An Automata Theoretic Approach to Branching Time Model Checking

Arie Gurfinkel

arie@cs.toronto.edu

University of Toronto

CTL Satisfiability

- Bounded branching property of CTL
  - a CTL formula φ is satisfiable iff it is satisfiable on a tree with branching degree bounded by $|\varphi|$
- Satisfiability via alternating automata
  - Given a CTL formula φ
    - build an ATA A_φ that works over $|\varphi|$-ary trees
    - φ is satisfiable iff A_φ is non-empty
- Non-emptiness problem for ATA is EXPTIME-complete
  - satisfiability problem for CTL is in EXPTIME

CTL Model-Checking

- K is a Kripke structure with branching degrees in $\mathcal{D}$
- φ is a CTL formula
- Build an ATA $A_{\mathcal{D},\varphi}$ such that
  - $\mathcal{L}(A_{\mathcal{D},\varphi})$ contains all $\mathcal{D}$-trees that satisfy φ
- $K \models \varphi$ iff the computation tree T_K induced by K is in $\mathcal{L}(A_{\mathcal{D},\varphi})$
- Automata-based model-checking algorithm
  - build a product automaton of K and $A_{\mathcal{D},\varphi}$ whose language is $\mathcal{L}(A_{\mathcal{D},\varphi}) \cap T_K$
  - check if it is empty or not

Complexity

- In general, checking non-emptiness of an ATA is expensive
- But, we are dealing with a special case
  - only interested in automata that arise from CTL formulas
  - the language of the product automaton either empty, or contains a single tree
- Using the above we obtain a linear automata-based CTL algorithm

Weak Alternating Automata

- A Büchi ATA $A = (\Sigma, Q, q_0, \delta, F)$ is weak iff
  - exists a partitioning of Q into $Q_1, Q_2, \ldots, Q_n$
  - each Q_i is either accepting
    - $Q_i \subseteq F$
  - or rejecting
    - $Q_i \cap F = \emptyset$
  - there exists a partial order such that
    - if $q \in Q_i$, $q_j \in Q_j$, and q_j appears in the transition of q, then $i \geq j$
- Any run of a Weak Alternating Automaton (WAA) gets trapped in either accepting or rejecting state

Weak Alternating Automata and CTL

- ATA constructed for a CTL formula φ is weak
  - each formula in $cl(\varphi)$ forms a singleton set in the partition
  - the partial order is given by the sub-formula ordering
- Let $\varphi = EF(AGp)$, then
  - the partition is $\{EF(AGp)\} > \{AGp\} > \{p\}$
  - $\{AGp\}$ is the accepting set
  - $\{EF(AGp)\}$, and $\{p\}$ are rejecting

The Product Automaton

- The product automaton of ATA A_φ and a Kripke structure K is
  - weak if A_φ is weak
  - Büchi if A_φ is Büchi
  - an automaton over infinite words!
  - a 1-letter automaton!
    - i.e. $\Sigma = \{a\}$
- The product automaton is weak alternating Büchi word automaton

Constructing the Product

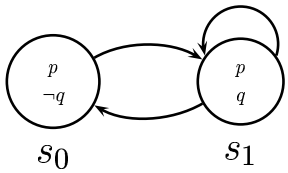
- Let $A_\varphi = (2^{AP}, Q, q_0, \delta_\varphi, F)$ be a WAA for CTL formula φ
- Let $K = (AP, S, s_0, L, R)$ be a Kripke structure
- The product automaton is $A_{K,\varphi} = (\{a\}, Q \times S, \langle q_0, s_0 \rangle, \delta, S \times F)$
  - if $\delta_\varphi(q, L(s), k) = \theta$
  - and $R(s) = t_0, \ldots, t_k$, then
  - $\delta(\langle q, s \rangle, a) = \theta'$,
    - where θ' is obtained from θ by replacing each (c, q') with $\langle q', t_c \rangle$
- The weakness partition is induced by A_φ
- The product automaton simulates the run of A_φ on the computation tree of K

