SNEAKY COMPLETE LANGUAGES

JOEL FRIEDMAN

Contents

Disclaimer: The material may sketchy and/or contain errors, which I will elaborate upon and/or correct in class. For those not in CPSC 421/501: use this material at your own risk...

There is a standard way to produce an languages that are complete for NP and PSPACE (under polynomial time reductions). Let us start with the NP-complete language.

Let

$$\text{NP-SNEAKY} = \{\langle M, w, 1^t\rangle \mid M \text{ is a non-deterministic T.m. that accepts } w \text{ within time } t\}.$$

We claim that NP-SNEAKY is NP-complete. To prove this we need to show that (1) NP-SNEAKY lies in NP, and (2) any $L \in \text{NP}$ can be reduced in polynomial time to NP-SNEAKY. Claim (2) is almost immediate, and claim (1) requires a bit more thought: you run a (non-deterministic) universal Turing machine for t steps of M on input w, and you have to verify that the simulation runs in time polynomial of

$$\langle M\rangle + \langle w\rangle + t.$$

This is easy (since the input size is at least t), and was done in class. You should be aware that the simulation will *not* run in time in in time polynomial of

$$\langle M\rangle + \langle w\rangle + \log_2 t.$$

For this reason the language

$$\text{NP-FAIL} = \{\langle M, w, t\rangle \mid M \text{ is a non-deterministic T.m. that accepts } w \text{ within time } t\}$$

will fail to be in NP, when you describe t in base 10 or binary, as one is accustomed to doing.

You might compare this to showing that SAT is NP-complete: showing that SAT is in NP is easy, but showing that any language in NP can be reduced to SAT is the essence of the Cook-Levin theorem, and is much more elaborate. For NP-SNEAKY both steps in showing NP-completeness are easy, but the first step—which requires a universal Turning machine—is more difficult than the second.

Research supported in part by an NSERC grant.

Another comparison between NP-SNEAKY and SAT (and 3COLOR, VERTEX-EXPANSION, PARTITION, etc.) is that the latter problems are interesting in applications, whereas NP-SNEAKY is just a formal construction that doesn't seem to have applications beyond giving a language with a simple proof of NP-completeness.

Similar remarks hold for the language:

$$\text{PSPACE-SNEAKY} = \{\langle M, w, 1^s\rangle \mid M \text{ is a non-deterministic T.m. that accepts } w \text{ using at most space } s\},$$

which we easily show is complete for PSPACE under polynomial time reductions, i.e., (1) PSPACE-SNEAKY lies in PSPACE, and (2) if L lies in PSPACE, then there is a polynomial time reduction of L to PSPACE-SNEAKY.

Department of Computer Science, University of British Columbia, Vancouver, BC V6T 1Z4, CANADA.

E-mail address: jf@cs.ubc.ca

URL: http://www.cs.ubc.ca/~jf