On the Complexity of Intersection Non-Emptiness Problems

Michael Wehar
University at Buffalo

Summer Talk @ IBM Almaden

June 11, 2016

Overview

The General Problem

Intersection Non-Emptiness for DFA's

Given a finite list of DFA's, is there a string that simultaneously satisfies all the DFA's?

- We denote this problem by $\mathrm{IE}_{\mathcal{D}}$.
- We use n for the total length of the input's encoding.
- We use k for the number of DFA's in the list.

Simple Solution

Intersection Non-Emptiness for DFA's

Given a finite list of DFA's, is there a string that simultaneously satisfies all the DFA's?

- Let DFA's D_1, D_2, ..., D_k be given.
- Construct the Cartesian product DFA $\mathcal{D}$.
- If each DFA has at most m states, then the product has at most m^k states.
- $\mathcal{D}$ accepts a string $x \Leftrightarrow D_i$ accepts x for each $i \in [k]$.
- We just need to check is $\mathcal{D}$ is satisfiable.

Hardness and Insights

Intersection Non-Emptiness for DFA's

Given a finite list of DFA's, is there a string that simultaneously satisfies all the DFA's?

- It is a classic PSPACE-complete problem [Kozen 77].
- There are two ways of viewing the problem:
  - Directed Reachability: Can I get from a start state to a final state in the product graph?
  - Constraint Satisfaction: Is there a string that satisfies all of the DFA's?
- We use k-$\mathrm{IE}_{\mathcal{D}}$ to denote the problem for fixed k-many DFA's.

Fixed Parameter Problem

Theorem

Solving k-$\mathrm{IE}_{\mathcal{D}}$ is equivalent to simulating a NTM that uses $k\log(n)$ bits of memory.[a]

[a] Originally proven by Kasai and Iwata in 1985.

- Let a $k\log(n)$-space bound NTM M be given.
- Let an input string x of length n be given.
- A computation of M on x is a sequence of configurations.
- Each configuration includes the tape content.

Fixed Parameter Problem

Theorem

Solving k-$\mathrm{IE}_{\mathcal{D}}$ is equivalent to simulating a NTM that uses $k\log(n)$ bits of memory.[a]

[a] Originally proven by Kasai and Iwata in 1985.

- The tape is a sequence of $k\log(n)$ bits.
- We can break up this sequence into k regions.
- Each region will consist of $\log(n)$ bits from the tape.
- We build k DFA's to collectively verify a "computation".
- We assign one DFA to each region.

Overview

The Problem Plus a Stack

Intersection Non-Emptiness for DFA's and One PDA

Given a finite list of DFA's and one PDA, is there a string that simultaneously satisfies all of the automata?

- We denote this problem by $\mathrm{IE}_{\mathcal{P}}$.
- We use n for the total length of the input's encoding.
- We use k for the number of DFA's in the list.

The Problem Plus a Stack

Intersection Non-Emptiness for DFA's and One PDA

Given a finite list of DFA's and one PDA, is there a string that simultaneously satisfies all of the automata?

- We can solve $\mathrm{IE}_{\mathcal{P}}$ by building a product PDA.
- This variation of the problem is $\mathrm{EXPTIME}$-complete.
- We use k-$\mathrm{IE}_{\mathcal{P}}$ to denote the problem for fixed k-many DFA's.

Verifying Computations

- We described how to build k DFA's that verify a $k\log(n)$-space bounded NTM's computation.
- We will show that with k DFA's and one PDA, we can verify a $k\log(n)$-space bounded AuxPDA's computation.
- An auxiliary PDA has a two-way read only input tape, a stack, and a two-way read/write auxiliary binary work tape.

Fixed Parameter Problem

Theorem

Solving $k\text{-IE}_{\mathcal{P}}$ is equivalent to simulating an AuxPDA that uses $k\log(n)$ bits of memory.

- The reduction is essentially the same.
- We build k DFA's and one PDA to collectively verify an AuxPDA "computation".
- The auxiliary work tape is split up into k regions.
- We assign one DFA to each region.
- The single PDA is used to keep track of the stack.

