DM de Complexité (L3)

à rendre avant le mercredi 20 à 14h

Ce devoir n'est ni long ni difficile mais les correcteurs attendent une rédaction soignée. D'où le conseil suivant : commencez la rédaction au brouillon avant de finaliser, ne vous y prenez pas au dernier moment pour rédiger la copie finale, relisez-vous soigneusement (idéalement : un autre jour) avant de rendre votre copie. **Attention : la date limite ne sera pas repoussée**.

Seules les questions marquées d'une étoile demandent soit un peu d'inventivité, soit une construction un peu lourde à mettre en place. Les autres questions sont élémentaires et ne demandent que d'avoir compris les concepts vu en cours.

Exercice 1 : Formules booléennes et arithmétique.

On note $\mathsf{NATURALS}$ le langage des nombres entiers naturels écrits en binaire, i.e., sur l'alphabet $\Sigma = \{\mathtt{0}, \mathtt{1}\}$. Pour fixer les choses, on considère qu'il s'agit de nombres écrits sans marqueurs particuliers, commençant par le chiffre le plus significatif : $\mathtt{1101}$ représente le nombre 13. On considère que toute suite de chiffre est valide et représente un nombre. Ainsi, le nombre 0 peut être représenté par « $\mathtt{0}$ », par « $\mathtt{000}$ », mais aussi par le mot vide, ϵ. On note $\mathsf{PROPFORMULAS}$ le langage des formules booléennes propositionnelles, telles que « $\top$ », « $p \vee \neg q$ », « $(q_1 \vee q_2) \implies \bot$ », ..., écrites dans la syntaxe de votre choix qu'on ne cherchera pas à préciser en détail. $\mathsf{SAT} \subseteq \mathsf{PROPFORMULAS}$ est le langage des formules satisfaisables et $\mathsf{DIV3} \subseteq \mathsf{NATURALS}$ est le langage des entiers divisibles par 3.

1. Définissez une réduction logspace $f : \Sigma^* \to \mathsf{PROPFORMULAS}$ montrant que $\mathsf{DIV3} \leq \mathsf{SAT}$. Énoncez et démontrez sa correction.

Solution:

Plusieurs approches sont possibles.

Exemple 1 : on définit deux langages supplémentaires $\mathsf{DIV3}', \mathsf{DIV3}'' \subseteq \mathsf{NATURALS}$ qui contiennent les (écritures des) nombres congrus à 1 (ou 2) modulo 3, et on définit 3 fonctions $f_0, f_1, f_2 : \Sigma^* \to \mathsf{PROPFORMULAS}$ montrant $\mathsf{DIV3}, \mathsf{DIV3}', \mathsf{DIV3}'' \leq \mathsf{SAT}$ respectivement. La réduction f_0 est celle que demande l'énoncé.

Une instance $x \in \Sigma^*$ est de la forme $x_1 x_2 \cdots x_n$. Les réductions sont définies simultanément (les programmeurs parlent de récurrence mutuelle) par induction sur la longueur de x :

$$f_0(x_1 \dots x_n) = \begin{cases} \top & \text{si } n = 0, \\ f_0(x_1 \dots x_{n-1}) & \text{si } n > 0 \text{ et } x_n = \mathtt{0}, \\ f_1(x_1 \dots x_{n-1}) & \text{si } n > 0 \text{ et } x_n = \mathtt{1}, \end{cases}$$

$$f_1(x_1 \dots x_n) = \begin{cases} \bot & \text{si } n = 0, \\ f_2(x_1 \dots x_{n-1}) & \text{si } n > 0 \text{ et } x_n = \mathtt{0}, \\ f_0(x_1 \dots x_{n-1}) & \text{si } n > 0 \text{ et } x_n = \mathtt{1}, \end{cases}$$

$$f_2(x_1 \dots x_n) = \begin{cases} \bot & \text{si } n = 0, \\ f_1(x_1 \dots x_{n-1}) & \text{si } n > 0 \text{ et } x_n = \mathtt{0}, \\ f_2(x_1 \dots x_{n-1}) & \text{si } n > 0 \text{ et } x_n = \mathtt{1}. \end{cases}$$

