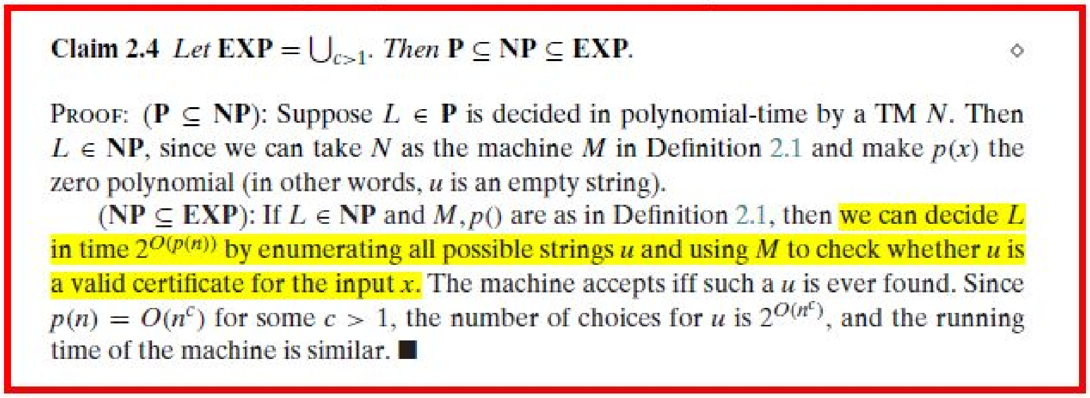
Algumas coisas não vistas no encontro 2:

Inclusões básicas de classes:

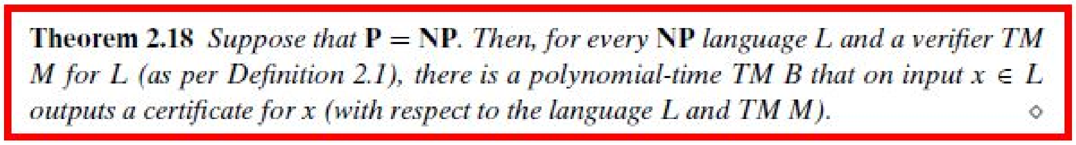


Claim 2.4 *Let* **EXP** $= \bigcup_{c>1}$. *Then* **P** $\subseteq$ **NP** $\subseteq$ **EXP**. ◇

Proof: (**P** $\subseteq$ **NP**): Suppose $L \in$ **P** is decided in polynomial-time by a TM N. Then $L \in$ **NP**, since we can take N as the machine M in Definition 2.1 and make $p(x)$ the zero polynomial (in other words, u is an empty string).

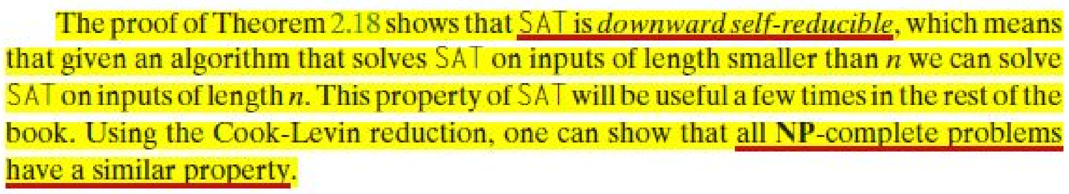
(**NP** $\subseteq$ **EXP**): If $L \in$ **NP** and $M, p()$ are as in Definition 2.1, then we can decide L in time $2^{O(p(n))}$ by enumerating all possible strings u and using M to check whether u is a valid certificate for the input x. The machine accepts iff such a u is ever found. Since $p(n) = O(n^c)$ for some $c > 1$, the number of choices for u is $2^{O(n^c)}$, and the running time of the machine is similar. ■

Se decidir é polinomial, então buscar solução é polinomial



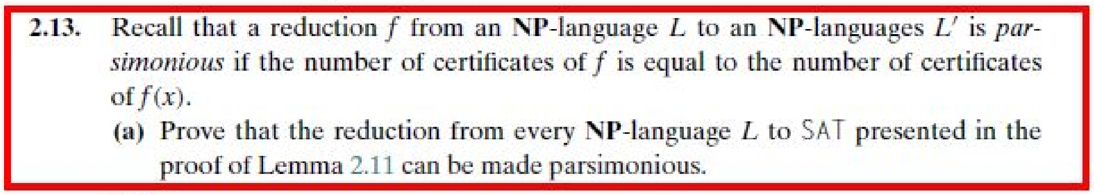
Theorem 2.18 *Suppose that* **P** = **NP**. *Then, for every* **NP** *language L and a verifier TM M for L (as per Definition 2.1), there is a polynomial-time TM B that on input $x \in L$ outputs a certificate for x (with respect to the language L and TM M).* ◇