Auxiliary Space and Deterministic Time

- Deterministic Polynomial Time is equivalent to Auxiliary Logspace [Cook 71].
  - $\mathrm{P} = \mathrm{AuxL}$.
- A more careful look reveals that n^k-time bounded DTM's are essentially equivalent to $k \log(n)$-space bounded AuxPDA's.
- Therefore, solving k-$\mathrm{IE}_{\mathcal{P}}$ is equivalent to simulating a DTM that runs for at most n^k time.

Complexity Lower Bounds

- Solving $k\text{-IE}_{\mathcal{D}}$ is equivalent to simulating a NTM that uses $k\log(n)$ bits of memory.
  - $\exists c_1\ \forall k\ \ k\text{-IE}_{\mathcal{D}} \notin \mathrm{NSPACE}(c_1 k \log(n))$
- Solving $k\text{-IE}_{\mathcal{P}}$ is equivalent to simulating a DTM that runs for at most n^k time.
  - $\exists c_2\ \forall k\ \ k\text{-IE}_{\mathcal{P}} \notin \mathrm{DTIME}(n^{c_2 k})$

Overview

Restricted Classes of Finite Automata

- Let's make the problem easier by looking at restricted classes of DFA's.
- We only want classes of DFA's that are closed under the product construction.
- Consider the following restriction examples:
  - Graph Structure: DFA's with an acyclic state diagram.
  - Algebraic Structure: DFA's with a commutative transition monoid.

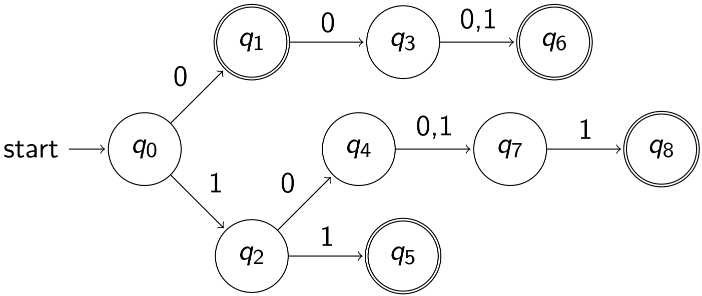
Tree Shaped Finite Automata

Definition

A Tree Shaped DFA is a DFA whose state diagram forms a tree (ignoring the dead state).

- Tree shaped DFA's have a root, a height, and they only accept finite languages.
- Tree shaped DFA's are closed under products.
- We focus on tree shaped DFA's over a binary input alphabet.

Tree Shaped DFA: Example



- Notice that the DFA above accepts the finite language: $\{0, 11, 000, 001, 1001, 1011\}$.
- We can simply represent this language by: $\{0, 11, 00*, 10*1\}$.

Overview

Hardness with 3 Final States

- We denote the intersection non-emptiness problem for tree shaped DFA's by $\mathrm{IE}_{\mathcal{TD}}$.

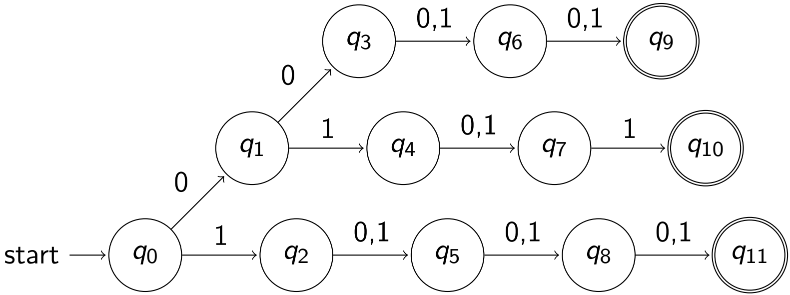
Theorem

$\mathrm{IE}_{\mathcal{TD}}$ is NP-complete even if each DFA has only 3 final states.

- A witness is any string that is in the intersection.
- The intersection problem is in NP because the witness length is linear in the number of states.
- Hardness follows by a reduction from 3-SAT.
- For a given 3-SAT formula, we can construction a tree shaped DFA for each clause.

Clause: Example

- Consider a clause $(v_1 \vee \neg v_2 \vee v_4)$ from a 3-SAT instance.
- The following DFA is associated with the clause:



Hardness for 2 Tree Shaped DFA's

Theorem

If $2\text{-IE}_{\mathcal{TD}}$ is solvable in $O(n^{2-\varepsilon})$ time, then CNF-SAT is solvable in $O(\text{poly}(n) \cdot 2^{(1-\frac{\varepsilon}{2})n})$ time.

