École polytechnique — X2016 — INF412 Fondements de l'informatique

PC5 notée

Sujet proposé par Stéphane Graham-Lengrand et Samuel Mimram et Bruno Salvy
(corrigé)

Cet énoncé comporte trois parties indépendantes et qui pourront être résolues dans n'importe quel ordre. Dans chaque partie, on pourra, pour répondre à une question, admettre les résultats dont on demande la démonstration aux questions *précédentes*. Il n'est pas nécessaire de traiter toutes les questions pour avoir la note maximale. Les correcteurs vous remercient d'avance d'écrire lisiblement.

1 Complétude équationnelle des algèbres de Boole

Dans cette exercice, on fixe un entier n et on considère les formules propositionnelles sur l'ensemble de variables $\mathcal{V} = \{x_1, \dots, x_n\}$ et les connecteurs $\wedge$, $\vee$, $\neg$, $\top$ et $\bot$ (ces deux derniers étant respectivement interprétés par vrai et faux). On note $\approx$, la plus petite relation d'équivalence sur les formules telle que

$$\begin{array}{rcll}
(a \wedge b) \wedge c & \approx & a \wedge (b \wedge c) & (A) \\
a \wedge \top & \approx & a & (B) \\
a \wedge \bot & \approx & \bot & (C) \\
a \wedge b & \approx & b \wedge a & (D) \\
a \wedge a & \approx & a & (E) \\
a \wedge (b \vee c) & \approx & (a \wedge b) \vee (a \wedge c) & (F) \\
\neg(a \wedge b) & \approx & \neg a \vee \neg b & (G) \\
a \wedge \neg a & \approx & \bot & (H) \\
\neg\neg a & \approx & a & (I)
\end{array}
\qquad
\begin{array}{rcll}
(a \vee b) \vee c & \approx & a \vee (b \vee c) & (J) \\
a \vee \bot & \approx & a & (K) \\
a \vee \top & \approx & \top & (L) \\
a \vee b & \approx & b \vee a & (M) \\
a \vee a & \approx & a & (N) \\
a \vee (b \wedge c) & \approx & (a \vee b) \wedge (a \vee c) & (O) \\
\neg(a \vee b) & \approx & \neg a \wedge \neg b & (P) \\
a \vee \neg a & \approx & \top & (Q)
\end{array}$$

et qui soit une congruence, c'est-à-dire si $F \approx F'$ et $G \approx G'$ alors $F \wedge G \approx F' \wedge G'$, $F \vee G \approx F' \vee G'$ et $\neg F \approx \neg F'$.

Deux formules F et G telles que $F \approx G$ sont dites *congruentes*. La *valeur de vérité* d'une formule est la fonction $\{0,1\}^n \to \{0,1\}$ qui lui est associée, de la façon décrite dans le cours. On admettra que deux formules congruentes ont toujours la même valeur de vérité. L'objectif de cet exercice est de montrer la réciproque : si deux formules admettent la même valeur de vérité alors elles sont congruentes.

Question 1.1. *Montrez que la formule*

$$\neg x_1 \wedge \neg(x_3 \wedge \neg x_1) \tag{1}$$

est congruente à une formule sous forme normale disjonctive.

Solution : On a

$$\begin{aligned}
\neg x_1 \wedge \neg(x_3 \wedge \neg x_1) &\approx \neg x_1 \wedge (\neg x_3 \vee \neg\neg x_1) && (G) \\
&\approx \neg x_1 \wedge (\neg x_3 \vee x_1) && (I) \\
&\approx (\neg x_1 \wedge \neg x_3) \vee (\neg x_1 \wedge x_1) && (F)
\end{aligned}$$

□

Dans la suite on admettra que toute formule est congruente à une formule sous forme normale disjonctive. Une formule F est *canonique* si elle est sous forme normale disjonctive

$$F \quad = \quad \bigvee_{i=1}^{k} C_i \tag{2}$$

et chaque formule C_i est une conjonction de littéraux, appelée *cube*, de la forme

$$C_i \quad = \quad L_{i,1} \wedge L_{i,2} \wedge \ldots \wedge L_{i,n} \tag{3}$$

où $L_{i,j}$ est soit x_j, soit $\neg x_j$, pour $1 \leq j \leq n$.

