GOAL: A Graphical Tool for Manipulating Büchi Automata and Temporal Formulae

Yih-Kuen Tsay and Yu-Fang Chen
Dept. of Information Management
National Taiwan University

Joint work with Ming-Hsien Tsai, Kang-Nien Wu, and Wen-Chin Chan

Outline

- Background and Motivations
- Overview of the GOAL Tool
- Quick Review of
  - Büchi Automata
  - Propositional (Linear) Temporal Logic (PTL)
  - Quantified Propositional Temporal Logic (QPTL)
- Examples: Formulae vs. Automata
- Demo
- Conclusion

Note: will use LTL to refer to either (1) the LTL in model checking or (2) the general LTL in Manna and Pnueli's books; should be clear from the context.

Background

- Basics of the automata-theoretic approach to model checking [Vardi and Wolper, LICS 1986]:
  - The (finite-state) system is modeled as a Büchi automaton A.
  - A desired property is given by an LTL formula f.
  - Let B_f ($B_{\sim f}$) denote a Büchi automaton equivalent to f ($\sim f$).
  - The model checking problem translates into

$$L(A) \subseteq L(B_f) \text{ or } L(A) \cap L(B_{\sim f}) = \emptyset \text{ or } L(A \times B_{\sim f}) = \emptyset.$$

  - So, with LTL to automata translation, the difficult complementation problem is avoided (and hence neglected by most).
- The well-used model checker SPIN, for example, adopts the automata-theoretic approach.

Background (cont.)

- So, **Büchi automata** and **linear temporal logic** were linked together and became two fundamental components of model checking.
- People researching or teaching model checking ought to know these two formalisms.
- However, they had not been sufficiently well-equipped.

Motivations

- How familiar are you with Büchi automata?
  - Try drawing one, e.g., for [](p --> (p *U* q)).
  - How long did it take?
  - How many states does the automaton contain?
  - Are you sure it is correct?
- Do we have something like those nice **graphical** tools for learning *classic* automata and formal languages, e.g., JFLAP?
  - Graphical interfaces are convenient for doing exercises and preparing class material.
- How do you grade the 10 different Büchi automata that your students turn in for a LTL formula?

Motivations (cont.)

- Is your algorithm for LTL to Büchi automata translation correct?
  - The generated automaton looks different from one by another well-used algorithm.
  - How can you be sure the two are equivalent?
- Wouldn’t it be nice if you can quickly draw a Büchi automaton and submit it to SPIN, instead of writing Promela code?
  - The hand-drawn automaton may be smaller than the machine-translated one.

Summary of Motivations

- Learning/Teaching
  - To help understand the correspondence between Büchi automata and temporal formulae.
  - A graphical interactive tool should be helpful for both the *student* and the *teacher*.
- Research
  - To help test correctness of a translation algorithm
- Model Checker Enhancement
  - Graphical front end to model checkers
  - Convenience of *past* temporal operators
  - “Manual-optimization” of a specification automaton

Main Features of GOAL (Currently)

- Drawing and Testing Büchi Automata
- Tests on Büchi Automata
  - Input, Emptiness, Containment, and **Equivalence**
- Boolean Operations on Büchi Automata
  - Union, Intersection, and **Complementation**
- QPTL Formulae to Büchi Automata Translation
- Repository of Pre-Drawn and ***Checked*** Büchi Automata: 60 cases now and still counting
- Exporting Automata as Promela Code for SPIN

Implementation Highlights

- Implemented in Java for easy installation
- Textual and graphical interfaces adapted from those of JFLAP
- QPTL to Büchi: adaptation and extension of the tableau construction in Manna and Pnueli's book
  - Handle all future and past temporal operators
  - Can see which sub-formula is true in which state
- Büchi complementation: Safra's construction
  - It's most famous (or infamous)
  - People will want to compare theirs with it.

Büchi Automata

- ***Büchi automata*** (BAs) are a variant of so-called ω-automata, which are finite automata (FAs) operating on *infinite* words $w \in \Sigma^\omega$.
- BAs describe **non-terminating** behaviors, while classic FAs describe **terminating** behaviors.
- For BAs, non-determinism adds expressive power.
- We will consider only non-deterministic BAs.

Büchi Automata (cont.)

- BAs recognize ω-***regular*** languages expressible as $\cup\alpha_i\beta_i^{\omega}$, where α_i and β_i are regular expressions.
- BAs are expressively equivalent to QPTL when the infinite words are seen as models for QPTL formulae.
- BAs are closed under the boolean operations.

