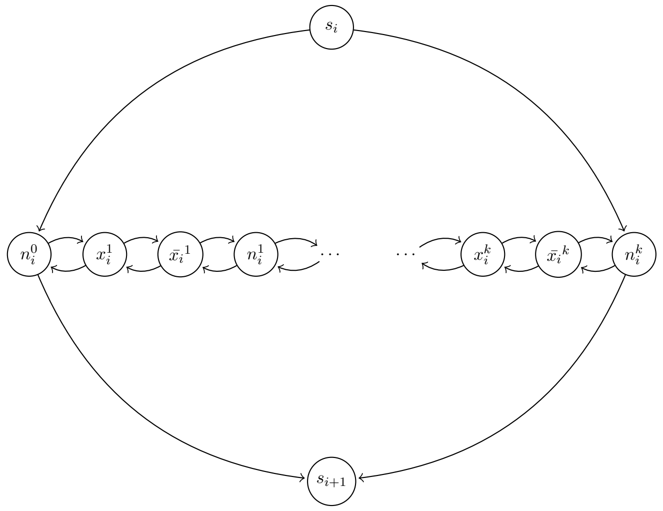
1 Reducing from CNF-SAT to HAMPATH

A path from s to t in a graph G is hamiltonian when the path goes through all nodes exactly once. The $HAMPATH$ problem is defined as:

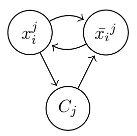
$$HAMPATH = \{(G, s, t) \mid \text{ there exists an hamiltonian path in the oriented graph } G \text{ from } s \text{ to } t\}$$

Question 1. Show that $HAMPATH$ is $\mathcal{NP}$-complete.

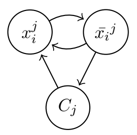
Suggestion: start from a CNF-SAT formula with k clauses and l variables. Build a graph with (for i a variable and j a clause): 3 nodes x_i^j, $\bar{x}_i^j$ et n_j^i; 2 nœuds s_i and n_i^0; 1 node C_j plus one node s_{l+1} using the following gadgets:



(a) Variable gadget



(b) Gadget for x_i used in C_j



(c) Gadget for $\bar{x}_i$ used in C_j

Question 2. Show that the $HAMPATH$ problem stays $\mathcal{NP}$-complete if we remove the orientedness of the graphs considered.

2 Time hierarchy theorem

In this section, we will consider 2-tape deterministic Turing machines. We select a coding $\langle M \rangle$ for the Turing machine M. For instance, $\langle M \rangle$ codes the states with words of $\{0, 1\}^*$, the tape alphabet with words of $\{\alpha, \beta\}^*$, heads direction with $\{<, >, \_\}^2$. A TM is then represented with a sequence of transitions for each state. For the state e, we represent it as $e|a_1, b_1, c_1, d_1, e_1, f_1, \dots a_k, b_k, c_k, d_k, e_k, f_k|$ where the $e, (e_i)_i$ are the states, the $(a_i)_i, (b_i)_i, (c_i)_i, (d_i)_i$ are all words in both tape and the $(f_i)_i$ are the directions. a (resp. b) is the letter read on the first tape (resp. second tape) during the transition and c (resp. d) is the letter written on the first tape (resp. second tape).

Définition 1. *The set $TIME(f(n))$ corresponds to the set of problems for which there exists a Turing machine deciding them in less than $f(n)$ computing steps over inputs of size n.*

In the same manner that we say that sorting can be done in $O(n \times ln(n))$ (and thus omitting to say "when the input has size n", we always write $TIME(f(n))$ but this should be read as $TIME(n \to f(n))$ because the n represents the size of the input.
In the remaining part of the TD, when needed, you can use the linear speedup theorem stating that for all g and all $\epsilon > 0$ **fixed**, we have $TIME(g(n)) \subseteq TIME(n + 2 + \epsilon g(n))$.

Définition 2. *A function f is* time constructible *when there exists a machine M taking 1^n as input and producing $f(n)$ in time $f(n)$.*

The time hierarchy theorem states that when f is a time constructible function then:

$$TIME\left(o\left(\frac{f(n)}{log f(n)}\right)\right) \subsetneq TIME(f(n))$$

We will demonstrate here a lighter version of the theorem: $TIME(f(n)) \subsetneq TIME(f(2n+1)^3)$ *with* $n^3 \leq f(n)$.
For f time constructible, we define $Halt_f = \{\langle M\rangle\#x \mid M \text{ accepts } x \text{ in } f(|x|) \text{ steps}\}$.
Question 3. Justify that $Halt_f \in TIME(f(n)^3)$.

Let us suppose that $Halt_f \in TIME\left(\frac{f(\lfloor n/2\rfloor)}{2}\right)$. Let K be a TM deciding $Halt_f$ in time $f(\lfloor n/2\rfloor)/2$, we can build D_K taking $\langle M\rangle$ as input and running K on $\langle M\rangle\#\langle M\rangle$. D_K accepts when K refuses and refuses when K accepts.
Question 4. Justify that D_K can be built such that $D_K \in TIME(f(n))$.

Question 5. Justify that D_K can not exists using a diagonalization argument.

Question 6. Conclude that $TIME(f(\lfloor n/2\rfloor)/2) \subsetneq TIME(f(n)^3)$.

Question 7. Show that $P \subsetneq EXPTIME \subsetneq 2-EXPTIME$.
We recall: $P = \bigcup_{k\in\mathbb{N}} TIME(n^k)$, $EXPTIME = \bigcup_{k\in\mathbb{N}} TIME(2^{n^k})$, $2-EXPTIME = \bigcup_{k\in\mathbb{N}} TIME(2^{2^{n^k}})$

3 An $EXPTIME$-complete problem

Question 8. Justify that $f(x) = 2^x$ is time constructible and show that $Halt_f \in EXPTIME$.

Question 9. Let $L \in EXPTIME$, show that L is polynomial time reductible to $Halt_f$ and thus prove that $Halt_f$ is $EXPTIME$-complete.

4 A $PSPACE$-complete problem

The set of Quantified Boolean Formulæ (QBF) is defined by induction as:

- all propositional variables are QBF;
- if ϕ is QBF then $\neg\phi$ also is QBF;
- if ϕ and ψ are QBF then $\phi \wedge \psi$ is QBF
- if ϕ is QBF and p is a propositional variable then $\forall p\phi$ and $\exists p\phi$ are QBF

QBF are equipped with their usual Boolean valuation. The language of $TQBF$ is the language of QBF evaluating to true.
Question 10. Show that $TBQF$ is $PSPACE$-complete.

Suggestion for $PSPACE$ hardness: start from $L \in PSPACE$. Show that L is decided by K for which it exists $P \in \mathbb{N}[X]$ such that on an input of size n, K decides in time $2^{P(n)}$ and space $P(n)$. Reduce L to $TQBF$. Encode the state of K and its memory with a $P(n)$ uplet of $\{0,1\}$ and build a formula $\Phi(c_1, c_2, t)$ describing whether K can move from the state c_1 to c_2 in time 2^t (use a fast exponentiation).