Depois da prova do teorema, ver esta obversação interessante:



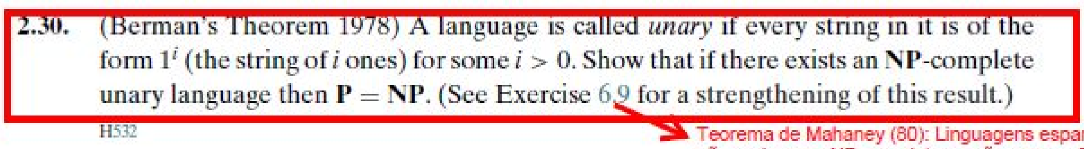
The proof of Theorem 2.18 shows that SAT is *downward self-reducible*, which means that given an algorithm that solves SAT on inputs of length smaller than n we can solve SAT on inputs of length n. This property of SAT will be useful a few times in the rest of the book. Using the Cook-Levin reduction, one can show that all **NP**-complete problems have a similar property.

Exercícios interessantes:



2.13. Recall that a reduction f from an **NP**-language L to an **NP**-languages L' is *parsimonious* if the number of certificates of f is equal to the number of certificates of $f(x)$.

(a) Prove that the reduction from every **NP**-language L to SAT presented in the proof of Lemma 2.11 can be made parsimonious.



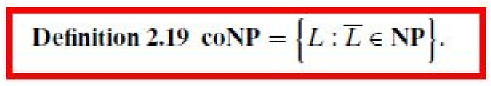
2.30. (Berman's Theorem 1978) A language is called *unary* if every string in it is of the form 1^i (the string of i ones) for some $i > 0$. Show that if there exists an **NP**-complete unary language then **P** = **NP**. (See Exercise 6.9 for a strengthening of this result.)

H532

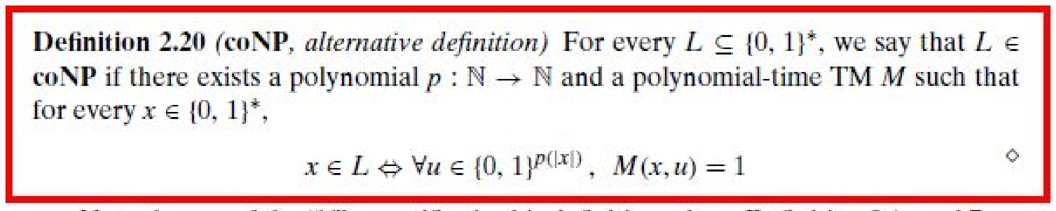
Teorema de Mahaney (80): Linguagens esparsas não podem ser NP-completas a não ser que P=NP

Assuntos do encontro 3:

Duas definições de coNP:



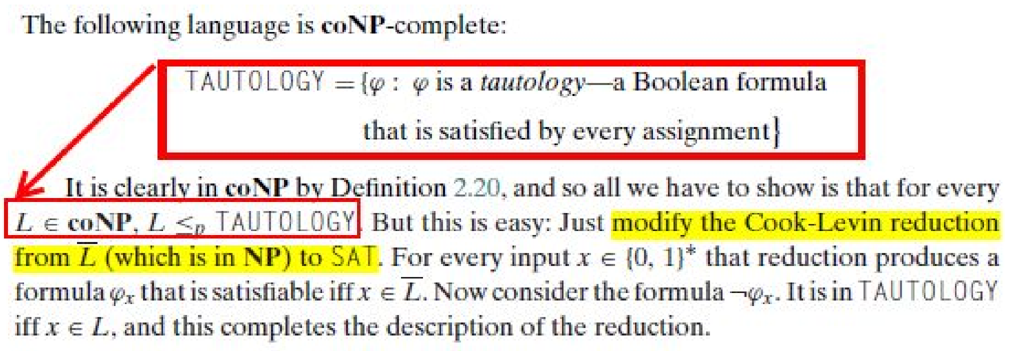
Definition 2.19 coNP $= \left\{ L : \overline{L} \in \mathbf{NP} \right\}$.