- We will define a special kind of reduction from CNF-SAT to Intersection Non-Emptiness.
- Let a CNF formula ϕ with n variables and m clauses be given.
- We construct Tree Shaped DFA's D_1 and D_2 such that ϕ is satisfiable if and only if $L(D_1) \cap L(D_2) \neq \emptyset$.
- Each DFA has $O((m+n) \cdot 2^{\frac{n}{2}})$ states.

Hardness for 2 Tree Shaped DFA's

Theorem

If 2-$\mathrm{IE}_{\mathcal{TD}}$ is solvable in $O(n^{2-\varepsilon})$ time, then CNF-SAT is solvable in $O(\text{poly}(n) \cdot 2^{(1-\frac{\varepsilon}{2})n})$ time.

- A variable assignment for ϕ is a bit string of length n.
- This string can be broken up into two blocks of $\frac{n}{2}$ bits each.
- Each clause c_i is assigned a clause bit b_i.
- A clause bit b_i is valid if either:
  - $b_i = 0$ and block 1 forces c_i to be satisfied
  - $b_i = 1$ and block 2 forces c_i to be satisfied.

Hardness for 2 Tree Shaped DFA's

Theorem

If 2-$\mathrm{IE}_{\mathcal{TD}}$ is solvable in $O(n^{2-\varepsilon})$ time, then CNF-SAT is solvable in $O(\mathrm{poly}(n) \cdot 2^{(1-\frac{\varepsilon}{2})n})$ time.

- The DFA's read in block 1 and block 2 of a variable assignment followed by a string of m clause bits.
- D_1 branches for block 1 and D_2 branches for block 2.
- D_1 verifies that for each i, if $b_i = 0$, then block 1 satisfies c_i.
- D_2 verifies that for each i, if $b_i = 1$, then block 2 satisfies c_i.
- Together, D_1 and D_2 verify that each clause bit is valid and hence each clause is satisfied by some block.

Reduction to k-Clique

Theorem

k-$\mathrm{IE}_{\mathcal{TD}}$ is efficiently reducible to k-Clique.

- Let k Tree Shaped DFA's with n states each be given.
- Form a graph G with $O(n \cdot k)$ vertices such that each tree branch denotes a vertex.
- There is an edge connecting branches b_i and b_j if:
  - b_i and b_j come from different DFA's
  - b_i and b_j have no bit mismatches.
- A k-clique in G represents a valid choice of k branches where there are no mismatches.

Reduction to k-Hyperclique

- We further denote the intersection problem for k tree shaped DFA's over an alphabet of size c by $(c, k)\text{-IE}_{\mathcal{TD}}$.

Theorem

$(c, k)\text{-IE}_{\mathcal{TD}}$ is reducible to c-uniform k-Hyperclique.

- We construct a c-uniform hypergraph H.
- The vertices of H are similarly tree branches.
- A group of c-many branches forms a hyperedge if:
  - no two branches come from the same DFA
  - at each character position, the c-ary intersection of possible characters from each branch is non-empty.
- A k-hyperclique in H represents a valid choice of k branches where there are no c-ary mismatches.

Summary of Complexity Results

- The intersection problem is NP-complete even when each Tree Shaped DFA's has at most 3 final states.
- If we can solve $2\text{-}\mathrm{IE}_{\mathcal{TD}}$ in less than quadratic time, then we get faster algorithms for SAT.
- There is an efficient reduction to k-Clique.
- Using a known approach for k-Clique, we can solve the intersection problem in $O(n^{0.792k})$ time for $k \geq 3$.
- Further, there is a connection between larger alphabets and higher dimensional graphs.

Overview

Intersection Non-Emptiness Problems

- We discussed intersection non-emptiness for the following:
  - DFA's
  - DFA's and one PDA
  - Tree Shaped DFA's
- We could additionally consider:
  - Acyclic DFA's
  - Symmetric Finite Automata
  - Unary Finite Automata
- We could also consider non-string based automata such as tree automata.