Autrement dit dans ces cubes, toutes les variables dans $\mathcal{V} = \{x_1, \ldots, x_n\}$ apparaissent exactement une fois, et dans l'ordre.

Question 1.2. *Montrez que la formule* (1) *est congruente à une formule canonique (pour* $n = 3$*).*

Solution : On poursuit le raisonnement de la question précédente :

$$\begin{aligned}
\neg x_1 \wedge \neg(x_3 \wedge \neg x_1) &\approx (\neg x_1 \wedge \neg x_3) \vee (\neg x_1 \wedge x_1) \\
&\approx (\neg x_1 \wedge \neg x_3) \vee \bot && (D) \text{ et } (H) \\
&\approx \neg x_1 \wedge \neg x_3 && (K) \\
&\approx \neg x_1 \wedge \neg x_3 \wedge \top && (B) \\
&\approx (\neg x_1 \wedge \neg x_3) \wedge (x_2 \vee \neg x_2) && (Q) \\
&\approx ((\neg x_1 \wedge \neg x_3) \wedge x_2) \vee ((\neg x_1 \wedge \neg x_3) \wedge \neg x_2) && (F) \\
&\approx (\neg x_1 \wedge (\neg x_3 \wedge x_2)) \vee (\neg x_1 \wedge (\neg x_3 \wedge \neg x_2)) && (A) \\
&\approx (\neg x_1 \wedge (x_2 \wedge \neg x_3)) \vee (\neg x_1 \wedge (\neg x_2 \wedge \neg x_3)) && (D)
\end{aligned}$$

(dans la suite on omettra les étapes d'associativité). □

Question 1.3. *Montrez que toute formule est congruente à une formule canonique.*

Solution : Étant donnée une formule F, cette formule est congruente à une formule sous forme normale disjonctive, de la forme donnée par (2) et (3). S'il existe un cube C_i qui ne contient pas une variable x_j, on utilise la suite de congruences

$$C_i \approx C_i \wedge \top \approx C_i \wedge (x_j \vee \neg x_j) \approx (C_i \wedge x_j) \vee (C_i \wedge \neg x_j)$$

qui nous permet de remplacer le cube C_i par les deux cubes $C_i \wedge x_j$ et $C_i \wedge \neg x_j$. Pour j variant de 1 à n, effectue itérativement cette procédure sur toutes les cubes ne contenant pas x_j. À la fin de la procédure, tous les cubes contiennent toutes les variables. Par permutations successives, on peut mettre ces variables dans l'ordre dans chaque cube. Enfin, si une variable x_j apparaît deux fois dans un cube C_i, celui-ci est de l'une des deux formes suivantes :

— $C_i' \wedge x_j \wedge x_j \wedge C_i''$: dans ce cas on le remplace par le cube congruent $C_i' \wedge x_j \wedge C_i''$,

— $C_i' \wedge x_j \wedge \neg x_j \wedge C_i''$: dans ce cas le cube est congruent à $\bot$ et on peut l'enlever.

Ces opérations faisant strictement diminuer la longueur de la formule, la procédure consistant à les appliquez itérativement jusqu'à ce que la formule ne contienne pas deux fois une même variable termine. La formule résultante est sous forme canonique. □

Question 1.4. *Donnez la valeur de vérité de la formule* (1)*.*

Solution : La valeur de vérité de cette formule est la fonction $f : \{0,1\}^3 \to \{0,1\}$ qui vaut 1 lorsqu'elle est appliquée à $(0,1,0)$ ou $(0,0,0)$ et 0 sinon. □

Question 1.5. *Étant donnée une formule sous forme canonique, notée comme dans* (2) *et* (3)*, donnez sa valeur de vérité en fonction des littéraux apparaissant dans les différents cubes.*

Solution : La disjonction $\bigvee_{i=1}^{k} C_i$ est vraie si et seulement si l'un des C_i est vrai, et un C_i est vrai si et seulement si tous les littéraux apparaissant dedans sont vrais. Étant donné un littéral L, on note b_L le booléen qui vaut 0 si L est une variable niée et 1 sinon. La valeur de vérité de la formule est donnée par la fonction $f : \{0,1\}^n \to \{0,1\}$ telle que $f(b_1, \ldots, b_n) = 1$ si et seulement si il existe i avec $1 \leq i \leq k$ tel que $(b_1, \ldots, b_n) = (b_{L_{i,1}}, \ldots, b_{L_{i,n}})$. □

