Comp409
Lecture 11

February 20, 2003

1 Splitting Method

Recall from our last time the splitting method. It is easy to prove the correctness of the algorithm.

Correctness criterion: $SAT(\phi) = 1$ iff ϕ is satisfiable. If $\phi \in \{0, 1\}$ then $SAT(\phi) = \phi$. Therefore, the algorithm is correct in this case. Else, by induction it is easy to see why the algorithm would still be correct.

It is important to note that the correctness criterion also means that $SAT(\phi)$ returns 0 if ϕ is not satisfiable.

Let's observe the time and space complexity of the algorithm.

> Time Complexity: $2^{|AP(\phi)|}$ – exponential
>
> Space Complexity: nr.of recursive calls $\cdot$ amount of memory per recursive call $= O(|AP(\phi)| \cdot |\phi|)$ – polynomial (note that $|\phi[p \rightarrowtail c]| \leq |\phi|$).

We could trade between time and space requirements by keeping remembering nodes visited on the search tree. However, this may push the space complexity from polynomial to exponential. Therefore, the question of what we should remember and what we can forget becomes important.

2 Data Representation

Consider the problem of multiplying two polynomials $(a_n x^n + a_{n-1}x^{n-1} + \cdots + a_0)(b_n x^n + b_{n-1}x^{n-1} + \cdots + b_0)$. The "traditional" method (i.e., multiplying the corresponding coefficients for each index) takes $O(n^2)$ time. However, we solve the problem in this way just because we are confined to this particular representation. It turns out that there is an algorithm, using Fast Fourier Transform (FFT) to transform the polynomials to another representation, takes $O(n \log n)$ time. Pushing it even further, assume that the polynomials are represented in the roots form. That is, assume $(a_n x^n + a_{n-1}x^{n-1} + \cdots + a_0) = (x - z_1^1)(x - z_2^1)\cdots(x - z_n^1)$ and $(b_n x^n + b_{n-1}x^{n-1} + \cdots + b_0) = (x - z_1^2)(x - z_2^2)\cdots(x - z_n^2)$. Then finding the product becomes trivial (by simply concatenating them) and only takes linear time.

Due to this observation, we now go back to the satisfiability problem to see if we can find alternate representations with more efficient algorithms. We could represent the set of truth assignments for a formula ϕ by $models(\phi)$.

Semantic Space: The discussion by far is on the syntactic space, which is defined by $\varphi \in \mathit{Form}$. We also have the semantic space defined by $\text{models}(\varphi) \in 2^{2^{\text{Prop}}}$. Then the satisfiability problem is reduced to $\varphi \mapsto \text{models}(\varphi) \stackrel{?}{=} \emptyset$.

Let's see if semantic space is a good representation:

> If $\phi = p$ then $models(p) = 2^{n-1}$ (exponential!)
>
> If $\phi = (\theta \vee \psi)$ then $models(\phi) = models(\theta) \cup models(\psi)$. (polynomial time operation)
>
> If $\phi = (\theta \wedge \psi)$ then $models(\phi) = models(\theta) \cap models(\psi)$. (polynomial time operation)
>
> If $\phi = \neg\theta$ then $models(\phi) = 2^{\text{Prop}} - models(\theta)$. This operation can blow up since if $models(\theta)$ is a small set then its complement will be big. Conclusion: It is not feasible to work with such a representation for truth assignments.

We observe that semantic space has easy SAT test ($\text{models}(\varphi) \stackrel{?}{=} \emptyset$) and hard base case ($\text{models}(p_i) = \{\tau : \tau(p_i) = 1\}$) is exponentially large. For inductive case, union and intersection are also in polynomial time. However, the complement could be a very large set (i.e., slow). So semantic space is not a good representation.

The previous representation did not degrade gracefully. We need a representation that has the following desiderata (desired properties):

1. Easy non-emptiness test.
2. Easy construction.
   - Easy base case.
   - Easy inductive case.
3. We should be able to easily (i.e. polynomially) go back and forth between representations.

2.1 Deterministic Finite Automata

Consider writing all propositions consecutively $p_1 p_2 \cdots p_n$, i.e. we fix the enumeration of propositions. Then, a truth assignment could be written as a string (bit vector) like $00 \cdots 0 \in \{0,1\}^n$, which is a binary word of length n. So a set of truth assignments can also be written as a set of words, which forms a language.

Fact 2.1. *Finite languages are regular.*

Since the number of propositions is finite, the language must also be finite, and so the language must be regular. Therefore, we have $\varphi \mapsto A_\varphi$ (an automata for φ), and $L(A_\varphi) = \text{models}(\varphi)$. We now define a deterministic finite automata (DFA).

Definition 1. *A DFA* $A = (\Sigma, S, s_0, \rho, F)$ *with*

$$\begin{aligned} \Sigma &: \textit{finite alphabet} \\ S &: \textit{finite set of states} \\ s_0 \in S &: \textit{initial state} \\ \rho : S \times \Sigma \to S &: \textit{transition function} \\ F \subseteq S &: \textit{set of accepting states} \end{aligned}$$

Definition 2. *Define* $\overline{\rho} : S \times \Sigma^* \to S$ *inductively.*

- $\overline{\rho}(s, \varepsilon) = s$
- $\overline{\rho}(s, wa) = \rho(\overline{\rho}(s, w), a)$

Definition 3. *A DFA A accepts w if* $\overline{\rho}(s_0, w) \in F$. $L(A) = \{w \mid A \text{ accepts } w\}$.

