Wolper
AT&T Bell Laboratories
600 Mountain Ave.
Murray Hill, NJ 07974

LICS '86

Gödel Prize
2000
Automata-theoretic approach

- Translates the formula into an $\omega$-automaton (automaton on infinite words) and "throws the formula away"
- Proofs replaced by automata-theoretic algorithms
- No need for hierarchies, proof rules, or axiom systems

Duret-Lutz: Spot Online Translator
https://spot.lrde.epita.fr/app/
Automata-theoretic approach

- Translates the formula into an $\omega$-automaton (automaton on infinite words) and "throws the formula away"
- Proofs replaced by automata-theoretic algorithms
- No need for hierarchies, proof rules, or axiom systems
- LTL "demoted" to syntax for automata

Duret-Lutz: Spot Online Translator
https://spot.lrde.epita.fr/app/
Automata-theoretic approach

During the next decades the automata-theoretic approach
Automata-theoretic approach

During the next decades the automata-theoretic approach

- is implemented in sophisticated, very successful tools
Automata-theoretic approach

During the next decades the automata-theoretic approach

• is implemented in sophisticated, very successful tools
• is extended to the verification of probabilistic systems
Automata-theoretic approach

During the next decades the automata-theoretic approach
- is implemented in sophisticated, very successful tools
- is extended to the verification of probabilistic systems
- is applied to reactive synthesis: automatic synthesis of reactive systems from LTL specifications
The challenge

• Reactive synthesis requires to translate LTL into deterministic ω-automata

• Probabilistic verification requires to translate LTL into limit deterministic (or deterministic) ω-automata
The theoretical challenge

On The Complexity of $\omega$-Automata

Shmuel Safra

Department of Applied Mathematics
Weizmann Institute of Science
Rehovot 76100, Israel

FOCS 1988: Determinization procedure for $\omega$-automata

The Complexity of Probabilistic Verification

COSTAS COURCOUBETIS
University of Crete, and ICS, FORTH, Heraklion, Greece
AND
MIHALIS YANNAKAKIS
AT&T Bell Laboratories, Murray Hill, New Jersey

JACM 1995: Limit-determinization procedure for $\omega$-automata

(see also Vardi 1985)
The algorithmic challenge

These translations

- have **double-exponential** blow-up.
  (contrary to single-exponential for LTL → nondet. automata)
- are „monolithic“ and very combinatorial
  e.g. states of Safra‘s det. automaton are
  **trees of sets of states** of the nondet. automaton
  \[\implies\]
  Implementations struggle to control combinatorial explosion
The algorithmic challenge

How can I do better in the future?
How can I do better in the future?

Do better in the past!
Back to the 1980s: The Safety-Progress Hierarchy

Normal form theorem
Every formula is equivalent to a reactivity formula.

\[
\bigwedge_i (\text{FG } \phi_i \land \text{GF } \psi_i)
\]

\[
\bigwedge_i (\text{F } \phi_i \land \text{G } \psi_i)
\]

\(\phi_i\) and \(\psi_i\) are past formulas
The Alternation Hierarchy

- **Δ₂**: boolean combination of Σ₂ and Π₂
- **Σ₂**: at most one alternation from U to W, e.g. \( \phi U (\psi U (\chi W \rho)) \)
- **Π₂**: at most one alternation from W to U, e.g. \( \phi W (\psi U (\chi U \rho)) \)
- **Δ₁**: boolean combination of Σ₁ and Π₁
- **Σ₁**: only U, e.g. \( \phi U (\psi U \chi) \)
- **Π₁**: only W, e.g. \( \phi W (\psi W \chi) \)
- **Σ₀**: only U, e.g. \( \phi U \)
- **Π₀**: only W, e.g. \( \phi W \)
The Alternation Hierarchy

Pelánek and Strejček, CIAA 2005
The Alternation Hierarchy

Normal form theorem

Every formula is equivalent to a $\Delta_2$-formula.