La correction découle d'équivalences telles que « $p \equiv 2 \mod 3$ *ssi* $2p+1 \equiv 2 \mod 3$ » et d'observations telles que « *une écriture de* $2p+1$ *est obtenue en ajoutant le chiffre* `1` *à la droite d'une écriture de* p ». On note que la réduction proposée ne peut produire que les formules $\top$ et $\bot$, c.-à-d. que f_0 fait déjà tout le travail de décider si $x \in \mathsf{DIV3}$ puisqu'on a

$$x \in \mathsf{DIV3} \text{ ssi } f_0(x) = \top \quad (\text{ssi } f(x) \in \mathsf{SAT})\,.$$

Évidemment f_0 est logspace : il suffit de lire x *de droite à gauche*, en mémorisant l'indice $i \in \{0,1,2\}$ désignant la fonction f_i qu'on cherche à appliquer au préfixe courant (on démarre avec $i=0$).

Exemple 2 : on peut associer à chaque chiffre x_i une variable booléenne p_i dont la valeur est fixée par x_i, puis définir les variables $\{q_i^0, q_i^1, q_i^2\}_{i=0,..,n}$ de telle sorte que q_i^j soit vraie ssi le préfixe $x_1 \cdots x_i$ de x représente un entier congru à j modulo 3. On définit alors

$$f(x_1 \cdots x_n) = q_0^n \wedge \psi_n \wedge \bigwedge_{0<i\leq n} \left\{ \begin{array}{ll} p_i & \text{si } x_i = 1, \\ \neg p_i & \text{sinon} \end{array} \right\}$$

$$\text{avec } \psi_n = \left\{ \begin{array}{l} q_0^0 \wedge \bigwedge\limits_{0<i\leq n} q_i^0 \;\Leftrightarrow\; (p_i \wedge q_{i-1}^1) \vee (\neg p_i \wedge q_{i-1}^0) \\ \wedge \neg q_0^1 \wedge \bigwedge\limits_{0<i\leq n} q_i^1 \;\Leftrightarrow\; (p_i \wedge q_{i-1}^0) \vee (\neg p_i \wedge q_{i-1}^2) \\ \wedge \bigwedge\limits_{0\leq i\leq n} (\neg q_i^2) \;\Leftrightarrow\; (q_i^0 \vee q_i^1) \end{array} \right.$$

Ici, la formule auxiliaire ψ_n ne dépend que de la longueur de x. Elle *définit* les q_i^j de façon unique en fonction des p_i, c.-à-d. qu'une valuation quelconque des p_i ne peut être étendue que d'une unique façon en une valuation satisfaisant ψ_n. Par ailleurs $f(x)$ définit les p_i et n'est satisfaisable que si q_n^0 s'évalue à vrai, c.-à-d. si x représente un entier congru à 0 modulo 3. Évidemment, f est logspace.

On considère $\mathsf{COMPOSITES} \subseteq \mathsf{NATURALS}$ qui est le langage des nombres entiers ≥ 2 qui ne sont pas des nombres premiers.

(*) 2. Définissez une réduction logspace $f : \Sigma^* \to \mathsf{SAT}$ montrant $\mathsf{COMPOSITES} \leq \mathsf{SAT}$. Énoncez et démontrez sa correction.

Solution:
$x \in \mathsf{COMPOSITES}$ ssi il existe deux entiers $a, b \geq 2$ dont le produit $c = ab$ est (représenté par) x. On va décrire les chiffres de ces entiers par des variables booléennes, et utiliser d'autres variables pour représenter les étapes de la multiplication de a par b.