Example

- $\varphi = EFq$

state	$\delta(q, \emptyset, k)$	$\delta(q, \{q\}, k)$
q	**false**	**true**
EFq	$\bigvee_{c=0}^{k-1}(c, EFq)$	**true**

- Kripke Structure



- Product automaton

state	$\delta(q, a)$
$\langle EFq, s_0\rangle$	$\langle EFq, s_1\rangle$
$\langle EFq, s_1\rangle$	**true**

Example

- $\varphi = EGAXp$

state	$\delta(q, \emptyset, k)$	$\delta(q, \{p\}, k)$
p	**false**	**true**
AXp	$\bigwedge_{c=0}^{k-1}(c, p)$	$\bigwedge_{c=0}^{k-1}(c, p)$
$EGAXp$	$\bigwedge_{c=0}^{k-1}(c, p) \wedge \bigvee_{c=0}^{k-1}(c, EGAXp)$	$\bigwedge_{c=0}^{k-1}(c, p) \wedge \bigvee_{c=0}^{k-1}(c, EGAXp)$

- Product automaton

state	$\delta(q, a)$
$\langle p, s_0\rangle$	**true**
$\langle p, s_1\rangle$	**true**
$\langle EGAXp, s_0\rangle$	$\langle p, s_1\rangle \wedge \langle EGAXp, s_1\rangle$
$\langle EGAXp, s_1\rangle$	$\langle p, s_0\rangle \wedge \langle p, s_1\rangle \wedge (\langle EGAXp, s_0\rangle \vee \langle EGAXp, s_1\rangle)$

Non-Emptiness Algorithm

- The algorithm proceeds up in the weakness partial order
  - each state is labeled with either $\mathbf{true}$ or $\mathbf{false}$
  - an automaton is non-empty iff its initial state is labeled with $\mathbf{true}$

Non-Emptiness Algorithm

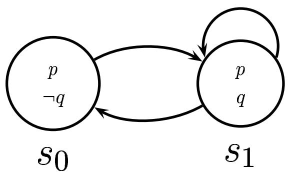
- The algorithm
  - pick a Q_i with the smallest i that is not yet labeled
  - repeatedly, for each $q \in Q_i$
    - label q with $\mathbf{true}$ is $\delta(q, a) = \mathbf{true}$
    - label q with $\mathbf{false}$ is $\delta(q, a) = \mathbf{false}$
    - use the labeling on q to simplify any δ in which q occurs
  - if there are any unlabeled states in Q_i
    - if Q_i is accepting, label them with $\mathbf{true}$
    - if Q_i is rejecting, label them with $\mathbf{false}$
- Complexity
  - linear in the size of property automaton and the Kripke structure

Example

- $\varphi = EFq$

state	$\delta(q, \emptyset, k)$	$\delta(q, \{q\}, k)$
q	**false**	**true**
EFq	$\bigvee_{c=0}^{k-1}(c, EFq)$	**true**

- Kripke Structure



- Product automaton

state	$\delta(q, a)$
$\langle EFq, s_0\rangle$	$\langle EFq, s_1\rangle$
$\langle EFq, s_1\rangle$	**true**

Example

- $\varphi = EGAXp$

state	$\delta(q, \emptyset, k)$	$\delta(q, \{p\}, k)$
p	**false**	**true**
AXp	$\bigwedge_{c=0}^{k-1}(c, p)$	$\bigwedge_{c=0}^{k-1}(c, p)$
$EGAXp$	$\bigwedge_{c=0}^{k-1}(c, p) \wedge \bigvee_{c=0}^{k-1}(c, EGAXp)$	$\bigwedge_{c=0}^{k-1}(c, p) \wedge \bigvee_{c=0}^{k-1}(c, EGAXp)$

- Product automaton

state	$\delta(q, a)$
$\langle p, s_0\rangle$	**true**
$\langle p, s_1\rangle$	**true**
$\langle EGAXp, s_0\rangle$	$\langle p, s_1\rangle \wedge \langle EGAXp, s_1\rangle$
$\langle EGAXp, s_1\rangle$	$\langle p, s_0\rangle \wedge \langle p, s_1\rangle \wedge (\langle EGAXp, s_0\rangle \vee \langle EGAXp, s_1\rangle)$

Space Efficient Algorithm

- For LTL we have an on-the-fly algorithm that only builds as much of the structure as necessary
- Is this possible for CTL?
- For a long time it was considered that the bottom-up nature of the CTL algorithm requires to construct the full structure first!
- With HAA we show that a space efficient algorithm for CTL is possible
- Actually, the same algorithm applies to CTL*
- But, not to the alternation free μ-calculus!

