# COMP 409, Lecture No. 7

February 7, 2002

## 1 Review

In the previous class, we talked about **constraint satisfaction**, which represents the majority of problems to which students are exposed during their elementary and secondary schooling. It is a generic problem-solving framework, particularly in continuous mathematics. When you move to the discrete domain, things become somewhat more difficult.

## 2 Satisfiability as a generic problem-solving framework

Satisfiability is an example of a constraint satisfaction problem. So is the problem of **graph coloring**:

*Given a graph $G = (V, E)$ where $E \subseteq V^2$ and a number $k > 1$, does there exist a mapping function $c : V \to \{0, 1, \ldots, k-1\}$ that produces no monochromatic edges? That is, given $(u, v) \in E$, $c(u) \neq c(v)$.*

2-coloring a graph is quite simple; we will look at 3-coloring a graph and we will express the problem in terms of a boolean formula.

- We begin by defining our atomic propositions $\textsc{Prop} = \{p_{v,r}, p_{v,g}, p_{v,b}\} \forall v \in V$, where $p_{v,k} = \mathit{true}$ when $c(v) = k$.
- Next, we express the constraint that every node is assigned a unique color:

$$\bigwedge_{v \in V} (p_{v,r} \wedge \neg p_{v,g} \wedge \neg p_{v,b}) \vee (\neg p_{v,r} \wedge p_{v,g} \wedge \neg p_{v,b}) \vee (\neg p_{v,r} \wedge \neg p_{v,g} \wedge p_{v,b})$$

- Finally, we ensure that there are no monochromatic edges:

$$\bigwedge_{(u,v) \in E} \neg(p_{u,r} \wedge p_{v,r}) \wedge \neg(p_{u,g} \wedge p_{v,g}) \wedge \neg(p_{u,b} \wedge p_{v,b})$$

We now need to show that with our expression of the problem in this formula $\varphi_G$, finding a truth assignment for $\varphi_G$ is equivalent to finding a coloring for the graph $G$.

**Lemma 1** *$G$ is 3-colorable iff $\varphi_G$ is satisfiable.*

We want to show a one-to-one relation between $models(\varphi_G)$ and $colorings(G)$.

Given $C \in colorings(G)$, $\tau_C(p_{v,k}) = 1 \leftrightarrow color(v) = k$. We can easily show that $\tau_C \models \varphi_G$.

Going the other way, assume $\tau \models \varphi_G$. Define $c : V \to \{r, g, b\}$ where $c(v) = k \leftrightarrow \tau(p_{v,k}) = 1$. The first part of $\varphi_G$ assures us that $c$ is actually a function, whereas the second part of $\varphi_G$ tells us there are no monochromatic edges.

# 3 Further applications of satisfiability

We can also use formulas to express logic puzzles in terms of satisfiability. For example, the Wolf, Goat and Cabbage puzzle: you need to get a wolf, goat, and a cabbage across a river in a small rowboat. Said boat is only large enough to carry you and one other item at a time. In addition, you cannot leave the wolf and goat on the same shore alone (the wolf will devour the goat) or the goat and the cabbage on the same shore (goats love to eat cabbage). Devise a schedule of trips to move the animals and vegetable across the river without losing any of them. In particular, can you find a schedule for no more than $2k + 1$, $k \in N$, trips?

We will not show the entire formula here, but we will show the atomic propositions and how certain conditions may be expressed in terms of those propositions.

Our set of propositions is defined in two parts. The first part indicates which object is being transported across the river during which trip:

$$\{p_{o,i} : o \in \{W, G, C\}, 0 \leq i \leq 2k + 1\}$$

Set two indicates which side the objects are on after each trip. We define the starting side as 0 and the opposite side as 1.

$$\{q_{o,i,s} : o \in \{W, G, C\}, 0 \leq i \leq 2k + 1, s \in \{0, 1\}\}$$

With these propositions, we can express the constraint that the wolf and goat cannot be on the same side (similarly for the cabbage and the goat):

$$\bigwedge_{0 \leq i \leq 2k+1, s \in \{0,1\}} \neg(q_{W,i,s} \wedge q_{G,i,s})$$

An object cannot be on both sides at the same time:

$$\bigwedge_{0 \leq i \leq 2k+1, o \in \{W,G,C\}} \neg(q_{o,i,0} \wedge q_{o,i,1})$$

Similarly, an object cannot be on the boat during a given round and on the shore:

$$\bigwedge_{0\le i\le 2k+1, o\in\{W,G,C\}} \neg(q_{o,i,0} \wedge p_{o,i}) \wedge \neg(q_{o,i,1} \wedge p_{o,i})$$

The remaining parts of the formula deal with actually moving the objects across (changing the state of the objects) and the initial conditions.