Büchi Automata (cont.)

- A BA is given, as in finite automata, by a 5-tuple $(\Sigma, Q, \delta, Q_0, F)$, where $F \subseteq Q$ is the set of accepting states.
- An infinite word $w \in \Sigma^\omega$ is accepted by a BA *B* if there exists a run *r* of *B* on *w* satisfying the condition:

$$\mathrm{Inf}(r) \cap F \neq \emptyset$$

where Inf(*r*) denotes the set of states occurring **infinitely many times** in *r*.

Generalized Büchi Automata

- A generalized Büchi automaton (GBA) is like a BA but with $F \subseteq 2^Q$, i.e., $F = \{F_1, \ldots, F_k\}$ where $F_i \subseteq Q$.

- A word $w \in \Sigma^\omega$ is accepted by a generalized Büchi automaton $\mathcal{B}$ if there exists a run r of $\mathcal{B}$ on w satisfying the condition:

$$\forall F_i \in F\text{:}\ \ \mathrm{Inf}(r) \cap F_i \neq \emptyset$$

About the Alphabet

- To link Büchi automata to temporal formulae, we will consider automata with an alphabet like:
  - {p, ~p},

    consisting of two symbols: “p” and “~p”, or
  - {p q, p ~q, ~p q, ~p ~q},

    consisting of four symbols: “p q”, “p ~q”, “~p q”, and “~p ~q”.

PTL and QPTL

- Both are variants of linear temporal logic (LTL).
- PTL and QPTL formulae are interpreted over an infinite sequence of states, which can be seen as an infinite word over alphabet like **{p, ~p}** or **{p q, p ~q, ~p q, ~p ~q}**.
- Every PTL formula is equivalent to some Büchi automaton, but not vice versa.
- QPTL extends PTL with quantification over propositions.
- QPTL is as expressive as Büchi automata.

Temporal Operators

- Future temporal operators:
  - next: $\bigcirc$ or X
  - eventually (sometime): $\Diamond$ or F
  - hence-forth (always): $\Box$ or G
  - until: $\mathcal{U}$
  - wait-for (unless): $\mathcal{W}$
- Past temporal operators (some in textual format):
  - previous: $\ominus$ or Y
  - before: (~) or Z
  - once: <-> or O
  - so-far: $\boxminus$ or H
  - since: $\mathcal{S}$
  - back-to: $\mathcal{B}$

Semantics of Future Operators

Let π be an infinite sequence of states.

- $(\pi, i) \models (\,)f$ iff $(\pi, i+1) \models f$
- $(\pi, i) \models \Diamond f$ iff $(\pi, j) \models f$ for some $j \geq i$
- $(\pi, i) \models [\,]f$ iff $(\pi, j) \models f$ for all $j \geq i$
- $(\pi, i) \models f\,U\,g$ iff for some $k \geq i$, $(\pi, k) \models g$ and for all j, $i \leq j < k$, $(\pi, j) \models f$
- $(\pi, i) \models f\,W\,g$ iff $(\pi, i) \models [\,]f$ or $(\pi, i) \models f\,U\,g$

Semantics of Past Operators

- $(\pi, i) \models (-)f$ iff $i \geq 1$ and $(\pi, i-1) \models f$
- $(\pi, i) \models (\sim)f$ iff $i=0$ or $(\pi, i-1) \models f$
- $(\pi, i) \models \text{<->}f$ iff $(\pi, j) \models f$ for some j, $0 \leq j \leq i$
- $(\pi, i) \models [-]f$ iff $(\pi, j) \models f$ for all j, $0 \leq j \leq i$
- $(\pi, i) \models f \mathcal{S} g$ iff for some $k \leq i$, $(\pi, k) \models g$ and for all j, $k < j \leq i$, $(\pi, j) \models f$
- $(\pi, i) \models f \mathcal{B} g$ iff $(\pi, i) \models [-]f$ or $(\pi, i) \models f \mathcal{S} g$

