Compsci 501 Lecture 37: Ladner’s Theorem

**Theorem 37.1** ([Ladner, 1975])

*If $P \neq NP$ then there exists an intermediate problem $I \in NP - P$ that is not NP complete.*


**Proof:** Assume that $P \neq NP$. We will construct $I$ by a method called “delayed diagonalization”.

The construction will make sure that:

- **$I$ is not hard:** $SAT \not\leq I$. \(R_1, R_3, R_5, \ldots\)
- **$I$ is not easy:** $I \not\in P$. \(R_2, R_4, R_6, \ldots\)

\(R_{2k+1} : “M_k \text{ isn’t a DSPACE}[k \log n] \text{ reduction from SAT to } I”\)

\(R_{2k+2} : “M_k \text{ isn’t a DTIME}[kn^k] \text{ recognizer of } I”\)

**Observation:** If all the $R_i$’s are met, then we’re done.
Conditions to Satisfy: $R_i, \ i = 1, \ldots \infty$

$R_{2k+1}$: “$M_k$ isn’t a DSPACE$[k \log n]$ reduction from SAT to $I$”

$R_{2k+2}$: “$M_k$ isn’t a DTIME$[kn^k]$ recognizer of $I$”

On input $w$, recursively $I(w)$ does following:

1. **do** for a total of $|w|$ steps {
2. **for** $i = 1 \ldots \infty$ **do** {
3. **for** $x = 1 \ldots \infty$ **do** {
4. if ($R_i$ verified on input $x$) **then** next $i$
5. } }
6. if ($i$ is even and $w \in$ SAT) **then** ACCEPT
7. **else** REJECT

**Note:** In line 4, $I$ simulates itself **deterministically**. Thus, to check if an input is in SAT it might need exponential time. Thus, it might only find out exponentially later that condition $R_i$ has been met. That’s why this method is called **delayed diagonalization**. The key idea, is that if $i$ is even we are simulating SAT, so if we do this long enough we cannot be in P, whereas if $i$ is odd then we are rejecting all inputs, so if we do this long enough we cannot be NP complete.