Definition 2.20 *(coNP, alternative definition)* For every $L \subseteq \{0,1\}^*$, we say that $L \in$ **coNP** if there exists a polynomial $p : \mathbb{N} \to \mathbb{N}$ and a polynomial-time TM M such that for every $x \in \{0,1\}^*$,

$$x \in L \Leftrightarrow \forall u \in \{0,1\}^{p(|x|)},\ \ M(x,u) = 1$$ ◇

Problema coNP-completo

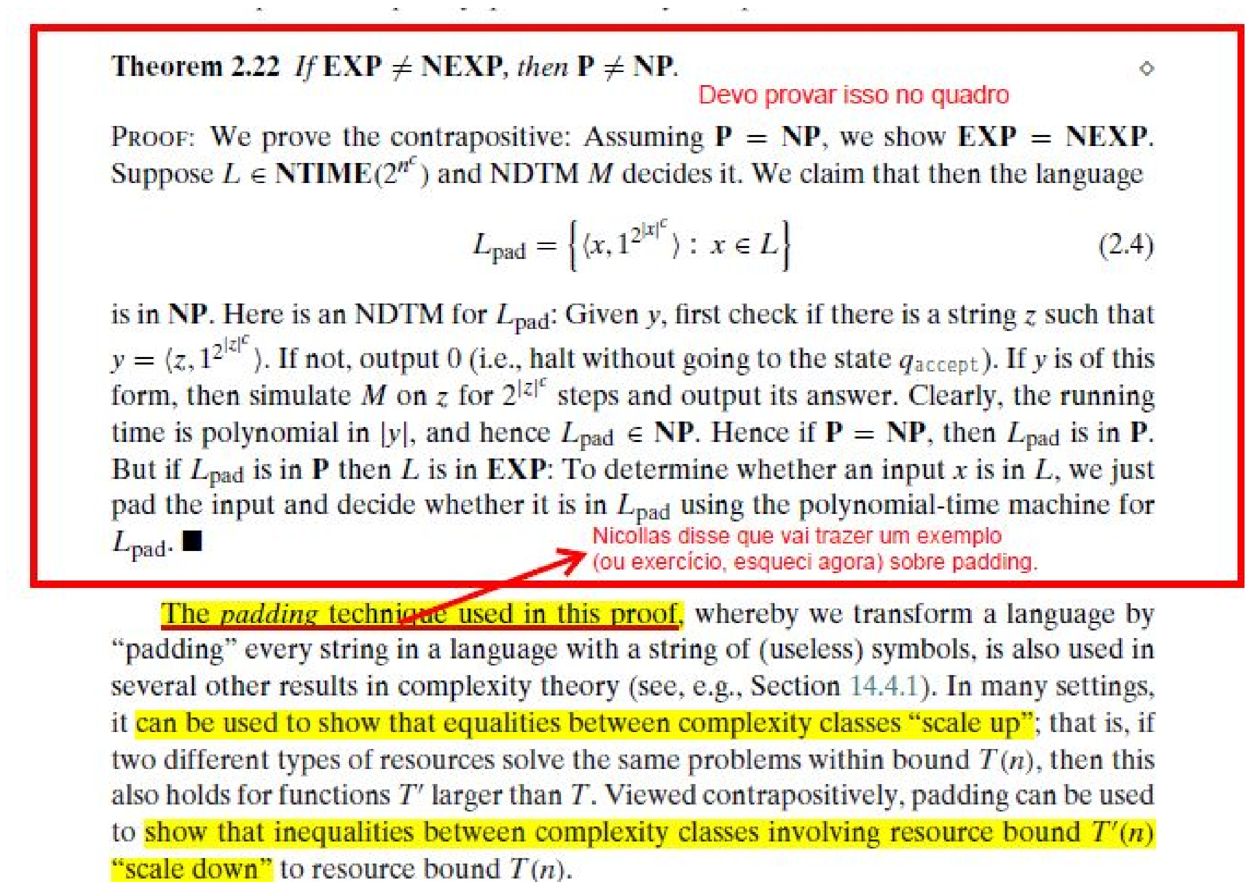


The following language is **coNP**-complete:

TAUTOLOGY $= \{\varphi :\ \varphi$ is a *tautology*—a Boolean formula that is satisfied by every assignment$\}$

It is clearly in **coNP** by Definition 2.20, and so all we have to show is that for every $L \in$ **coNP**, $L \leq_p$ TAUTOLOGY. But this is easy: Just modify the Cook-Levin reduction from $\overline{L}$ (which is in **NP**) to SAT. For every input $x \in \{0,1\}^*$ that reduction produces a formula φ_x that is satisfiable iff $x \in \overline{L}$. Now consider the formula $\neg\varphi_x$. It is in TAUTOLOGY iff $x \in L$, and this completes the description of the reduction.