Hesitation Partition

- $A_{K,\varphi} = K \times A_{\mathcal{D},\varphi}$ — product automaton of K and φ
  - each state is an element of $S \times cl(\varphi)$
  - weakness partition based on the second component
  - at most $|cl(\varphi)|$ elements of the partition
- Each partition set Q_i can be classified as
  - *transient* – all transitions lead to states in lower Q_i's
    - corresponds to all elements of $cl(\varphi)$ except for U and R formulas
  - *existential* – a transition only contains disjunctively related elements of the same Q_i
    - corresponds to EU and ER formulas
  - *universal* – a transition only contains conjunctively related elements of the same Q_i
    - corresponds to AU and AR formulas

Hesitant Automata

- $A = (\Sigma, \mathcal{D}, Q, \delta, q_0, \langle G, B \rangle)$ is a hesitant automaton iff
  - Q can be partitioned into sets Q_i
    - each Q_i is either transient, existential, or universal
  - there exists a partial order on the partition such that transitions in Q_i lead either to the same Q_i, or to a lower one
  - $G, B \subseteq Q$ is a acceptance condition
  - the partial order is called *hesitation order*
  - the longest chain in it is the *hesitation depth*
- Each infinite path along a run of a HAA is trapped in either existential or universal Q_i
  - an infinite path π is accepting if
    - Q_i is existential and $Inf(\pi) \cap G \neq \emptyset$
    - Q_i is universal and $Inf(\pi) \cap B = \emptyset$

From CTL to HAA

- The WAA constructed from a CTL formula already satisfies the hesitation partition
  - we only need to change the acceptance condition
- The acceptance condition of WAA contains all ER and AR sub-formulas
  - a path on a run is allowed to get trapped only in a set corresponding to ER or AR formulas

From CTL to HAA

- In HAA we get
  - a path can get trapped in an existential set iff it corresponds to ER formula
  - a path can get trapped in a universal set iff it does not correspond to AU formula
- The acceptance condition for HAA is $\langle G, B \rangle$
  - G contains all ER sub-formulas
  - B contains all AU sub-formulas
- The transition relation of HAA is more constrained than of WAA, but its acceptance condition is more expressive

Example

- $\varphi = EFq$

state	$\delta(q, \emptyset, k)$	$\delta(q, \{q\}, k)$
q	**false**	**true**
EFq	$\bigvee_{c=0}^{k-1}(c, EFq)$	**true**

  - acceptance condition $(\emptyset, \emptyset)$
- $\varphi = EGAXp$

state	$\delta(q, \emptyset, k)$	$\delta(q, \{p\}, k)$
p	**false**	**true**
AXp	$\bigwedge_{c=0}^{k-1}(c, p)$	$\bigwedge_{c=0}^{k-1}(c, p)$
$EGAXp$	$\bigwedge_{c=0}^{k-1}(c, p) \wedge \bigvee_{c=0}^{k-1}(c, EGAXp)$	$\bigwedge_{c=0}^{k-1}(c, p) \wedge \bigvee_{c=0}^{k-1}(c, EGAXp)$

  - acceptance condition $(\{EGAXp\}, \emptyset)$

Non-Emptiness Algorithm for HAA

- Intuition
  - for a state from an existential set look for a witness
    - if one is found, label the state with $\mathbf{true}$
    - otherwise, label it with $\mathbf{false}$
  - for a state from a universal set look for a counterexample
    - if one is found, label the state with $\mathbf{false}$
    - otherwise, label it with $\mathbf{true}$
  - an automaton is non-empty iff the initial state is labeled with $\mathbf{true}$
- Space complexity is $O(m \log^2 n)$
  - m is the depth of the automaton
  - n is its size