Semantics of Quantifiers

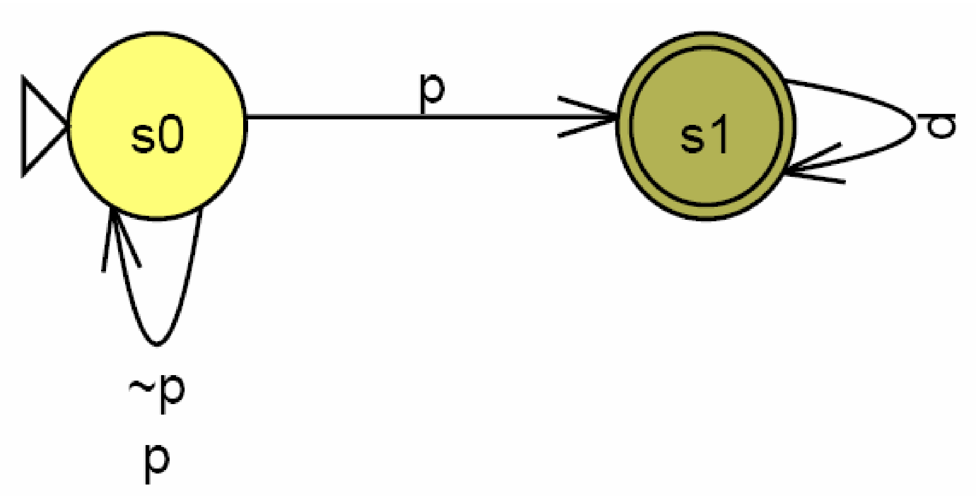
A sequence π' is a *p*-variant of π if for every $i \geq 0$, s_i' differs from s_i at most in the valuation of *p*.

- $(\pi, i) \models \exists p{:}\ f$ iff $(\pi', i) \models f$, for some *p*-variant π' of π
- $(\pi, i) \models \forall p{:}\ f$ iff $(\pi', i) \models f$, for every *p*-variant π' of π

Example 1: <>[]p

- Meaning: p always holds after some point
- Satisfying models:
  - $(p)^{\omega}$, i.e., ppp…
  - $p{\sim}p{\sim}pp{\sim}p(p)^{\omega}$
- Unsatisfying models:
  - $p{\sim}p{\sim}pp({\sim}pp)^{\omega}$

<>[]p as a Büchi Automaton



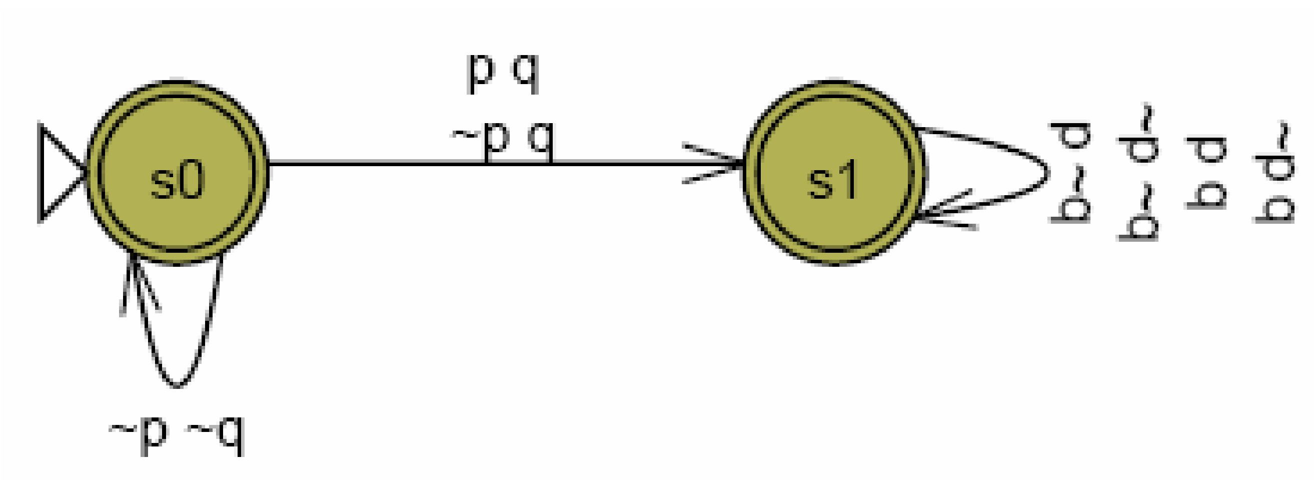
Example 2: [](p --> <->q)

- Meaning: Every p is preceded by a q.

