Course ITI8531: Software Synthesis and Verification

Lecture 12: Acacia+ LTL Synthesis - I

Spring 2019

Leonidas Tsiopoulos

leonidas.tsiopoulos@taltech.ee

Avoiding the Classical Approach to LTL Synthesis

- LTL synthesis is a challenging problem due to 2EXPTIME theoretical complexity and lack of scalable algorithms for determinization of automata and solving games.
- There are some LTL-based synthesis approaches offering „Safraless“ solutions to avoid the very complex determinisation step and also better algorithms working on „symbolic“ representation of the state space during the game.
  - Even translating LTL formulae to symbolic automaton in the first place.
    - More for this and other „Safraless“ approaches in the 4th lecture.
- Acacia+ and the techniques around it is one such „Safraless“ approach.

Classical solution by Pnueli and Rosner

LTL

$2^{O(n)}$ ↓

Büchi Word Automata

$2^{O(m \log m)}$ ↓ [Safra, 88]

Det. Rabin Word Automata

Realizability

~

Rabin Game

The problem has been shown to be 2ExpTime-Complete by the same authors.

Acacia+: A tool for LTL synthesis

- Main contributions:
  - Efficient *symbolic* incremental algorithms based on *antichains* for game solving.
  - Synthesis of *small* winning strategies, when they exist.
  - *Compositional* approach for *large conjunctions* of LTL formulas.
  - Performance is better or similar to other existing tools but its *main advantage* is the generation of *compact strategies.*
- Application scenarios:
  - **Synthesis of control code from high-level LTL specifications**.
  - *Debugging* of unrealizable specifications by inspecting compact counter strategies.
  - *Generation of small deterministic automata* from LTL formulas, when they exist.

Acacia+ Safraless approach

LTL

$\downarrow 2^{O(n)}$

Universal coBüchi Word automata

$\downarrow O(1)$

Universal **KcoBüchi** Word automata

$\downarrow 2^{O(m^{\wedge}2)}$

Det. KcoBüchi Word automata

Realizability

~

Safety game

- Safety games are the simplest games to solve!
- Details and comparison to other games of other LTL-based synthesis approaches in Lectures III and IV

Acacia+ Safraless approach

LTL

$\downarrow 2^{O(n)}$

Universal coBüchi Word automata

$\downarrow O(1)$

Universal **KcoBüchi** Word automata

$\downarrow 2^{O(m^2)}$

Det. KcoBüchi Word automata

Doubly Exponential Size ...

... Game solved on-the-fly with **antichain** technics

- Safety games are the simplest games to solve!
- Details and comparison to other games of other LTL-based synthesis approaches in Lectures III and IV

Acacia+ and LTL Transformation to Automata (1)

- An *infinite word automaton* is a tuple $A = (\Sigma, Q, q_0, \alpha, \delta)$ where:
  - Σ is the *finite alphabet*,
  - Q is a *finite set of states*,
  - $q_0 \in Q$ is the *initial* state,
  - $\alpha \subseteq Q$ is a set of *final states* and
  - $\delta \subseteq Q \times \Sigma \times Q$ is the *transition relation*.
    - For all $q \in Q$ and all $\sigma \in \Sigma$, $\delta(q, \sigma) = \{q' \mid (q, \sigma, q') \in \delta\}$.

- A is *deterministic* if $\forall q \in Q \cdot \forall \sigma \in \Sigma \cdot |\delta(q, \sigma)| \leq 1$.
- A is *complete* if $\forall q \in Q \cdot \forall \sigma \in \Sigma \cdot \delta(q, \sigma) = \emptyset$.

Acacia+ and LTL Transformation to Automata (2)

- A *run* of A on a *word* $w = \sigma_0\sigma_1 \cdot\cdot\cdot \in \Sigma^\omega$ is an infinite sequence of states $\rho = \rho_0\rho_1 \cdot\cdot\cdot \in Q^\omega$ such that $\rho_0 = q_0$ and $\forall i \geq 0 \cdot q_{i+1} \in \delta(q_i, \rho_i)$.
- The *set of runs* of A on w is denoted by $\mathrm{Runs}_A(w)$.
- The number of times state q occurs along run ρ is denoted by $\mathrm{Visit}(\rho, q)$.
- Three *acceptance conditions* (a.c.) are considered for infinite word automata. A word w is *accepted by* A if:
  - Non-deterministic Büchi : $\exists\rho \in \mathrm{Runs}_A(w) \cdot \exists q \in \alpha \cdot \mathrm{Visit}(\rho, q) = \infty$
    - Runs visits final states infinitely often.
  - Universal Co-Büchi : $\forall\rho \in \mathrm{Runs}_A(w) \cdot \forall q \in \alpha \cdot \mathrm{Visit}(\rho, q) < \infty$
    - Runs visit final states finitely often.
  - Universal K-Co-Büchi : $\forall\rho \in \mathrm{Runs}_A(w) \cdot \forall q \in \alpha \cdot \mathrm{Visit}(\rho, q) \leq K$
    - Runs visit at most K final states.