Non-elementary blow-up!!!
Demystifying normalization

\[ F(\alpha \land G (b \lor Fc)) \]
Demystifying normalization

\[ F(a \land G(b \lor Fc)) \]
Demystifying normalization

\[ F(a \land G(b \lor Fc)) \]

Case 1: \( Fc \) holds infinitely often \((GFc\) holds\)

Case 2: \( Fc \) only holds finitely often \((\neg GFc\) holds\)
Demystifying normalization

\[ \text{Case 1: } Fc \text{ holds infinitely often (GF} \neg c \text{ holds)} \]

Then \[ G (b \lor Fc) \equiv^{GFc} true \]

\[ \text{Case 2: } Fc \text{ only holds finitely often (} \neg GFc \text{ holds)} \]
Demystifying normalization

\[ F(\alpha \land G (b \lor Fc)) \]

Case 1: \(Fc\) holds infinitely often \((GFc\) holds) 
Then \( G (b \lor F c) \equiv^{GFc} true \)

Case 2: \(Fc\) only holds finitely often \((\neg GFc\) holds) 
Then \( G (b \lor F c) \equiv^{\neg GFc} (b \lor F c) U (G b) \)
Demystifying normalization

F(a ∧ G(b ∨ Fc))

Case 1: Fc holds infinitely often (GFc holds)
Then G(b ∨ Fc) ≡^{GFc} true

Case 2: Fc only holds finitely often (¬GFc holds)
Then G(b ∨ Fc) ≡¬^{GFc} (b ∨ Fc) U (G b)

WU → UW !!
Demystifying normalization

\[ F(a \land G(b \lor Fc)) \]
Demystifying normalization

\[ F(a \land G (b \lor Fc)) \equiv GFc \land F(a \land G (b \lor Fc)) \lor \neg GFc \land F(a \land G (b \lor Fc)) \]
Demystifying normalization

\[ F(a \land G (b \lor FC)) \equiv GFc \land F(a \land G (b \lor FC)) \lor \neg GFc \land F(a \land G (b \lor FC)) \]
Demystifying normalization

\[ F(a \land G(b \lor Fc)) \equiv GFc \land F(a \land \text{true}) \]
\[ \lor \]
\[ \neg GFc \land F(a \land G(b \lor Fc)) \]
Demystifying normalization

\[ F(a \land G(b \lor Fc)) \equiv GFc \land F(a \land \text{true}) \]
\[ \lor \]
\[ \neg GFc \land F(a \land G(b \lor Fc)) \]

Correct because \( GFc \)
holds at some moment
iff it holds at every moment!
Demystifying normalization

\[ F(a \land G (b \lor Fc)) \equiv GFc \land F(a \land \text{true}) \]
\[ \lor \]
\[ \neg GFc \land F(a \land (b \lor Fc) \lor Gb) \]

Correct because \( \neg GFc \) holds at some moment iff it holds at every moment!
... et voilá!

\[ F(a \land G(b \lor Fc)) \equiv GFc \land Fa \]
\[ \lor \]
\[ F(a \land (b \lor Fc)U Gb) \]
Demystifying normalization

\[ F \left( a \land G \left( b \lor F(c \land Gd) \right) \right) \]
Demystifying normalization

\[ F\left( a \land G\ (b \lor F(c \land Gd)) \right) \]
Demystifying normalization

\[
\begin{align*}
\mathbf{F} \left( a \land G \left( b \lor \mathbf{F} (c \land Gd) \right) \right) \\
\equiv \mathbf{FGd} \land \mathbf{F} \left( a \land G \left( b \lor \mathbf{F} (c \land Gd) \right) \right) \\
\lor \\
\neg \mathbf{FGd} \land \mathbf{F} \left( a \land G \left( b \lor \mathbf{F} (c \land Gd) \right) \right)
\end{align*}
\]
Demystifying normalization

\[
\begin{align*}
  &\ F\left(a \land G\left(b \lor F(c \land Gd)\right)\right) \\
  \equiv &\ FGd \land F\left(a \land G\left(b \lor (Fc\ W\left(c \land Gd\right))\right)\right) \\
    &\lor \\
  &\ \neg FGd \land F\left(a \land G\left(b \lor false\right)\right)
\end{align*}
\]
Demystifying normalization

\[ F \left( a \land G \left( b \lor F(c \land Gd) \right) \right) \]

\[ \equiv \quad FGd \land F \left( a \land G \left( b \lor (Fc \lor W(c \land Gd)) \right) \right) \]

\[ \lor \]

\[ GF \neg d \land F(a \land Gb) \]
Demystifying normalization

\[ F \left( a \land G \left( b \lor F(c \land Gd) \right) \right) \]
\[ \equiv FGd \land F \left( a \land G \left( b \lor (F \neg c \ W (c \land Gd)) \right) \right) \]
\[ \lor \]
\[ GF\neg d \land F(a \land Gb) \]
Demystifying normalization