S'ils existent, a, b ne nécessitent pas plus de n chiffres binaires donc on peut les écrire $a = \sum_{i=0}^{n-1} a_i 2^i$ et $b = \sum_{i=0}^{n-1} b_i 2^i$, et on peut, par abus de notation, considérer les a_i et b_i comme des variables booléennes. Pour $l = 0, \ldots, n$, on va noter $c_\ell = a \times (\sum_{i<\ell} b_i 2^i)$ le produit partiel où seuls les ℓ chiffres les moins significatifs de b ont été pris en compte. Il suffit de $2n$ chiffres pour écrire chaque c_ℓ sous la forme $c_\ell = \sum_{i<2n} c_{\ell,i} 2^i$ et donc on va introduire $2n(n+1)$ variables booléennes supplémentaires, les $c_{\ell,i}$ pour $0 \leq \ell \leq n$ et $0 \leq i < 2n$, ainsi que, pour faire bonne mesure, $2n(n+1)$ variables $r_{\ell,i}$ qui indiquent si le calcul du chiffre $c_{\ell,i}$ donne lieu à une retenue (qui participera alors au calcul de $c_{\ell,i+1}$). Les $c_{\ell,i}$ et $r_{\ell,i}$ sont complètement définis en fonction des a_i et b_i : d'abord pour $\ell = 0$ on a $c_{0,i} = r_{0,i} = 0$ pour tout i ; ensuite pour $\ell > 0$, le cas où $b_{\ell-1} = 0$ est simple : $c_{\ell,i} = c_{\ell-1,i}$ et $r_{\ell,i} = 0$; enfin si $b_{\ell-1} = 1$ on aura

$$c_{\ell,i} = c_{\ell-1,i} + r_{\ell,i-1} + a_{i-\ell+1} \mod 2\,, \qquad r_{\ell,i} = \left\lfloor \frac{c_{\ell-1,i} + r_{\ell,i-1} + a_{i-\ell+1}}{2} \right\rfloor . \qquad (\dagger)$$

Notons que $a_{i-\ell+1}$ dans la formule (†) ne correspond pas à un chiffre de a si $i < \ell-1$, ni si $i \geq n+\ell-1$: au lieu de modifier la formule pour ces cas particuliers, on introduira des variables

supplémentaires $a_n, a_{n+1}, \ldots, a_{2n-1}$ et $a_{-1}, a_{-2}, \ldots, a_{-n+1}$ qui vaudront systématiquement 0. Pour la même raison, on posera $r_{\ell,-1} = 0$ pour $\ell = 0, \ldots, n$.

Notons enfin qu'on peut éviter de distinguer deux cas suivant la valeur de $b_{\ell-1}$. Quand $b_{\ell-1} = 0$, on peut utiliser (†) si on y remplace $a_{i-\ell+1}$ par $a_{i-\ell+1} \times b_{\ell-1}$.

Nous sommes prêts à coder toutes ces contraintes en logique booléenne :

- $\psi_1 \equiv \left(\bigwedge_{i=n}^{2n-1} \neg a_i\right) \wedge \left(\bigwedge_{i=1}^{n-1} \neg a_{-i}\right) \wedge \left(\bigwedge_{\ell=0}^{n} \neg r_{\ell,-1}\right)$ pour les variables supplémentaires ;
- $\psi_2 \equiv \left(\bigvee_{i=1}^{n-1} a_i\right) \wedge \left(\bigvee_{i=1}^{n-1} b_i\right)$ pour forcer $a, b \geq 2$;
- $\psi_3 \equiv \bigwedge_{i=0}^{n-1} \left(c_{n,i} \Leftrightarrow x_{n-i} = 1\right) \wedge \bigwedge_{i=n}^{2n-1} \neg c_{n,i}$ pour garantir que c_n est bien le nombre représenté par x ;
- $\psi_4 \equiv \bigwedge_{i=0}^{2n-1} \neg c_{0,i}$ pour la définition de c_ℓ quand $\ell = 0$;
- $\psi_5 \equiv \bigwedge_{\ell=1}^{n} \bigwedge_{i=0}^{2n-1} \phi_{\ell,i} \wedge \phi'_{\ell,i}$ où, pour $\ell > 0$, $\phi_{\ell,i} \wedge \phi'_{\ell,i}$ exprime (†) par
  $\phi_{\ell,i} \equiv c_{\ell,i} \Leftrightarrow c_{\ell-1,i} \oplus r_{\ell,i-1} \oplus (a_{i-\ell+1} \wedge b_{\ell-1})$
  $\phi'_{\ell,i} \equiv r_{\ell,i} \Leftrightarrow \left[a_{i-\ell+1} \wedge b_{\ell-1} \wedge (c_{\ell-1,i} \vee r_{\ell,i-1})\right] \vee \left[\neg(a_{i-\ell+1} \wedge b_{\ell-1}) \wedge c_{\ell-1,i} \wedge r_{\ell,i-1}\right]$
  utilisant $\oplus$ pour le « ou exclusif ».