Acacia+ and LTL Transformation to Automata (3)

- The *set of words* accepted by A with the non-deterministic Büchi a.c. is denoted by $L_{\mathrm{b}}(A)$.
  - This implies that A is a non-deterministic Büchi word automaton (NBW).

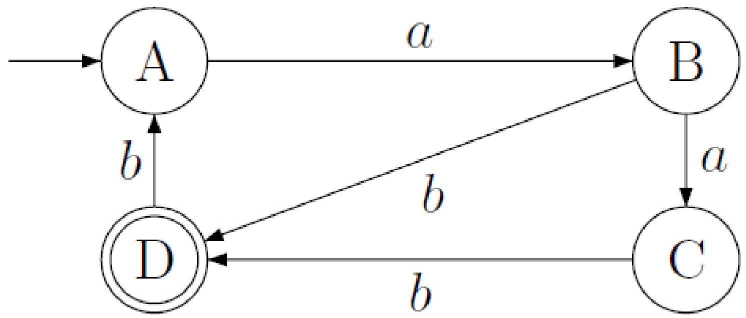
- Similarly, the set of words accepted by A with the universal co-Büchi and universal K-co-Büchi a.c., are denoted respectively by $L_{\mathrm{uc}}(A)$ and $L_{\mathrm{uc},K}(A)$.
  - With those interpretations, A is a universal co-Büchi automaton (UCW) and that (A,K) is a universal K-co-Büchi automaton (UKCW) respectively.

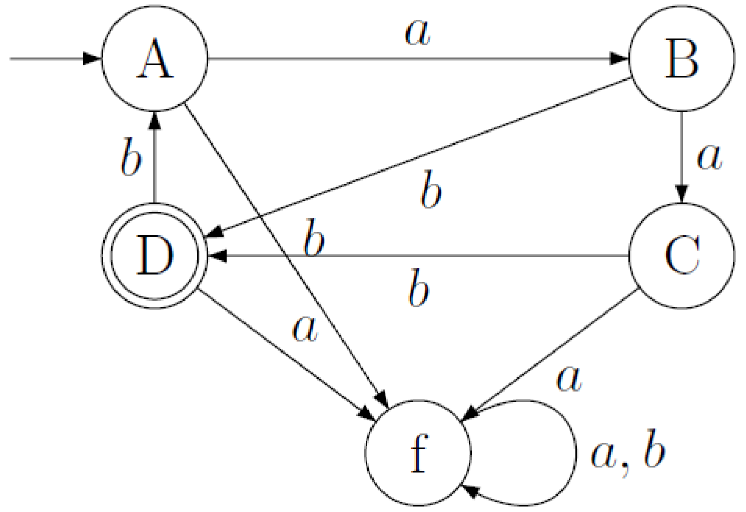
- By duality, $L_{\mathrm{b}}(A) = \overline{L_{\mathrm{uc}}(A)}$ for any infinite word automaton A.

- Also, for any $0 \leq K_1 \leq K_2$, $L_{\mathrm{uc},K_1}(A) \subseteq L_{\mathrm{uc},K_2}(A) \subseteq L_{\mathrm{uc}}(A)$.

Example of a NBW

- On input *aabbaabb* . . . the NBW shown has only the run: ABCDABCDABCD
- The language recognized by the NBW is: {*aabbaabbaabb* . . .}
- Is this NBW complete?
- No.
- Completing this NBW we obtain:





On any infinite input word the accepting runs of both NBWs correspond, because any run that reaches f stays in f, and since f is not an accepting state, such a run is not accepting.

Infinite automata and LTL

- NBWs subsume LTL, i.e., for an LTL formula φ, there is a NBW A_φ (possibly exponentially larger) such that $L_\mathrm{b}(A_\varphi) = \{w | w \vDash \varphi\}$.

- By duality, one can associate an equivalent UCW with any LTL formula φ:
  - Take $A_{\neg\varphi}$ with the universal co-Büchi a.c., so
    - $L_\mathrm{uc}(A_{\neg\varphi}) = \overline{L_\mathrm{b}(A_{\neg\varphi})} = L_\mathrm{b}(A_\varphi) = \{w | w \vDash \varphi\}$.