\[ F \left( a \land G \left( b \lor F(c \land Gd) \right) \right) \]

\[ \equiv \quad FGd \land F \left( a \land G \left( b \lor \left( Fc \land W(c \land Gd) \right) \right) \right) \]

\[ \lor \]

\[ GF\neg d \land F(a \land Gb) \]
Demystifying normalization

\[
F \left( a \land G \left( b \lor F \left( c \land Gd \right) \right) \right) \\
\equiv F Gd \land \left( GFc \land F \left( a \land G \left( b \lor \left( Fc \lor W \left( c \land Gd \right) \right) \right) \right) \lor \left( \neg GFc \land F \left( a \land G \left( b \lor \left( Fc \land W \left( c \land Gd \right) \right) \right) \right) \lor GF \neg d \land F \left( a \land Gb \right) 
\]
Demystifying normalization

\[
F\left(a \land G\left(b \lor F(c \land Gd)\right)\right)
\]

\[
\equiv FGd \land \left(\begin{array}{c}
GFc \land F(a \land \text{true}) \\
\lor \\
FG\neg c \land F\left(a \land F(c \land Gd) \lor G(b \lor (c \land Gd))\right)
\end{array}\right) \\
\lor \\
GF\neg d \land F(a \land Gb)
\]

... et voilá!

\[
F \left( a \land G \left( b \lor F(c \land Gd) \right) \right)
\]

\[
\equiv \quad FGd \land GFc \land Fa \\
\lor \\
FGd \land F \left( a \land F(c \land Gd) \lor G (b \lor (c \land Gd)) \right) \\
\lor \\
F(a \land Gb)
\]
Closed-form expression

\[ \varphi \equiv \bigvee_{M \subseteq U(\varphi)} \left( \varphi' \land \bigwedge_{\psi \in M} \text{GF} \, \psi' \land \bigwedge_{\chi \in N} \text{FG} \, \chi' \right) \]
\[
\varphi \equiv \bigvee_{M \subseteq U(\varphi)} \left( \varphi' \land \bigwedge_{\psi \in M} GF \psi' \land \bigwedge_{\chi \in N} FG \chi' \right)
\]

\(O(2^n)\) disjuncts

\(O(n)\) conjuncts of length \(O(n)\)

Formula of length \(O(2^n)\)

Sickert and Esparza, LICS 2020
Esparza, Křetínský, and Sickert, JACM 2020
Closed-form expression

\[ \varphi \equiv 2^{O(n)} \text{ length} \]

\[ O(2^n) \text{ disjuncts} \]

\[ O(n) \text{ conjuncts of length } O(n) \]

Formula of length \( O(2^n) \)

Sickert and Esparza, LICS 2020
Esparza, Křetínský, and Sickert, JACM 2020
Back from the past
LTL ⇒ Limit-deterministic Büchi automata

The Complexity of Probabilistic Verification

COSTAS COURCUBETIS
University of Crete, and ICS, FORTH, Heraklion, Greece
AND
MIHALIS YANNAKAKIS
AT&T Bell Laboratories, Murray Hill, New Jersey

1995: Limit-determinization procedure for \( \omega \)-automata
LTL ⇒ Limit-deterministic Büchi automata

1995: Limit-determinization procedure for \( \omega \)-automata

Formula
\[ \implies \Delta_2 \text{-formula} \]
\[ \implies \text{Limit-deterministic Büchi automaton} \]
LTL $\Rightarrow$ Limit-deterministic Büchi automata
LTL $\Rightarrow$ Limit-deterministic Büchi automata

$$F(a \land G(b \lor Fc))$$
LTL $\Rightarrow$ Limit-deterministic Büchi automata

$$F(a \land G (b \lor Fc)) \equiv (GFc \land Fa) \lor F(a \land (b \lor Fc) U Gb)$$
LTL $\Rightarrow$ Limit-deterministic Büchi automata

\[ F(a \land G (b \lor Fc)) \equiv (GFc \land Fa) \lor F(a \land (b \lor Fc) U Gb) \]
\[ F(a \land G(b \lor Fc)) \equiv (GFc \land Fa) \lor F(a \land (b \lor Fc) U Gb) \]