Question 1.6. *Donnez trois formules canoniques distinctes qui ont la même valeur de vérité que* (1)*.*

Solution : On peut par exemple prendre

$$(\neg x_1 \wedge x_2 \wedge \neg x_3) \vee (\neg x_1 \wedge \neg x_2 \wedge \neg x_3)$$
$$(\neg x_1 \wedge \neg x_2 \wedge \neg x_3) \vee (\neg x_1 \wedge x_2 \wedge \neg x_3)$$
$$(\neg x_1 \wedge \neg x_2 \wedge \neg x_3) \vee (\neg x_1 \wedge \neg x_2 \wedge \neg x_3) \vee (\neg x_1 \wedge x_2 \wedge \neg x_3)$$

(de façon générale, on peut en obtenir un nombre arbitraire en permutant et en dupliquant les cubes). □

Question 1.7. *Montrez que deux formules F et G quelconques qui ont la même valeur de vérité sont nécessairement congruentes.*

Solution : Notons $f : \{0,1\}^n \to \{0,1\}$ la valeur de vérité des deux formules. Par la question 1.3, toute formule étant congruente à une formule sous forme canonique, on peut supposer que F et G sont sous cette forme. Par la question 1.5, un cube $C = L_1 \wedge \ldots \wedge L_n$ apparaît dans F si et seulement si $f(b_{L_1}, \ldots, b_{L_n}) = 1$ si et seulement si le cube apparaît aussi dans G : F et G contiennent donc les mêmes cubes. Supposons fixé un ordre total sur l'ensemble des cubes. En utilisant des congruences, on peut réordonner les cubes apparaissant dans F en utilisant la commutativité de la disjonction (M), puis enlever les doublons en utilisant l'idempotence (N) ; on peut faire de même pour G. Les deux formules résultantes contiennent alors la même liste de cubes et sont donc égales. On en déduit que les formules F et G sont congruentes. □

2 Théorie des différences

Terminologie : étant donné une signature Σ et une théorie $\mathcal{T}$ sur Σ, une formule F sur Σ est *$\mathcal{T}$-satisfiable* s'il existe un modèle $\mathfrak{M}$ de $\mathcal{T}$ et une valuation v qui satisfont F.

La théorie des différences [1] s'exprime avec la signature $\Sigma = (\emptyset, \emptyset, \mathcal{R})$ qui n'a aucune constante ni symbole de fonction, et où $\mathcal{R}$ est un ensemble dénombrable de symboles de relation $(R_c)_{c \in \mathbb{Z}}$, tous d'arité 2, et indexés par les entiers relatifs.

On considère la structure $\mathfrak{M}_{\mathbb{Z}}$ de signature Σ, dont le domaine de base est précisément l'ensemble $\mathbb{Z}$ des entiers relatifs, et qui interprète tout symbole R_c comme l'ensemble des paires (r, r') telles que $r - r' \leq c$.

La théorie des différences $\mathcal{D}$iff désigne l'ensemble des formules closes sur la signature Σ qui sont satisfaites dans la structure $\mathfrak{M}_{\mathbb{Z}}$.

Question 2.1. *Montrez que pour tous entiers relatifs c et c', la formule suivante est un axiome de la théorie $\mathcal{D}iff$.*

$$\forall x \forall y \forall z ((R_c(x,y) \wedge R_{c'}(y,z)) \Rightarrow R_{c+c'}(x,z)$$

1. *difference logic* en anglais

Solution : Cette formule est satisfaite dans le modèle $\mathfrak{M}_{\mathbb{Z}}$ car si $x - y \leq c$ et si $y - z \leq c'$ alors $x - z \leq c + c'$. □

On appelle "problème de différences" toute conjonction de formules atomiques sur Σ. On cherche à savoir si un tel problème est $\mathcal{D}$iff-satisfiable.

Pour un tel problème P, on définit le graphe $\mathcal{G}_P$ avec poids suivant : les sommets sont les variables libres de P, et pour toute formule atomique $R_c(x, y)$ dans la conjonction P, on place une arête orientée de poids c du sommet x vers le sommet y, notée $x \xrightarrow{c} y$.