Turn-based Automata for Realizability of Games (1)

- To reflect the game point of view of the realizability problem the notion of *turn-based automata* is used to define the specification.
- *A turn-based automaton* A over the *input alphabet* Σ_I and the *output alphabet* Σ_O is a tuple $A = (\Sigma_I, \Sigma_O, Q_I, Q_O, q_0, \alpha, \delta_I, \delta_O)$ where:
  - Q_I, Q_O are *finite sets of input and output states* respectively,
  - $q_0 \in Q_O$ the *initial state*,
  - $\alpha \subseteq Q_I \cup Q_O$ is the set of *final states*,
  - $\delta_I \subseteq Q_I \times \Sigma_I \times Q_O$ and $\delta_O \subseteq Q_O \times \Sigma_O \times Q_I$ are the *input* and *output transition relations.*
- A is *complete* if for all $q_I \in Q_I$, and all $\sigma_I \in \Sigma_I$, $\delta_I(q_I, \sigma_I) \neq \emptyset$, **and** for all $q_O \in \Sigma_O$ and all $\sigma_O \in \Sigma_O$, $\delta_O(q_O, \sigma_O) \neq \emptyset$.

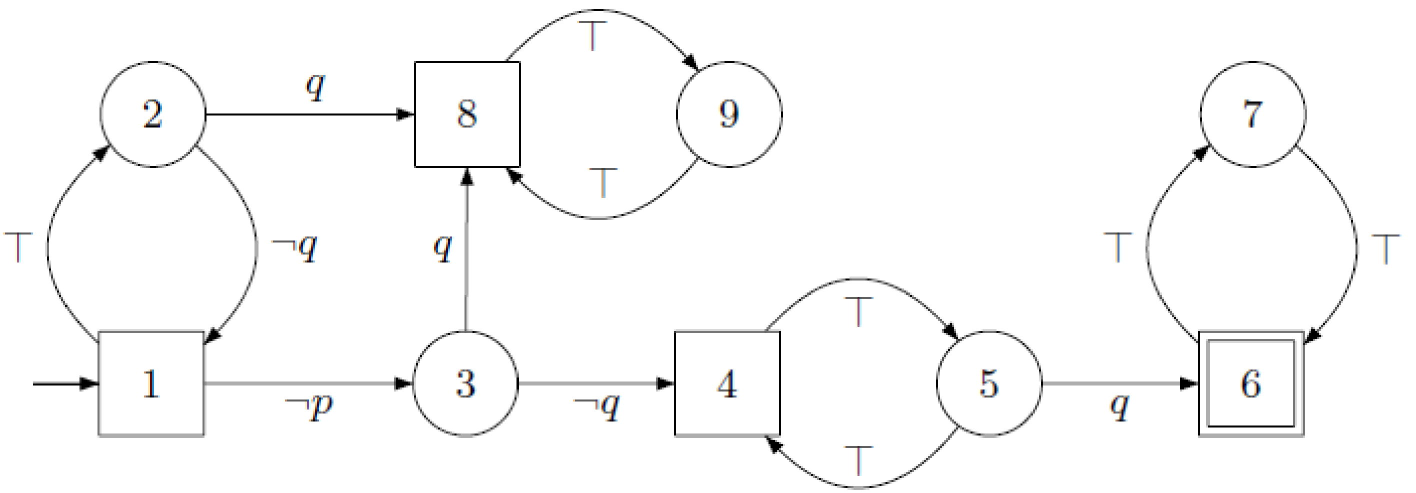
Turn-based Automata for Realizability of Games (2)

- Turn-based automata A run on words from Σ^ω.
- A *run* on a word $w = (o_0 \cup i_0)(o_1 \cup i_1) \cdot \cdot \cdot \in \Sigma^\omega$ is an infinite sequence of states $\rho = \rho_0\rho_1 \cdot \cdot \cdot \in (Q_O Q_I)^\omega$ such that $\rho_0 = q_0$ and for all $j \geq 0$,

  $(\rho_{2j}, o_j, \rho_{2j+1}) \in \delta_O$ and $(\rho_{2j+1}, i_j, \rho_{2j+2}) \in \delta_I$.
- All acceptance conditions we show carry over to turn-based automata.
- Every UCW (resp. NBW) with state set Q and transition set Δ is equivalent to a turn-based UCW (tbUCW) (resp. tbNBW) with $|Q| + |\Delta|$ states:
  - the new set of states is $Q \cup \Delta$,
  - final states remain the same,
  - and each transition $r = q \xrightarrow{\sigma_i \cup \sigma_o} q' \in \Delta$ where $\sigma_o \in \Sigma_O$ and $\sigma_i \in \Sigma_I$ is split into a transition $q \xrightarrow{\sigma_o} r$ and a transition $r \xrightarrow{\sigma_i} q'$.