Si $\psi \equiv \psi_1 \wedge \cdots \wedge \psi_5$ est satisfaisable, la valuation qui satisfait ψ donne les chiffres binaires de nombres $a, b, c_0, c_1, \ldots, c_n = c$ garantissant $a, b \geq 2$, $c_0 = 0$, $c_{\ell+1} = c_\ell + ab_\ell 2^\ell$ pour tout ℓ, etc., donc montrant que $c = ab$, le nombre représenté par x, n'est pas premier. Réciproquement, si c n'est pas premier, on tire d'une factorisation $c = ab$ une valuation basée sur les chiffres de a, b et des produits partiels $c_0, \ldots, c_n$ qui satisfait ψ.

Exercice 2 : Langages des sous-mots

On considère des mots $u, v, w, \ldots$ sur un alphabet fini Σ. On note $u \preceq v$ si u est un sous-mot de v, c.-à-d., est obtenu en supprimant des lettres n'importe où dans v. P.ex. $\texttt{baba} \preceq \texttt{barbara}$. On note $\downarrow u$ l'ensemble des sous-mots de u et, pour un langage $L \subseteq \Sigma^*$, $\downarrow L$ l'ensemble des sous-mots d'un mot de L. P.ex. $\downarrow(\texttt{abac})^+ = (\texttt{a} + \texttt{b} + \texttt{c})^*$.

3. Soit Σ fixé avec $|\Sigma| \geq 2$.
   Dites si les langages suivants sont dans NP ?, P ?, NL ?, L ?, SPACE(0) ?
   - $L_1 = \{(u, v) \mid u \preceq v\}$,
   - $L_2 = \{(u, v) \mid \exists w : u \preceq w \wedge v \preceq w\}$,
   - $L_3 = \{(u_1, u_2, \ldots, u_k, n) \mid \exists w : |w| = n \wedge w \preceq u_1 \wedge \cdots \wedge w \preceq u_k\}$ (NB : $n \in \mathbb{N}$),
   - $L_4 = \{(u, v) \mid \exists w : u \preceq w \wedge w.w \preceq v\}$.

   Pour cette question, et puisque $\mathsf{SPACE}(0) \subseteq \mathsf{L} \subseteq \mathsf{NL} \subseteq \mathsf{P} \subseteq \mathsf{NP}$, on demande d'indiquer la plus petite classe où vous savez situer chaque problème, et de justifier votre réponse pour l'appartenance. On ne demande pas de preuve de non-appartenance aux classes inférieures, ni de preuve de difficulté comme « L est NP-complet ».

Solution:
$L_1 \in \mathsf{L}$ car il suffit de chercher à construire le plongement le plus à gauche de u dans v ; $L_2 \in \mathsf{SPACE}(0)$ est trivial car w existe toujours ; $L_3 \in \mathsf{NP}$ car il suffit de deviner w de taille n (après avoir vérifié que $n \leq |u_1|$) et de vérifier en temps polynomial qu'il est sous-mot de chaque u_i ; enfin $L_4 \in \mathsf{L}$ car il suffit de tester si $u.u \preceq v$. (Le lecteur intéressé verra que L_3 est en fait NP-complet, cf. examen de l'an dernier.)

Soit $\mathcal{A} = (\Sigma, Q, I, F, \Delta)$ un automate fini non déterministe sur l'alphabet Σ. On note $n_\mathcal{A} = |Q|$ son nombre d'états et $L(\mathcal{A})$ le langage qu'il reconnaît.