On rappelle qu'un *chemin* d'un sommet x à un sommet y est une suite d'arêtes de la forme $x_0 \xrightarrow{c_1} x_1 \xrightarrow{c_2} \cdots \xrightarrow{c_n} x_n$ $(n \geq 0)$, avec $x = x_0$ et $y = x_n$, dont on définit le poids comme $c_1 + \cdots + c_n$.

On appelle *cycle négatif* un chemin qui va d'un sommet à lui-même et qui a un poids strictement négatif.

Question 2.2. *Montrez que s'il existe dans $\mathcal{G}_P$ un cycle négatif, alors le problème P n'est pas $\mathcal{D}$iff-satisfiable.*

Solution : Soient $x_0 \xrightarrow{c_1} x_1 \xrightarrow{c_2} \cdots \xrightarrow{c_n} x_n$ le cycle négatif et $R_{c_1}(x_0, x_1), \ldots, R_{c_n}(x_{n-1}, x_n)$ les formules atomiques de P correspondantes. Par la question 2.1, ces formules impliquent $R_{c_1+\cdots+c_n}(x_0, x_n)$, et par hypothèse, $x_0 = x_n$ et $c_1 + \cdots + c_n < 0$.

Or, $\forall x_0 \neg R_{c_1+\cdots+c_n}(x_0, x_0)$ est un axiome de $\mathcal{D}$iff, car c'est une formule satisfaite dans la structure $\mathfrak{M}_{\mathbb{Z}}$ (en effet, $\mathfrak{M}_{\mathbb{Z}} \models \neg(0 \leq c_1 + \cdots + c_n)$). Un modèle de $\mathcal{D}$iff avec une valuation v qui satiferaient P satisferaient également $R_{c_1+\cdots+c_n}(x_0, x_0)$, en violation de cet axiome ; P n'est donc pas $\mathcal{D}$iff-satisfiable. □

On cherche maintenant à montrer la réciproque. On définit le graphe $\mathcal{G}_P^*$ comme l'extension du graphe $\mathcal{G}_P$ avec un sommet de plus, y, et avec, pour tout sommet x de $\mathcal{G}_P$, une arête de plus $x \xrightarrow{0} y$. Pour tout sommet x de $\mathcal{G}_P$, on considère l'ensemble $\mathcal{C}(x)$ des chemins de x à y dans $\mathcal{G}_P^*$.

Question 2.3. *Montrez que si $\mathcal{G}_P$ n'a pas de cycle négatif, alors pour tout sommet x de $\mathcal{G}_P$, il existe dans $\mathcal{C}(x)$ un chemin de poids minimal. On notera $\delta(x)$ son poids.*

Solution : Remarquons que le nombre de chemins de x à y sans cycle (i.e. qui ne passent jamais deux fois par le même sommet), est fini. On considère alors parmi les chemins sans cycle un chemin p de poids minimal w. Montrons qu'aucun autre chemin de $\mathcal{C}(x)$ (avec ou sans cycle) ne peut avoir un poids strictement inférieur. Supposons qu'il en existe un, et parmi tout ceux qui existent, prenons-en un, p', de longueur minimale. Soit $w' < w$ son poids. Le chemin p' a forcément un cycle (par minimalité de p parmi les chemins sans cycle). En retirant ce cycle de p', on obtient un nouveau chemin de x à y, plus court que p'. Comme dans $\mathcal{G}_P$, tous les cycles sont de poids positifs ou nuls, le poids de p' est inférieur ou égal à $w' < w$, ce qui contredit le fait que p' est le plus court chemin de poids strictement inférieur à w. □

Question 2.4. *Montrez que si $\mathcal{G}_P$ n'a pas de cycle négatif, alors P est $\mathcal{D}$iff-satisfiable.*

Solution : On utilise le modèle $\mathfrak{M}_{\mathbb{Z}}$, qui vérifie tous les axiomes de $\mathcal{D}$iff. Il ne reste plus qu'à donner une valuation pour les variables libres de P, de manière à satisfaire P. On prend δ comme valuation. Vérifions que P est satisfait. Une formule atomique $R_c(x_1, x_2)$ dans P est satisfaite si et seulement si $\delta(x_1) - \delta(x_2) \leq c$. Par définition de $\delta(x_2)$, il existe un chemin de x_2 à y de poids $\delta(x_2)$. L'arête $x_1 \xrightarrow{c} x_2$ permet alors de construire un chemin de x_1 à y, de poids $\delta(x_2) + c$. Comme $\delta(x_1)$ est le poids d'un chemin minimal de x_1 à y, on a bien $\delta(x_1) - \delta(x_2) \leq c$. □