Definition 4. *The following DFA accepts strings that terminate with a* 1*:*

$$\begin{aligned} \Sigma &\colon \{0,1\} \\ S &\colon \{0,1\} \\ s_0 &\colon \mathit{0} \\ \rho &\colon \ \rho(s,a) = a \\ F &\colon \{1\} \end{aligned}$$

Definition 5. *The following DFA accepts strings of length* n.

$$\begin{aligned} \Sigma &\colon \{0,1\} \\ S &\colon \{0,1,\ldots,n,n+1\} \\ s_0 &\colon \mathit{0} \\ F &\colon \{n\} \end{aligned}$$

where ρ is defined as:

$$\rho(i,a) = \begin{cases} i+1, & i \le n \\ i, & i = n+1 \end{cases}$$

The state $n+1$ is also known as the *sink state.*

Definition 6 (Emptiness Problem). *Given* A, $L(A) \stackrel{?}{=} \emptyset$.

Let's modify the above DFA by defining $\rho(1, a) = 1$. It is obvious that the above DFA does not accept any input. We can check whether a DFA is empty by verifying that there is no path from the start state to an accepting state.

Definition 7. *Let $A = (\Sigma, S, s_0, \rho, F)$ be a DFA. The graph $G_A = (S, E_A)$ has*

$$E_A = \{(s,t) : t = \rho(s,a) \text{ for some } a \in \Sigma\} \ .$$

Lemma 1. *$L(A) \neq \emptyset$ iff there is a path from s_0 to F in G_A.*

Proof. We prove the "if" and "only if" parts separately.

- Suppose $L(A) \neq \emptyset$. Then there is some $a_1 a_2 \cdots a_n \in L(A)$ and an input $s_0 s_1 \cdots s_n$ such that $s_{i+1} = \rho(s_i, a_{i+1})$ for $0 \leq i \leq n-1$ and $s_n \in F$. By definition of G_A there is a path from s_0 to s_n in G_A.
- Suppose $s_0 s_1 \cdots s_n$ is a path in G_A from s_0 to $s_n \in F$. Since there is an edge in G_A from s_i to s_{i+1} for $0 \leq i \leq n$, there is a letter $a_i \in \Sigma$ such that $s_{i+1} = \rho(s_i, a_i)$ for $0 \leq i \leq n$. Then $s_0 s_1 \cdots s_{n-1} \in L(A)$.

□

By Lemma 1, an algorithm to verify emptiness can eimply be a breadth-first search on the edges of the graph to see if there is a path from s_0 to the final state. Thus, the problem is reduced to a simple graph-reachability problem.

Corollary 2. *DFA emptiness problem is solvable in linear time ($O(|S| + |E_A|)$).*

2.2 Representation using DFA

2.3 Base Case

This is the case when there is only one atomic proposition, i.e., $\varphi = p_i$. We can construct the automata A_{p_i} as follows:

Definition 8. *The following DFA accepts strings of length n that have 1 in bit i.*

$$\begin{aligned} \Sigma &: \{0,1\} \\ S &: \{0,1,\ldots,n,n+1\} \\ s_0 &: \mathit{0} \\ F &: \{n\} \end{aligned}$$

where ρ is defined as:

$$\rho(j,a) = \begin{cases} j+1, & j \le n \text{ and if } j = i \text{ then } a = 1 \\ j, & j = n+1 \end{cases}$$

Therefore, the base case is easy.

2.3.1 Conjunction

Consider formulas φ_1 and φ_2 represented by automata A_1 and A_2, respectively. Assume $A_1 = (\Sigma, S^1, s_0^1, \rho^1, F^1)$ and $A_2 = (\Sigma, S^2, s_0^2, \rho^2, F^2)$. We would like to have $L(A_1 \cap A_2) = L(A_1) \cap L(A_2)$. Let $A_1 \cap A_2 = (\Sigma, S^1 \times S^2, (s_0^1, s_0^2), \rho, F^1 \times F^2)$, and let $\rho((s,t),a) = (\rho^1(s,a), \rho^2(t,a))$. The time complexity is quadratic.

2.3.2 Disjunction

We would like to have $L(A_1 \cup A_2) = L(A_1) \cup L(A_2)$. Let $A_1 \cup A_2 = (\Sigma, S^1 \times S^2, (s_0^1, s_0^2), \rho, (F^1 \times S^2) \cup (S^1 \times F^2))$. The time complexity is again quadratic.

2.3.3 Complement

The complement of $L(A)$, $L(\overline{A}) = \Sigma^* - L(A)$. Let $\overline{A} = (\Sigma, S, s_0, \rho, S - F)$.

Thus, we have established a good representation that has easy SAT test, base case, and inductive case. In the next lecture, we will see whether it is useful in solving the satisfiability problem.