# 4 Solving satisfiability

It is all well and good to talk about expressing problems in terms of (potentially) satisfiable formulas, but how do we set about finding out if those formulas are actually satisfiable or not? There exists a very simple algorithm to do so:

```
SAT(φ)
      B := False
      for all τ ∈ 2^{AP(φ)}
         B = B ∨ (τ ⊨ φ)
                                   end
      return B
```

This sort of algorithm is called an "Exhaustive Search" or "Brute-force Search" algorithm, since it looks through every possible element in the search space, and tries it to see if it fulfills the requirement. In this case, the search space is $2^{AP(\varphi)}$. So, traversing all of its elements takes exponential time, and therefore Sat($\varphi$) takes exponential time. On the other hand, given $\tau \in 2^{AP(\varphi)}$, checking whether $\tau \models \varphi$ takes only polynomial time.

# 5 Complexity Theory

There are many problems that have similar properties as Sat($\varphi$), i.e. having very large search space, and having lack of mathematical structure. In most of the cases, it is very hard to find out a solution to the problem, but it is rather easy to check whether a given solution is correct. Such problems are called to be in **NP** class, and NP-completeness theory deals with this class of problems.

## 5.1 NP class

**Definition 1** *Given a finite alphabet* $\Sigma$*, a* Decision Problem *is some* $L \subseteq \Sigma^*$. *It is* nontrivial *if* $L \neq \emptyset$ *and* $L \neq \Sigma^*$.

**Notation**: $\Sigma^{\le k} = \cup_{i=1}^{k} \Sigma^i$.

**Definition 2** *A problem* $L \subseteq \Sigma^*$ *is in NP if there is a polynomial* $p$ *and a polynomial time algorithm* $A$ *such that* $x \in L$ *iff there exists* $y \in \Sigma^{\le p(|x|)}$ *where* $A(x,\ y) = 1$.

In other words a problem $L$ is in NP if when we "magically" guess a "short witness" $y$ , a polynomial time algorithm exist to "check" the witness.

Here $y$ is called the *witness* for $x$, since it witnesses the membership of $x$ in $L$. For the case of SAT, the witness $y$ is $\tau$, and the algorithm $A$ is Eval. A problem is in NP there are always short witnesses and it is possible to check witnesses in polynomial time. Alternatively, we can say that a problem in NP can be solved in polynomial time by a *nondeterministic* algorithm: an algorithms that first *guesses* a witness and then checks it.

## 5.2 Other Examples of problems

1. **Diophantine Equation:** *Does* $P(x_1, x_2) = 0$ *have integer solutions?* The solution is undecidable. Hilbert's 10th problem asked if a technique for solving a general Diophantine existed. The impossibility of obtaining a general solution was proven by Matyasevich in the early 1970's. However, verifying the correctness of a particular solution $(x'_1, x'_2)$ takes only polynomial time. This problem is not in NP, because short witnesses may not exist.

2. **Compositeness Problem** : *Given a number* $n \in N$*, is* $n$ *composite ?* Given the factors $n_1$ and $n_2$, it is easy to verify if $n = n_1 * n_2$. However, the time taken to find suitable $n_1$ and $n_2$ is exponential of the length of the binary representation of $n$. This problem is in NP.

3. **Eulerian Cycle** on graph $G(V, E)$: *Is there a bijective mapping* $P : E \to \{1, \ldots, |E|\}$ *such that if* $P(e_2) = (P(e_1) + 1) \bmod |E|$ *then there exist* $u, v, w \in V$ *such that* $e_1 = (u, v)$ *and* $e_2 = (v, w)$*?* An Eulerian Cycle on a graph $G$ is a cycle that visits each edge exactly once. An Eulerian Cycle exists iff the degree of each node in the graph is even. Thus, this problem is not only in NP, it is in fact in PTIME.

4. **Hamiltonian Cycle** on graph $G(V, E)$: *Is there a bijective mapping* $P : V \to \{1, \ldots, |v|\}$ *such that if* $P(v) = (P(u) + 1) \bmod |V|$*, then* $(u, v) \in E$*?* A Hamiltonian cycle on a graph $G$ is a cycle that visits each node exactly once and terminates at the initial node. This problem is in NP.

5. **K-Coloring Problem** : Consider a graph representation of a map. $G = (V, E)$, whose nodes are the partitions in the map and neighboring partitions (nodes) are connected by edges. A *k-coloring* is a mapping $C : V \mapsto \{1..k\}$, obeying the constraint $C(u) = C(v) \to (u, v) \notin E$. We say $G$ is $k$-colorable if it has a $k$-coloring. *Graph k-colorability* is the problem if checking whether a given graph is $k$-colorable. Graph $k$-colorability is also in NP. The witness is the $k$-coloring $C$.