- Satisfying models:
  - $(\sim p \sim q)^{\omega}$
  - $(\sim p \sim q)(\sim pq)(\sim p \sim q)(p \sim q)^{\omega}$

- Unsatisfying models:
  - $(\sim p \sim q)(p \sim q)\ldots$

[](p --> <->q) as a Büchi Automaton



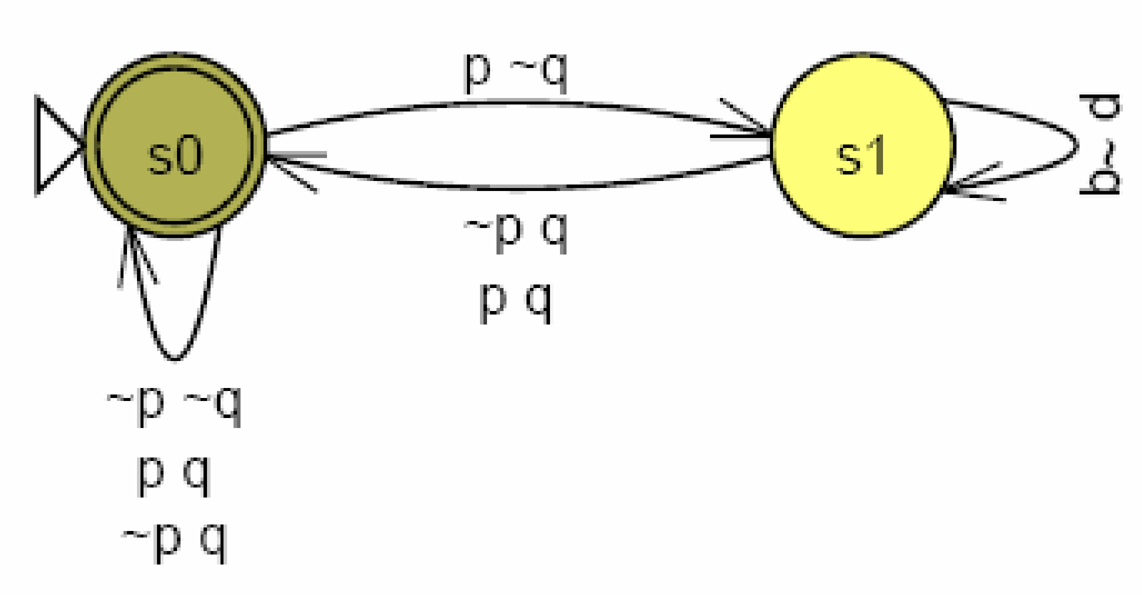
Example 3: [](p --> p *U* q)

- Meaning: Once p becomes true, it will remain true continuously until q becomes true, and q does become true.

- Satisfying models:
  - $(\sim p \sim q)^{\omega}$
  - $(\sim p \sim q)(p \sim q)(p \sim q)(p \sim q)(\sim pq)(\sim p \sim q)^{\omega}$

- Unsatisfying models:
  - (~p~q)(p~q)(~p~q)…

[](p --> p *U* q) as a Büchi Automaton



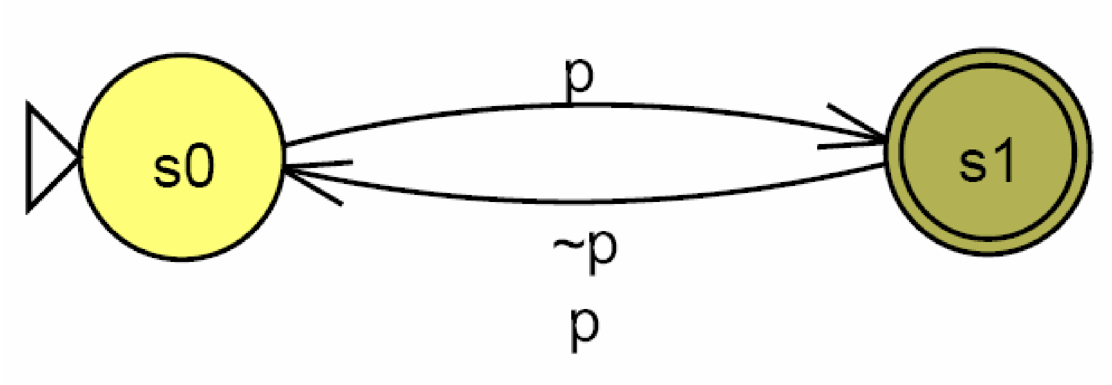
Example 4: “Even p”

- This is not PTL-expressible, but QPTL-expressible
- Meaning: p holds in every even state.