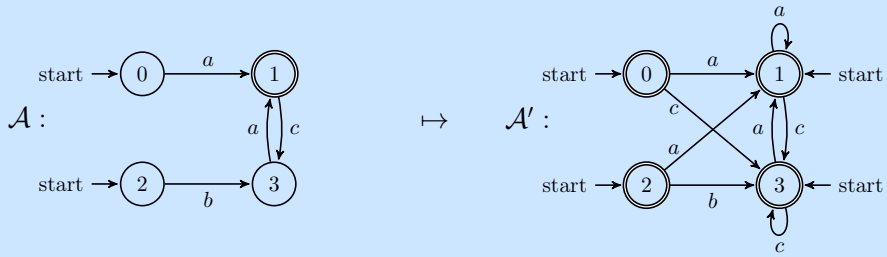
4. Montrez que le langage $\{(u, \mathcal{A}) \mid u \in L(\mathcal{A})\}$, c.-à-d. le problème, étant donnés un mot et un automate non déterministe, de dire si le mot est accepté par l'automate, est dans P.

Solution:

Pour $u = u_1 u_2 \cdots u_n$, on construit le run $S_0 \xrightarrow{u_1} S_1 \xrightarrow{u_2} S_2 \cdots \xrightarrow{u_n} S_n$ dans l'automate des parties. Formellement, $S_0 = I$ et, pour $i > 0$, $S_i = \Delta(S_{i-1}, u_i)$ est l'ensemble des états de $\mathcal{A}$ accessibles depuis I par le préfixe $u_1 u_2 \cdots u_i$. À la fin, $u \in L(\mathcal{A})$ ssi $S_n \cap F \neq \emptyset$.

5. Montrez que le langage $\downarrow L(\mathcal{A})$ des sous-mots de $L(\mathcal{A})$ est reconnu par un automate non déterministe $\mathcal{A}'$ tel que $n_{\mathcal{A}'} \leq n_{\mathcal{A}}$.

Solution:
On construit $\mathcal{A}'$ en modifiant $\mathcal{A}$: on ajoute comme états finaux tous les états desquels on peut atteindre un état de F, on ajoute comme états initiaux tous les états accessibles depuis I, et on ajoute une transition $q \xrightarrow{a} q''$ pour chaque état $q \in Q$ et chaque transition $q' \xrightarrow{a} q''$ telle que q' est accessible depuis q. Voici un exemple illustratif :



(*) 6. Soient $\mathcal{A}, \mathcal{B}$ deux automates finis non déterministes sur l'alphabet Σ. Montrez que si $\downarrow L(\mathcal{A}) \not\subseteq \downarrow L(\mathcal{B})$ alors il existe un mot u tel que $u \in \downarrow L(\mathcal{A})$, $u \notin \downarrow L(\mathcal{B})$ et $|u| \leq n_{\mathcal{B}}$.

Solution:
Soit $u = u_1 u_2 \cdots u_n$ le mot le plus court dans $\downarrow L(\mathcal{A}) \setminus \downarrow L(\mathcal{B})$. On considère l'automate $\mathcal{B}' = (\Sigma, Q_{\mathcal{B}}, I', F', \Delta')$ qui reconnaît $\downarrow L(\mathcal{B})$ comme vu à la question 5. Le run de u sur l'automate des parties basé sur $\mathcal{B}'$ a la forme $I' = S_0 \xrightarrow{u_1} S_1 \xrightarrow{u_2} S_2 \cdots \xrightarrow{u_n} S_n$ avec $S_n \cap F' = \emptyset$. Ces parties $S_i \subseteq Q_{\mathcal{B}}$ vérifient $S_{i-1} \supseteq S_i$ car S_0 contient tous les états accessibles, et si $q \in S_i$ est accessible par le mot $u_1 \cdots u_i$, alors les transitions ajoutées à $\mathcal{B}$ pour obtenir $\mathcal{B}'$ permettent d'atteindre q via $u_1 \cdots u_{i-1}$, donc $q \in S_{i-1}$. On a donc $I' = S_0 \supseteq S_1 \supseteq S_2 \supseteq \cdots \supseteq S_n$. Si pour un $i \in \{1, \ldots, n\}$ on a $S_{i-1} = S_i$, alors on peut retirer la lettre u_i de u : le mot u' résultant est toujours dans $\downarrow L(\mathcal{A})$ et on dispose bien d'un run montrant $u' \notin \downarrow L(\mathcal{B})$. Donc si u est le plus court possible, nécessairement $I' = S_0 \supsetneq S_1 \supsetneq S_2 \cdots \supsetneq S_n$. On déduit $n \leq |S_0| = |I'| \leq n_{\mathcal{B}}$.