On cherche maintenant à généraliser la notion de "problème de différences". Faisons d'abord une première remarque :

Question 2.5. *Montrez que si $\mathfrak{M}_\mathbb{Z}$ avec la valuation v satisfait un problème de différences P, alors pour tout entier relatif c, $\mathfrak{M}_\mathbb{Z}$ avec la valuation v' satisfait P, où $v'(x) = v(x) + c$ pour toute variable libre x de P.*

Solution : L'interprétation de chaque formule atomique $R_{c'}(x_x, x_2)$ est la même avec v et avec v', car $v(x_1) - v(x_2) \leq c'$ si et seulement si $v'(x_1) - v'(x_2) \leq c'$. □

On étend alors la signature Σ en une signature Σ^+ qui rajoute à Σ une infinité de symboles de relation $(U_c)_{c\in\mathbb{Z}}$ et $(L_c)_{c\in\mathbb{Z}}$, tous d'arité 1, et indexés par les entiers relatifs. La structure $\mathfrak{M}_\mathbb{Z}$ de signature Σ est étendue en une structure $\mathfrak{M}^+_\mathbb{Z}$ de signature Σ^+ qui interprète tout symbole U_c comme l'ensemble des entiers r tels que $r \leq c$ et interprète tout symbole L_c comme l'ensemble des entiers r tels que $-r \leq c$. La théorie $\mathcal{D}\text{iff}^+$ désigne l'ensemble des formules closes sur Σ^+ qui sont satisfaites dans $\mathfrak{M}^+_\mathbb{Z}$. Enfin, on appelle "problème de différences borné" toute toute conjonction de formules atomiques sur Σ^+. On cherche à savoir si un tel problème est $\mathcal{D}\text{iff}^+$-satisfiable.

Question 2.6. *Pour tout problème de différences borné P, proposez un problème de différences P' tel que P est $\mathcal{D}iff^+$-satisfiable si et seulement si P' est $\mathcal{D}iff$-satisfiable.*
Indication : on pourra utiliser la question précédente.

Solution : On laisse dans P' toutes les contraintes de P de la forme $R_c(x_1, x_2)$. On introduit une nouvelle variable z et pour chaque contrainte de P de la forme $U_c(x)$ (respectivement $L_c(x)$), on ajoute dans P une contrainte $R_c(x, z)$ (respectivement $R_c(z, x)$).

Soient $\{x_1, \ldots, x_n\}$ les variables libres de P.

Si P' est $\mathcal{D}$iff-satisfiable, alors $\forall x_1, \ldots, x_n \neg P'$ ne peut pas être un axiome de $\mathcal{D}$iff et donc ne peut pas être satisfaite dans $\mathfrak{M}_\mathbb{Z}$. Il doit donc exister une valuation v telle que $\mathfrak{M}_\mathbb{Z}$ et la valuation v satisfont P'. Soit v' la valuation définie par $v'(x) = v(x) - v(z)$ pour toute variable x de P'. Par la question précédente, le modèle $\mathfrak{M}_\mathbb{Z}$ et la valuation v' satisfont également P', et interprètent z par 0. Clairement, $\mathfrak{M}^+_\mathbb{Z}$ et la valuation v' satisfont P.

Réciproquement, si P est $\mathcal{D}\text{iff}^+$-satisfiable, alors $\forall x_1, \ldots, x_n \neg P$ ne peut pas être un axiome de $\mathcal{D}\text{iff}^+$ et donc ne peut pas être satisfaite dans $\mathfrak{M}^+_\mathbb{Z}$. Il doit donc exister une valuation v telle que $\mathfrak{M}^+_\mathbb{Z}$ et la valuation v satisfont P. Soit v' la valuation qui étend v avec $z \mapsto 0$. Clairement, $\mathfrak{M}_\mathbb{Z}$ et la valuation v' satisfont P'. □

On continue d'étendre la notion de "problème de différences".