Maintains the formula \( \psi_i \) that must hold when \( Gb \) starts to hold

Guesses the point at which \( Gb \) starts to hold

Checks \( \psi_i \land Gb \)
Size reduction

\[ \bigwedge_{i=1}^{j}(GFa_i) \implies \bigwedge_{i=1}^{j}(GFb_i) \]

\[ k: \bigwedge_{i=1}^{k}(GFa_i \lor FGb_i) \]

\[ f(0, j) = (GFa_0)U(X^j b) \]

\[ f(i+1, j) = (GFa_{i+1})U(Gf(i, j)) \]

<table>
<thead>
<tr>
<th></th>
<th>LDBA</th>
<th>Safra (spot+ltl2dstar)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>3</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>17</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>49</td>
</tr>
<tr>
<td></td>
<td>6</td>
<td>129</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>4385</td>
</tr>
<tr>
<td></td>
<td>9</td>
<td>*</td>
</tr>
<tr>
<td>f(0, 0)</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>f(0, 2)</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>f(0, 4)</td>
<td>16</td>
<td>12</td>
</tr>
<tr>
<td>f(1, 0)</td>
<td>6</td>
<td>196</td>
</tr>
<tr>
<td>f(1, 2)</td>
<td>28</td>
<td>109839</td>
</tr>
<tr>
<td>f(1, 4)</td>
<td>58</td>
<td>*</td>
</tr>
<tr>
<td>f(2, 0)</td>
<td>10</td>
<td>99793</td>
</tr>
<tr>
<td>f(2, 2)</td>
<td>46</td>
<td>*</td>
</tr>
<tr>
<td>f(2, 4)</td>
<td>92</td>
<td>*</td>
</tr>
</tbody>
</table>

Sickert, Esparza, Jaax, and Kretinsky, CAV 2016
LTL $\Rightarrow$ deterministic Rabin automata

On The Complexity of $\omega$-Automata*

Shmuel Safra

Department of Applied Mathematics
Weizmann Institute of Science
Rehovot 76100, Israel

1988: Determinization procedure for $\omega$-automata
LTL \Rightarrow \text{deterministic Rabin automata}

On The Complexity of $\omega$-Automata*

Shmuel Safra

Department of Applied Mathematics
Weizmann Institute of Science
Rehovot 76100, Israel

1988: Determinization procedure for $\omega$-automata

Formula

$\Rightarrow$ $\Delta_2$-formula

$\Rightarrow$ very weak $\Delta_2$-alternating Büchi automaton

$\Rightarrow$ deterministic Rabin automaton
From LTL to very weak alternating Büchi automata

\[ F(a \land XG(b \lor XF(c \land XGd))) \]

\[ G(b \lor XF(c \land XGd)) \]

\[ F(c \land XGd) \]

true

true
After $\Delta_2$-normalization

Disjunction of VWAA with $O(n)$ states s.t. each path has only one alternation between accepting and non-accepting states.
After $\Delta_2$-normalization

Lemma: A $\Delta_2$-VWAA accepts a word iff it has a run on it such that
- No level of the tree is (equiv. to) false, and
- All states of some level are accepting.

Equivalent deterministic Büchi or co-Büchi automaton using (a slight reformulation of) the breakpoint construction.

From trees of sets to pairs of sets.
Owl

A Java tool collection and library for Omega-words, ω-automata and Linear Temporal Logic (LTL). Batteries included.

Owl is a Java 11 tool collection and library for ω-words, ω-automata and linear temporal logic. It provides a wide range of algorithms for automata and LTL. It has

Křetínský, Meggendorfer, Sickert, ATVA 2018
Strix (strix.model.in.tum.de)

Tool for reactive LTL synthesis
- direct translation LTL-to-DPA
- multi-threaded, explicit-state solver for parity games.

Winner of the SYNTCOMP competition in 2018, 2019, 2020

State-of-the-art in reactive LTL synthesis

Luttenberger, Meyer, Sickert, CAV 2018 and Acta Informatica 2020
A fresh look at LTL
A fresh look at LTL

Imagine SAT without CNF
A fresh look at LTL

Imagine SAT without CNF

Imagine FOL without skolemization
A fresh look at LTL

Imagine SAT without CNF

Imagine FOL without skolemization

That’s what happened to LTL
Thank you for your attention!