7. On note $\mathsf{NO\_LESS\_SUBWORDS}^{\Sigma}_{ND}$ le langage des paires $(\mathcal{A}, \mathcal{B})$ d'automates non déterministes sur Σ tels que $L(\mathcal{A}) \not\subseteq \downarrow L(\mathcal{B})$. Montrez que ce langage est dans NP.

Solution:
Ce problème est équivalent à $\downarrow L(\mathcal{A}) \not\subseteq \downarrow L(\mathcal{B})$ et on a montré, avec les questions 5 et 6 l'existence d'un témoin de taille polynomiale. La question 4 a rappelé qu'on pouvait vérifier ce témoin en temps déterministe polynomial.

8. Pour $\Sigma = \{0, 1\}$, montrez que $\mathsf{3SAT} \leq \mathsf{NO\_LESS\_SUBWORDS}^{\Sigma}_{ND}$. Décrivez précisément votre construction, énoncez tous les points nécessaires à sa correction, et justifiez les.

Indication : on pourra chercher à associer à toute formule booléenne en forme normale conjonctive $\phi \equiv \bigwedge_{i=1}^{m} \bigvee_{j=1}^{3} l_{i,j}$, où chaque littéral $l_{i,j}$ est soit une variable booléenne $x_{i,j}$, soit la négation $\neg x_{i,j}$ d'une variable booléenne, prise parmi l'ensemble $Var_\phi = \{x_1, \ldots, x_r\}$, un automate non déterministe sur Σ qui reconnaît exactement l'ensemble des mots $b_1 \cdots b_r \in \Sigma^r$ tels que la valuation $v_{b_1 \cdots b_r}$ satisfait $\neg\phi$. Ici on identifie de façon canonique un mot de Σ^r et une valuation $v : Var_\phi \to \{\top, \bot\}$ des variables de ϕ.

Solution:
L'automate, qu'on va appeler $\mathcal{B}_{\neg\phi}$, est très simple : il choisit une clause $C_j = l_{j,1} \vee l_{j,2} \vee l_{j,3}$ de façon non déterministe et vérifie que la valuation $b_1 \cdots b_r$ invalide C_j, auquel cas on

accepte le mot. P.ex. si $C_1 = x_1 \vee \neg x_2 \vee x_4$, on accepte le mot $b_1 \cdots b_r$ si $b_1 = \mathtt{0}$ et $b_2 = \mathtt{1}$ et $b_4 = \mathtt{0}$. Cet automate a exactement $m(r+1)$ états, qu'on peut ramener à $mr+1$ en fusionnant les états finaux et on a $L(\mathcal{B}_{\neg\phi}) = \{b_1 \cdots b_r \mid v_{b_1 \cdots b_r} \not\models \phi\}$.

