Using and Avoiding Temporal Logic

Pierre Wolper

Université de Liège

Using temporal logic

- For specifying properties, . . .

Avoiding temporal logic

- When developing verification algorithms.
- By using the correspondence between temporal logic and automata.

Automata and Temporal Logic

Linear Time (LTL): A formula f can be translated to a Büchi automaton A_f (exponential in f).

- *Satisfiability:* A_f nonempty? (PSPACE-complete in the size of f).
- *Model Checking:* $K \times A_{\neg f}$ empty? (PSPACE-complete in $|f|$, NLOGSPACE in $|K|$).

Branching Time (CTL): A formula f can be translated to a Büchi tree automaton A_f (exponential in f).

- *Satisfiability:* A_f nonempty? (EXPTIME-complete in the size of f).
- *Model Checking:* $K \times A_{\neg f}$ empty? Yes, but *excessive complexity*! Solution: use alternating automata

Why Bother with Automata

Good method for extracting the combinatorial essence of Temporal Logic problems:

- simpler algorithms,
- easier implementations,
- better algorithms (on-the-fly, state compaction, partial-order methods, . . .),
- easier extensions (other logics, probabilistic logics and models, . . .),
- easier complexity analysis.

Many problems have been solved, many algorithms developed by using automata.

An application of automata: Deciding the closure of specifications (Work done with Doron Peled and Thomas Wilke)

Traces and trace equivalence (finite behaviors)

- We consider finite behaviors of a system that are words over an alphabet Σ
- A *dependency relation* $D \subseteq \Sigma \times \Sigma$ is given. The intuitive meaning of D is that is $(a, b) \notin D$, then exchanging adjacent occurrences of a and b leads to an equivalent behavior.
- Formally, for two words $u, v \in \Sigma^*$, $u \stackrel{1}{\equiv} v$ if there exist words w_1, w_2 and letters a, b such that $(a, b) \notin D$, $u = w_1 a b w_2$ and $v = w_1 b a w_2$,
- Trace equivalence $u \equiv v$ is the reflexive transitive closure of $\stackrel{1}{\equiv}$.

Partial-order methods and trace equivalence

- For a concurrent program, partial-order methods generate a reduced state space in which only some representatives of each trace equivalent class of behaviors are present.
- This allows properties to be checked reliably, except if the property can distinguish between trace equivalent behaviors.
- Two solutions:
  1. Modify the dependency relation to take the property into account. Correct, but leads to larger models
  2. Check that the property does not distinguish between elements of the same trace, i.e. is *trace closed*. Most correct properties of the system should satisfy this condition.

One thus needs an algorithm to check the trace closure of properties.

Checking the trace closure of properties of finite behaviors

How is the property expressed ? By an automaton !

Lemma Let $L \subseteq \Sigma^*$, then L is not $\equiv$-closed if and only if there exist words $u, v \in \Sigma^*$ such that $u \stackrel{1}{\equiv} v$, $u \in L$, and $v \notin L$.

Given the Lemma, the algorithm is simple

- Build an automaton on identical length pairs of words that checks that
  - The first word of the pair is in L,
  - The second word of the pair is in $\overline{L}$
  - The two words are related by $\stackrel{1}{\equiv}$
- Check whether this automaton is empty.

Complexity is PSPACE (nondeterministic automata or temporal logic)

Traces and infinite behaviors

What is the definition of trace equivalence on infinite traces ?

Definition The *limit extension* $\equiv^{\mathsf{lim}} \subseteq \Sigma^\omega \times \Sigma^\omega$ of an equivalence relation $\equiv \subseteq \Sigma^* \times \Sigma^*$ is defined by: $\alpha \equiv^{\mathsf{lim}} \beta$ if and only if

- for every $u \in \Sigma^*$ such that $u \sqsubset \alpha$ there exist $v, v' \in \Sigma^*$ such that $v \sqsubset \beta$ and $uv' \equiv v$, and
- for every $u \in \Sigma^*$ such that $u \sqsubset \beta$ there exist $v, v' \in \Sigma^*$ such that $v \sqsubset \alpha$ and $uv' \equiv v$.

A relation $\equiv$ on Σ^* is called a congruence relation if $uv \equiv u'v'$ whenever $u \equiv u'$ and $v \equiv v'$.

Lemma If $\equiv$ is a congruence relation on Σ^*, then $\equiv^{\mathsf{lim}}$ is an equivalence relation on Σ^ω.

Another definition of infinite trace equivalence

The previous definition is hard to use to decide trace closure. Here is another one.

Definition The *piecewise extension* $\equiv^{\omega *}$ of a binary relation $\equiv$ on Σ^* is the transitive closure of the relation $\equiv^{\omega} \subseteq \Sigma^{\omega} \times \Sigma^{\omega}$ defined by:

> $\alpha \equiv^{\omega}$ iff there exist decompositions $\alpha = u_0 u_1 u_2 \ldots$ and $\beta = v_0 v_1 v_2 \ldots$ such that $u_i \equiv v_i$ for every $i \geq 0$,

A quite nontrivial property is the following.