Example of tbUCW

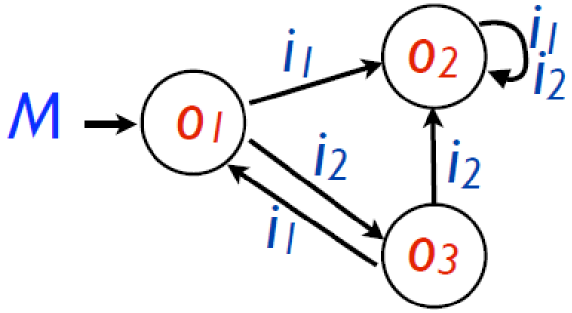
- tbUCW for $\mathbf{F}q \rightarrow (p\mathbf{U}q)$ where $I = \{q\}$ and $O = \{p\}$
- Output states $Q_O = \{1, 4, 6, 8\}$ are depicted by squares and input states $Q_I = \{2, 3, 5, 7, 9\}$ by circles
- $\top$ stands for the sets Σ_I or Σ_O, depending on the context, $\neg q$ (resp. $\neg p$) stands for the sets that do not contain q (resp. p), i.e. the empty set.
- At state 1, if controller does not assert p and next the environment does not assert q, then the run is in state 4. From this state, whatever the controller does, if the environment asserts q, then the controller loses, as state 6 will be visited infinitely often.



- A strategy for the controller is to assert p all the time, therefore the runs will loop in states 1 and 2 until the environment asserts q. Afterwards the runs will loop in states 8 and 9, which are non-final.

Finite state strategies

- We know that if an LTL formula is realizable, there exists a finite-state strategy that realizes it [PR89].
- Finite-state strategies are represented as complete Moore machines in Acacia+.



$L(M)$ = *traces of infinite paths*

E.g. $(o_I \cup i_I)(o_2 \cup i_2)^{\omega}$

- The LTL realizability problem reduces to decide, given a tbUCW A over inputs Σ_I and outputs Σ_O, whether there is a non-empty Moore machine M such that $L(M) \subseteq L_{uc}(A)$.
- The tbUCW is equivalent to an LTL formula given as input and is constructed by using tools *Wring* or *LTL2BA*.

Bounding the number of *visited* final states

Lemma 1. Given a Moore machine M with m states, and a tbUCW A with n states, if $L(M) \subseteq L_{\text{uc}}(A)$, then all runs on words of $L(M)$ visit at most $m \times n$ final states.

Proof. The infinite paths of M starting from the initial state define words that are accepted by A. Therefore in the product of M and A, there is no cycle visiting an accepting state of A, which allows one to bound the number of visited final states by the number of states in the product.

Corollary. $L(M) \subseteq L_{\text{uc}}(A)$ iff $L(M) \subseteq L_{\text{uc},\, m\text{x}n}(A)$

Reduction to a bounded universal *K*-co-Büchi automaton

Lemma 2. Given a realizable tbUCW A over *inputs* Σ_I and *outputs* Σ_O with n states, there exists a non-empty Moore machine with at most $n^{2n+2} + 1$ states that realizes it.

***Proof**. In the paper. Re-using an older result by Safra.*

Theorem. Let A be a tbUCW over Σ_I, Σ_O with n states and $K = 2n(n^{2n+2} + 1)$ (from above proof). Then A is realizable iff (A,K) is realizable.

Determinization of U*K*CWs

- What is left is to reduce the tbU*K*CW realizability problem to a safety game.
- It is based on the determinization of tbU*K*CWs into complete turn-based deterministic 0-Co-Büchi automata, which can also be viewed as safety games.
- The resulting deterministic automaton is always equipped with a *partial-order on states* that can be used to efficiently manipulate its state space using the antichain method.

- *Details in the next lecture.*

References

- An Antichain Algorithm for LTL Realizability . http://lit2.ulb.ac.be/acaciaplus/slides/cav09.pdf

  Slides of presentation of the following paper at CAV 2009 conference.
- Filiot E., Jin N., Raskin JF. (2009) An Antichain Algorithm for LTL Realizability. In: Bouajjani A., Maler O. (eds) Computer Aided Verification. CAV 2009. Lecture Notes in Computer Science, vol 5643. Springer, Berlin, Heidelberg.
  - This is one of three main papers regarding this part of the course.
- S. Safra, On the complexity of ω-automata. In: Proc. 29th Annual Symposium on Foundations of Computer Science (FOCS), IEEE Computer Society Press (1988).
- A. Pnueli and R. Rosner. On the synthesis of a reactive module. In Proceedings of the 16th ACM SIGPLAN-SIGACT symposium on Principles of programming languages (POPL '89) ACM, NY, USA, 179-190. DOI=http://dx.doi.org/10.1145/75277.75293, 1989