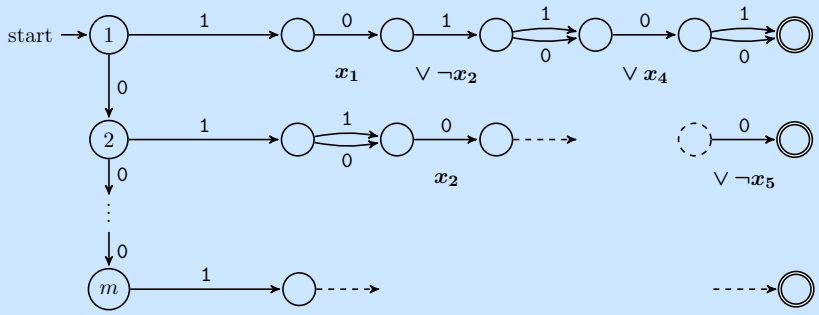
On a alors $L(\mathcal{B}_{\neg\phi}) = \Sigma^r$ ssi ϕ n'est pas satisfaisable. Donc en prenant un automate $\mathcal{A}_r$ qui accepte exactement Σ^r (ici $r+1$ états sont suffisants) on a $L(\mathcal{A}_r) \not\subseteq L(\mathcal{B}_{\neg\phi})$ ssi ϕ est satisfaisable. Par ailleurs, comme les deux automates n'acceptent que des mots de longueur exactement r, on a $L(\mathcal{A}_r) \not\subseteq L(\mathcal{B}_{\neg\phi})$ ssi $L(\mathcal{A}_r) \not\subseteq {\downarrow}L(\mathcal{B}_{\neg\phi})$, ce qui établit la correction de la réduction $\phi \mapsto (\mathcal{A}_r, \mathcal{B}_{\neg\phi})$. Il reste à vérifier que cette réduction est logspace.

(*) 9. On note $\mathsf{NO\_LESS\_SUBWORDS}^{\Sigma}_{D}$ pour le fragment de $\mathsf{NO\_LESS\_SUBWORDS}^{\Sigma}_{ND}$ réduit aux automates déterministes. Montrez que $\mathsf{3SAT} \leq \mathsf{NO\_LESS\_SUBWORDS}^{\Sigma}_{D}$, toujours pour $\Sigma = \{\mathtt{0}, \mathtt{1}\}$.

Solution:

Dans la réduction de la question précédente, $\mathcal{A}_r$ est déterministe mais pas $\mathcal{B}_{\neg\phi}$. On va donc modifier un peu cette réduction en considérant un nouveau langage associé à ϕ, $L'_{\neg\phi} = \{\mathtt{0}^{j-1}\mathtt{1}b_1 \cdots b_r \mid v_{b_1 \cdots b_r} \not\models C_j\}$, qui peut être reconnu par un automate *déterministe* $\mathcal{B}'_{\neg\phi}$ à $n_{\mathcal{B}'_{\neg\phi}} = m(r+2)$ états.

Voici $\mathcal{B}'_{\neg\phi}$ pour $\phi \equiv (x_1 \vee \neg x_2 \vee x_4) \wedge (x_2 \vee \cdots \vee \neg x_5) \wedge \cdots \wedge C_m$ où $r = |Var_\phi| = 5$:



Pour un mot $u \in L'_{\neg\phi}$, le préfixe $\mathtt{0}^{j-1}\mathtt{1}$ indique quelle clause on doit invalider et permet de circonscrire le non-déterminisme. On laisse le lecteur vérifier que $\mathtt{1}\Sigma^r \subseteq {\downarrow}L(\mathcal{B}'_{\neg\phi})$ ssi ϕ n'est pas satisfaisable, et compléter la réduction.

10. Quels problèmes parmi $\mathsf{NO\_LESS\_SUBWORDS}^{\Sigma}_{D}$ et $\mathsf{NO\_LESS\_SUBWORDS}^{\Sigma}_{ND}$ sont NP-complets ? On répondra dans les trois cas suivants : $|\Sigma|$=2, $|\Sigma| > 2$, et $|\Sigma| = 1$.

Solution:

Pour $|\Sigma| = 2$, les deux problèmes sont NP-durs (questions 8 et 9) et les deux sont dans NP (question 7), donc les deux sont NP-complets.

L'appartenance à NP (question 7) ne met pas de condition sur la taille de Σ donc ces problèmes restent NP-complets quand $|\Sigma| > 2$.

Dans le cas $|\Sigma| = 1$, on a $u \preceq v$ ssi v est au moins aussi long que u. On peut donc décider si $L(\mathcal{A}) \not\subseteq {\downarrow}L(\mathcal{B})$ en calculant la longueur maximale ($\in \mathbb{N} \cup \{\infty\}$) des mots acceptés par $\mathcal{A}$ et $\mathcal{B}$ et en les comparant, ce qui se fait facilement dans P et même dans NL.