Theorem $\equiv^{\mathrm{lim}} \;=\; \equiv^{\omega *}$

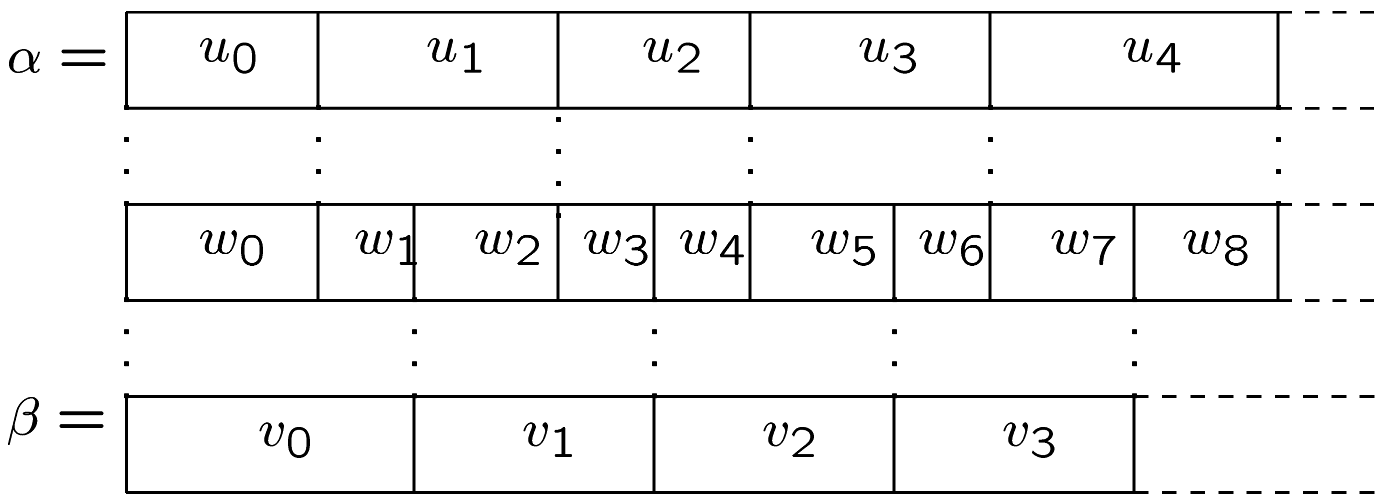
We will now focus on checking whether a specification is closed with respect to $\equiv^{\omega *}$

A hint at the proof of the theorem

1) If $\alpha \equiv^{\omega *} \beta$, then $\alpha \equiv^{\lim} \beta$.

Quite straightforward.

2) If $\alpha \equiv^{\lim} \beta$ then $\alpha \equiv^{\omega *} \beta$



Characterizing Languages closed under $\equiv^{\omega *}$

Intuitively, a language L is closed under $\equiv^{\omega *}$ if replacing a piece v of a word of L by a word $u \equiv v$ keeps the word in L. So we need to know which pieces of words can be substituted one for the other while preserving membership in L. The notion of *syntactic congruence* captures this.

Definition The *syntactic congruence* of a language $L \subseteq \Sigma^\omega$ is the relation $\approx_L$ over Σ^* defined by: $u \approx_L v$ iff

$$xuyz^\omega \in L \leftrightarrow xvyz^\omega \in L, \text{ and}$$
$$x(uy)^\omega \in L \leftrightarrow x(vy)^\omega \in L, \ (uy \neq \varepsilon \text{ and } vy \neq \varepsilon)$$

for $x, y \in \Sigma^*$, $z \in \Sigma^+$.

The following theorem relates syntactic congruence and $\equiv^{\omega *}$ for ω-regular languages.

Theorem Let $\equiv$ be a congruence relation on Σ^* and $L \subseteq \Sigma^\omega$ be an ω-regular language, then L is closed under $\equiv^{\omega *}$ if and only if $\equiv \subseteq \approx_L$.

Towards an algorithm

Similarly to what was done in the finite word case, one would like to replace checking nonclosure under $\equiv^{\omega *}$ by nonclosure under $\equiv^{\omega}$. It is possible to do even better. Using the syntactic congruence theorem, one can easily prove the following.

Theorem Let $\equiv$ be a congruence relation on Σ^* generated by a reflexive, symmetric relation $\stackrel{1}{\equiv}$ and $L \subseteq \Sigma^\omega$ an ω-regular language. Then L is closed under $\equiv^{\omega *}$ if and only if L is closed under $\stackrel{1}{\equiv}^{\omega}$.

Note that checking that $\alpha \stackrel{1}{\equiv}^{\omega} \beta$ can easily be done by a Büchi automaton operating on the pair (α, β). Indeed, one only needs to check that (α, β) can be decomposed into chuncks that are equivalent with respect to $\stackrel{1}{\equiv}$, i.e. that are identical except for the exchange of independent adjacent symbols.

The algorithm

To check that an ω-regular language L is trace-closed, proceed as follows.

- Build an automaton on pairs of infinite words that checks that
  - The first word of the pair is in L,
  - The second word of the pair is in $\overline{L}$
  - The two words are related by $\stackrel{1}{\equiv}{}^{\omega}$
- Check whether this automaton is empty.

Complexity is PSPACE (nondeterministic automata or temporal logic)

Conclusions

- Difficult problem
- Nontrivial theory leads to a very simple algorithm.
- Challenge: try doing this for temporal logic without going through automata.