O Argumento do preenchimento:



Theorem 2.22 *If* **EXP** $\neq$ **NEXP**, *then* **P** $\neq$ **NP**. ◇

Devo provar isso no quadro

Proof: We prove the contrapositive: Assuming **P** $=$ **NP**, we show **EXP** $=$ **NEXP**. Suppose $L \in$ **NTIME**(2^{n^c}) and NDTM M decides it. We claim that then the language

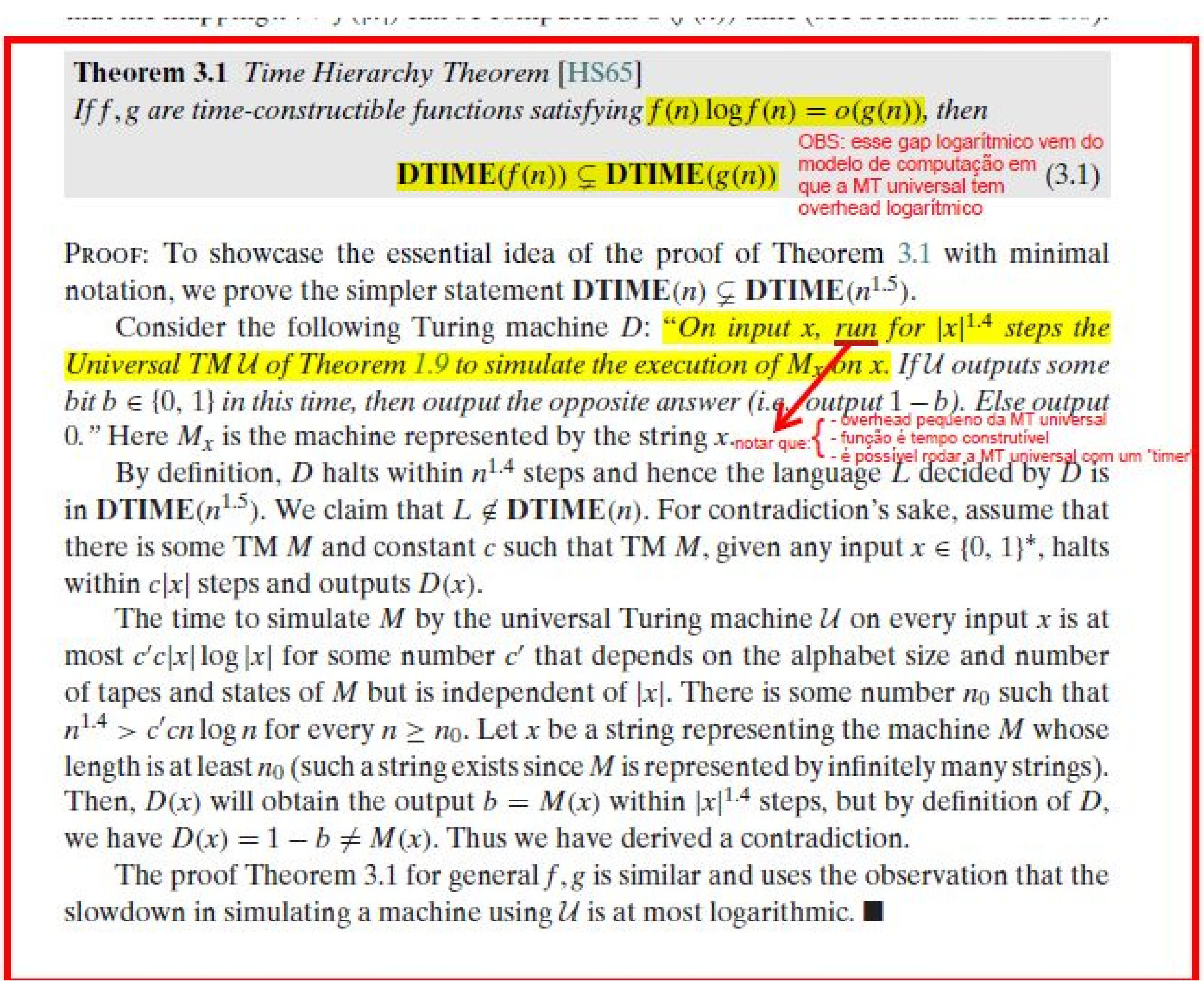
$$L_{\text{pad}} = \left\{ \langle x, 1^{2^{|x|^c}} \rangle : x \in L \right\} \tag{2.4}$$