  (Note: the states of a sequence are numbered 0,1,2,3,…)

- Satisfying models:
  - $(p)^\omega$
  - $(p\sim p)^\omega$
  - $p\sim pp\sim p(pp)^\omega$
- Unsatisfying models:
  - $p\sim ppp\sim p(pp)^\omega$

“Even p” as a Büchi Automaton



“Even p” in Temporal Logic

- A QPTL formula for “Even p” (in GOAL’s textual format):

  E t: t /\ [](t <--> ()~t) /\ [](t --> p)

- A seemingly correct, but wrong, PTL formula:

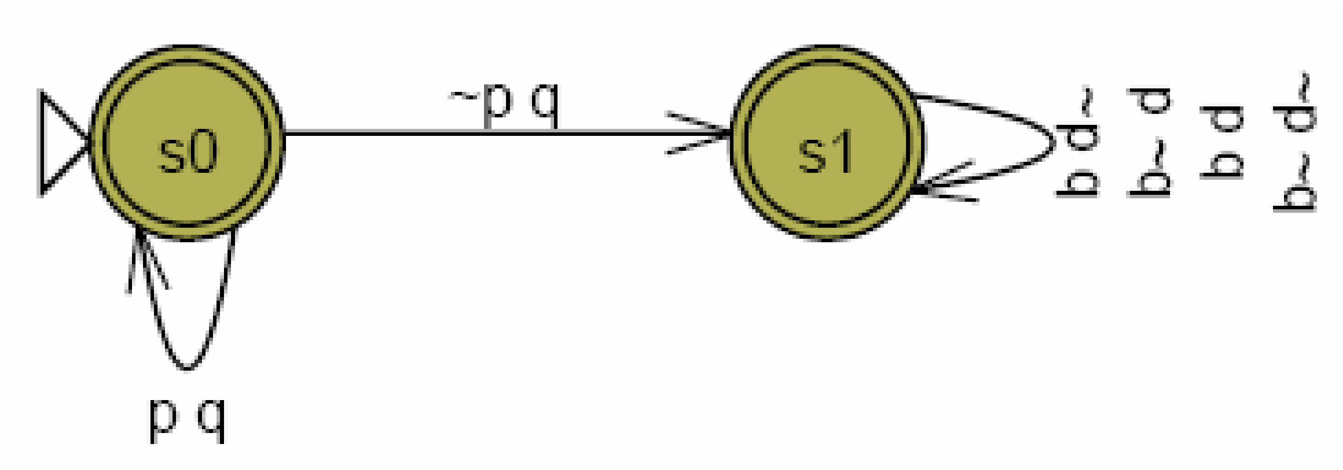
  p /\ [](p --> ()()p)

Example 5: Assume-Guarantee Spec.

- Equivalent formulations:
  - [](((~)[-]p --> q)
  - ~(p U ~q)
  - q W (~p /\ q)
- Meaning: q holds at least one step longer than p.

- Satisfying models:
  - ~pq(~p~q)$^{\omega}$
  - (pq)(pq)(pq)(pq)(~pq)(~p~q)$^{\omega}$
- Unsatisfying models:
  - (~p~q)(p~q)(~p~q)…

[]((~)[-]p --> q) as a Büchi Automaton



Demo Script

- Draw a BA, intended for <>[]p.
- Take a BA, for [](p --> <->q), from the repository.
- Check if it is correct, by comparing it with a machine-translated one.
- Try to specify “Even p” in PTL.
- See why it is incorrect.
- Perhaps more if time permits …

About Testing a BA on an Input

- To get an intuitive understanding of what language is being defined by the BA.
- Input format
  - Input string: ppp~pp(~pp)$^{\omega}$

    Real format: (p)(p)(p)(~p)(p){(~p)(p)}
  - Input string: (~pq)((~pq)(~p~q)(~p~q))$^{\omega}$

    Real format: (~p q){(~p q)(~p ~q)(~p ~q)}

Looking Back

- "…, there is nothing surprising about this tool …"
  – one anonymous reviewer
- True. (in terms of what the tool can do and how it does them)
- We were surprised that this kind of tools had not existed.
- Looking back, we are just glad that we made up our mind to create one.
- It has been a great learning experience for us.
- The **equivalence test** is particularly useful.

Looking Forward

GOAL is constantly being improved and extended; future work includes:

- GOAL as a comparative study platform:
  - More formulae to automata translation algorithms
  - More complementation algorithms
- Approaching the ultimate “GOAL” (**G**raphical Tool for **O**mega-**A**utomata and **L**ogics):
  - Manipulation of all common variants of ω-automata
  - Translation from S1S (SOLLO) formulae or ω -regular expressions to ω-automata

Thank You!

P.S. Go get the GOAL tool from its Web site: http://goal.im.ntu.edu.tw/

and let us know your suggestions.