Immediate Reachability

- Let q and q' be states of the same Q_i
- q' is *immediately reachable* from q iff
  - when $\delta(q, \sigma, k)$ is simplified using values of the states from the lower Q_i
  - q' appears in δ
  - i.e. the value of δ depends on q'
- q' is *reachable* from q if there exists a path of immediate reachable states from q to q'
- A state q is provably $\mathbf{true}$ if its transition simplifies to $\mathbf{true}$
- A state q is provable $\mathbf{false}$ if its transition simplifies to $\mathbf{false}$

Non-Emptiness Algorithm

- Start at the initial state
- If q is a transient state recurse to all successors and simplify the transition relation
- If q is from an existential Q_i
  - if there exists reachable state q' in the same Q_i that is provably $\mathbf{true}$
    - label q with $\mathbf{true}$
  - if not, search for a reachable state $q' \in G$ in the same Q_i that is reachable from itself
    - if found, label q with $\mathbf{true}$
    - otherwise, label q with $\mathbf{false}$

Non-Emptiness Algorithm

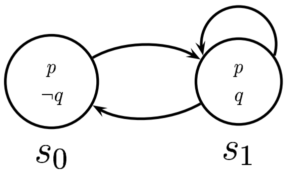
- If q is from a universal Q_i
  - if there exists reachable state q' in the same Q_i that is provably $\mathbf{false}$
    - label q with $\mathbf{false}$
  - if not, search for a reachable state $q' \in B$ in the same Q_i that is reachable from itself
    - if found, label q with $\mathbf{false}$
    - otherwise, label q with $\mathbf{true}$

Example

- $\varphi = EFq$

state	$\delta(q, \emptyset, k)$	$\delta(q, \{q\}, k)$
q	**false**	**true**
EFq	$\bigvee_{c=0}^{k-1}(c, EFq)$	**true**

- Kripke Structure



- Product automaton

state	$\delta(q, a)$
$\langle EFq, s_0\rangle$	$\langle EFq, s_1\rangle$
$\langle EFq, s_1\rangle$	**true**

Example

- $\varphi = EGAXp$

state	$\delta(q, \emptyset, k)$	$\delta(q, \{p\}, k)$
p	**false**	**true**
AXp	$\bigwedge_{c=0}^{k-1}(c, p)$	$\bigwedge_{c=0}^{k-1}(c, p)$
$EGAXp$	$\bigwedge_{c=0}^{k-1}(c, p) \wedge \bigvee_{c=0}^{k-1}(c, EGAXp)$	$\bigwedge_{c=0}^{k-1}(c, p) \wedge \bigvee_{c=0}^{k-1}(c, EGAXp)$

- Product automaton

state	$\delta(q, a)$
$\langle p, s_0\rangle$	**true**
$\langle p, s_1\rangle$	**true**
$\langle EGAXp, s_0\rangle$	$\langle p, s_1\rangle \wedge \langle EGAXp, s_1\rangle$
$\langle EGAXp, s_1\rangle$	$\langle p, s_0\rangle \wedge \langle p, s_1\rangle \wedge (\langle EGAXp, s_0\rangle \vee \langle EGAXp, s_1\rangle)$

Summary

- Automata over infinite objects
  - many possible acceptance conditions
  - more expressive acceptance conditions lead to simpler automata for the same language
- Alternating automata
  - extend non-determinism by allowing both disjunctive and conjunctive choice
  - greatly simplify constructing property automata

Summary – Model-Checking

- automata provide a uniform solution to the model-checking problem
- branching versus linear time is captured by
  - automata over strings, and
  - automata over trees
- same solution to both satisfiability and model-checking
- a formula φ is satisfiable iff
  - an automaton corresponding to φ is non-empty
- a model K satisfies a formula φ iff
  - the product automaton of K and φ is non-empty

Summary – Model-Checking

- clean separation between logic and algorithms
  - what does the formula mean?
    - how to construct an automaton for it
  - what is the complexity of model-checking
    - solving the non-emptiness problem