is in **NP**. Here is an NDTM for L_{pad}: Given y, first check if there is a string z such that $y = \langle z, 1^{2^{|z|^c}} \rangle$. If not, output 0 (i.e., halt without going to the state $q_{\texttt{accept}}$). If y is of this form, then simulate M on z for $2^{|z|^c}$ steps and output its answer. Clearly, the running time is polynomial in $|y|$, and hence $L_{\text{pad}} \in$ **NP**. Hence if **P** $=$ **NP**, then L_{pad} is in **P**. But if L_{pad} is in **P** then L is in **EXP**: To determine whether an input x is in L, we just pad the input and decide whether it is in L_{pad} using the polynomial-time machine for L_{pad}. ■

Nicollas disse que vai trazer um exemplo
(ou exercício, esqueci agora) sobre padding.

The *padding* technique used in this proof, whereby we transform a language by "padding" every string in a language with a string of (useless) symbols, is also used in several other results in complexity theory (see, e.g., Section 14.4.1). In many settings, it can be used to show that equalities between complexity classes "scale up"; that is, if two different types of resources solve the same problems within bound $T(n)$, then this also holds for functions T' larger than T. Viewed contrapositively, padding can be used to show that inequalities between complexity classes involving resource bound $T'(n)$ "scale down" to resource bound $T(n)$.

Teorema da Hierarquia de Tempo:



Theorem 3.1 *Time Hierarchy Theorem* [HS65]
If f, g are time-constructible functions satisfying $f(n) \log f(n) = o(g(n))$, then

$$\mathbf{DTIME}(f(n)) \subsetneq \mathbf{DTIME}(g(n)) \qquad (3.1)$$

OBS: esse gap logarítmico vem do modelo de computação em que a MT universal tem overhead logarítmico

PROOF: To showcase the essential idea of the proof of Theorem 3.1 with minimal notation, we prove the simpler statement $\mathbf{DTIME}(n) \subsetneq \mathbf{DTIME}(n^{1.5})$.

Consider the following Turing machine D: "*On input x, run for $|x|^{1.4}$ steps the Universal TM $\mathcal{U}$ of Theorem 1.9 to simulate the execution of M_x on x. If $\mathcal{U}$ outputs some bit $b \in \{0, 1\}$ in this time, then output the opposite answer (i.e., output $1 - b$). Else output* 0." Here M_x is the machine represented by the string x.

notar que:
- overhead pequeno da MT universal
- função é tempo construtível
- é possível rodar a MT universal com um "timer"

By definition, D halts within $n^{1.4}$ steps and hence the language L decided by D is in $\mathbf{DTIME}(n^{1.5})$. We claim that $L \notin \mathbf{DTIME}(n)$. For contradiction's sake, assume that there is some TM M and constant c such that TM M, given any input $x \in \{0, 1\}^*$, halts within $c|x|$ steps and outputs $D(x)$.

The time to simulate M by the universal Turing machine $\mathcal{U}$ on every input x is at most $c'c|x| \log |x|$ for some number c' that depends on the alphabet size and number of tapes and states of M but is independent of $|x|$. There is some number n_0 such that $n^{1.4} > c'cn \log n$ for every $n \geq n_0$. Let x be a string representing the machine M whose length is at least n_0 (such a string exists since M is represented by infinitely many strings). Then, $D(x)$ will obtain the output $b = M(x)$ within $|x|^{1.4}$ steps, but by definition of D, we have $D(x) = 1 - b \neq M(x)$. Thus we have derived a contradiction.

The proof Theorem 3.1 for general f, g is similar and uses the observation that the slowdown in simulating a machine using $\mathcal{U}$ is at most logarithmic. ■

Uma das conclusões deste teorema: P != EXP.