First-Order vs. Second-Order Encodings for LTL_f-to-Automata

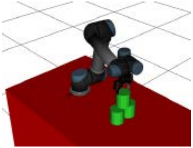
Shufang Zhu [1]   Geguang Pu [1]   Moshe Y. Vardi [2]

[1]East China Normal University

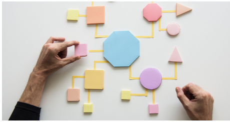
[2]Rice University

Introduction

- LTL_f: Linear Temporal Logic (LTL) over finite traces [De Giacomo & Vardi, IJCAI'13]

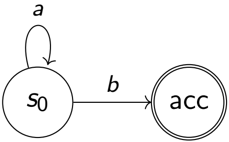


Planning in robotics



Business process modeling

- LTL_f to Deterministic Finite Automata (DFA): a critical step in many applications of LTL_f



Linear Temporal Logic over Finite Traces (LTL_f)

LTL_f formulas

- a set $\mathcal{P}$ of propositional symbols
- closed under
  - boolean connectives, Negation($\neg$), And($\wedge$), Or($\vee$)
  - temporal operators, Next(X), Until(U), Eventually(F), Release(R), Always(G)

Example

$\rho \models X\varphi$ next time step exists, and φ holds at the next time step

Note: LTL_f formula ϕ, DFA D such that ρ is accepted by D iff $\rho \models \phi$

LTL$_f$-to-Automata

- No direct tool for LTL$_f$-to-DFA
- MONA [Henriksen et al., 1995]:

  First-order or Second-order logic over finite traces $\rightarrow$ minimized DFAs

- First-order: quantifications over positions of the trace

$$\exists i, j. i < j$$

- Second-order: quantifications over relations of the positions of the trace

$$\exists R. R(i, j)$$

First-Order vs. Second-Order

- First-order (FOL) encoding: LTL_f formula to FOL [De Giacomo & Vardi, IJCAI'13]

- Second-order encoding?

Question:

Whether second-order (MSO) encoding can outperform first-order (FOL) encoding for LTL_f-to-automata translation?

Our Contributions

- Second-order encodings for LTL_f, simpler quantificational structure
  - MSO encoding: **Direct** translation from LTL_f to MSO
    - Semantics-driven translation
  - Compact MSO encoding: **Indirect** translation from LTL_f to MSO
    - Benefit from automata-theoretic minimization
  - Variations in terms of different optimizations
- First evaluation of the spectrum of encodings for LTL_f-to-Automata from first-order to second-order

MSO Encoding

Key Idea: Semantics-driven translation

- For each subformula θ of ϕ
  - Predicate Q_θ
  - $\mathrm{t}(\theta, i)$ (defined recursively) restricts Q_θ to follow the semantics of θ, such that $Q_\theta(i) \leftrightarrow \theta$ holds at i

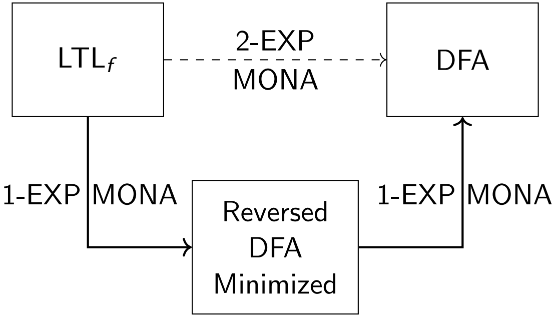
Example: $\theta = X\psi$, ρ satisfies θ at i iff $i \neq \mathit{last}$ and ρ satisfies ψ at $i+1$

- Predicates: Q_θ, Q_ψ
- $\mathrm{t}(\theta, i) = (\forall i)(Q_\theta(i) \leftrightarrow ((i \neq \mathit{last}) \wedge Q_\psi(i+1)))$

Formulation: $\mathrm{mso}(\phi) = (\exists Q_{\theta_1}) \cdots (\exists Q_{\theta_m})(Q_\phi(0) \wedge (\forall i)(\bigwedge_{k=1}^{m} \mathrm{t}(\theta_k, i))$

Compact MSO Encoding

Key Idea: Benefit from automata-theoretic minimization



Crux: encoding based on the symbolic representation of the **reversed** DFA as Binary Decision Diagrams (BDDs)

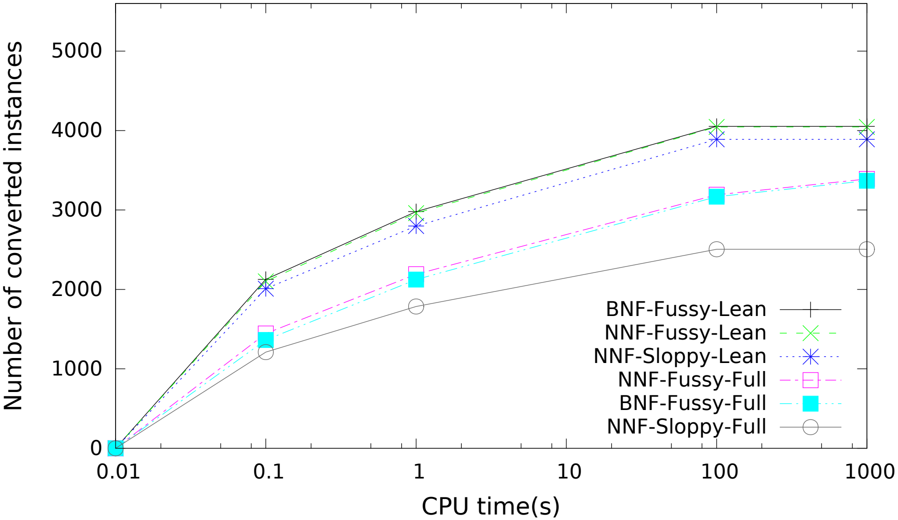
Optimizations of Second-Order Encodings

1. In the normal form: Boolean Normal Form (**BNF**) and Negation Normal Form (**NNF**) [Rozier & Vardi, 2011]
2. In the predicate form: **Lean** MSO encoding introduces fewer predicates than the **Full** MSO encoding [Pan, Sattler & Vardi, 2002]
3. In the constraint form: the **Sloppy** MSO encoding allows less tight constraint than the **Fussy** MSO encoding [Pan, Sattler & Vardi, 2003]

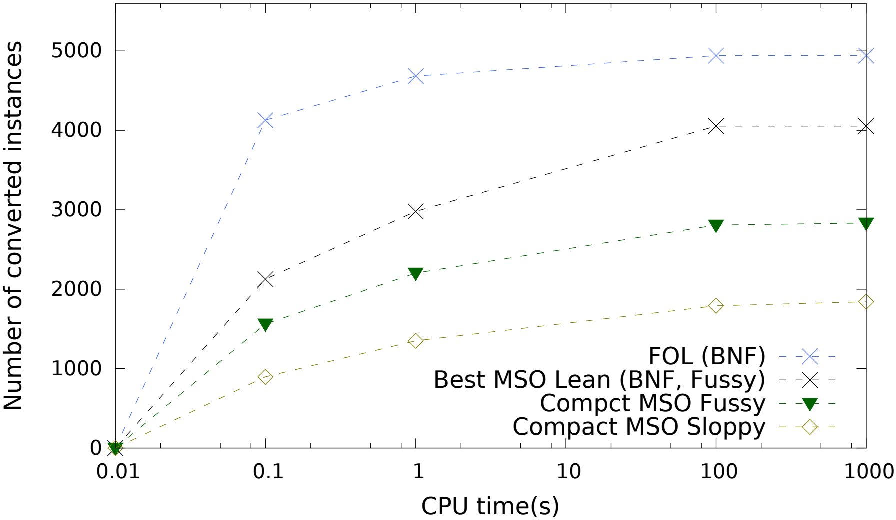
Experimental Evaluation

- First-order encoding
- 6 MSO encodings ($2^3 - 2 = 6$)
  - the Normal Form (BNF or NNF)
  - the Predicate Form (Full or Lean)
  - the Constraint Form (Fussy or Sloppy): Sloppy cannot be applied to BNF
- 2 Compact MSO encodings
  - the Constraint Form (Fussy or Sloppy)
- MONA: logic specifications to DFA

Less predicates (Lean) is more effective for MSO encodings



First-order encoding dominates Second-order encodings



Why First-order better?

$$\text{mso}(\phi) = (\exists Q_{\theta_1}) \cdots (\exists Q_{\theta_m})(Q_\phi(0) \wedge (\forall i)(\bigwedge_{k=1}^{m} \text{t}(\theta_k, i))$$

- MONA: process quantifiers over predicates one by one

Future Work

Better quantifier elimination strategy: a whole block of similar quantifiers in one operation

Conclusions

- Second-order encoding for LTL_f formulas with different optimizations
- First-order dominates Second-order for LTL_f-to-automata translation
- Potential for further improvement in second-order encoding for LTL_f