Question 2.7. *Soit F une formule sur Σ^+, sans quantificateur et en forme normale disjonctive. Proposez un ensemble fini de problèmes de différences $\{P_1, \ldots, P_n\}$, tel que F est $\mathcal{D}iff^+$-satisfiable si et seulement si l'un des problèmes $\{P_1, \ldots, P_n\}$ est $\mathcal{D}iff$-satisfiable.*

Solution : F est de la forme $F_1 \vee \cdots \vee F_n$, où chaque F_i est un *cube*, i.e. une conjonction de litéraux. Chaque cube F_i est d'abord transformé en un problème de différences borné, constitué de chaque litéral de F_i qui n'est pas nié, ainsi que, pour chaque litéral nié $\neg R_c(x, y)$ (respectivement $\neg U_c(x)$, $\neg L_c(x)$) dans F_i, de la formule atomique $R_{(-c-1)}(y, x)$ (respectivement $L_{(-c-1)}(x)$, $U_{(-c-1)}(x)$). Clairement, ce problème de différences borné est $\mathcal{D}\text{iff}^+$-satisfiable si et seulement si F_i l'est. Puis on transforme ce problème de différences borné en un problème de différences P_i selon la question précédente. □

On appelle "problème de différences généralisé" toute formule sans quantificateurs sur Σ^+.

Question 2.8. *Etant donné un algorithme pour décider si un graphe avec poids possède un cycle négatif, proposez un algorithme (à décrire en Français) pour décider si un problème de différences généralisé est $\mathcal{D}iff^+$-satisfiable.*

Solution : On met le problème sous forme normale disjonctive, ce qui préserve la satisfiabilité ; on calcule l'ensemble des problèmes de différences décrit à la question précédente ; pour chacun de ces problèmes, P, on calcule le graphe $\mathcal{G}_P$ et on appelle l'algorithme pour décider si $\mathcal{G}_P$ a un cycle négatif. Si pour l'un des problèmes la réponse est non, alors on répond que le problème généralisé est $\mathcal{D}\text{iff}^+$-satisfiable ; si pour tous les problèmes la réponse est oui, alors on répond que le problème généralisé n'est pas $\mathcal{D}\text{iff}^+$-satisfiable.

Bien sûr, on fait cela de manière paresseuse : c'est ce qu'on appelle *SAT modulo Difference Logic*. □

3 Machines à 2 compteurs

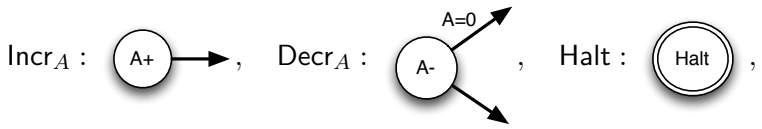
Une machine à 2 compteurs (M2C) possède deux compteurs A et B à valeurs dans $\mathbb{N}$ et un nombre fini d'états. Leur définition est similaire à la présentation des machines à k compteurs présentée dans les transparents du cours 5. On utilise une paire (a, b) pour représenter la valeur des compteurs à un instant donné, a (resp. b) étant la valeur de A (resp. B). Les instructions possibles d'une M2C sont :

- $\mathsf{Incr}_A(j)$: (a, b) devient $(a + 1, b)$ et on saute à l'état j ;
- $\mathsf{Decr}_A(j, k)$: si $a = 0$, on saute à l'état j ; sinon, (a, b) devient $(a - 1, b)$ et on saute à l'état k ;
- Incr_B et Decr_B définies de façon analogue ;
- Halt : le calcul s'arrête, le résultat est dans le compteur A.

Un état particulier est appelé état initial, l'entrée est alors dans le compteur A, le compteur B contenant généralement la valeur initiale 0.

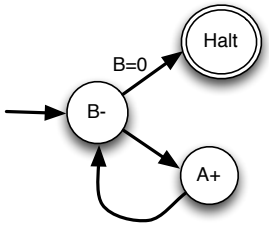
Une M2C *calcule* une fonction totale F de $\mathbb{N}$ dans $\mathbb{N}$ si pour tout $n \in \mathbb{N}$, l'exécution de la machine sur les compteurs $(n, 0)$ atteint un état Halt avec $(F(n), 0)$ dans ses compteurs.

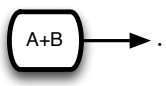
On utilisera la représentation graphique suivante :



ainsi que les graphes analogues pour Incr_B et Decr_B ; et on indiquera par une flèche sans origine l'état initial.

Par exemple, la machine suivante effectue l'opération $(a, b) \mapsto (a + b, 0)$:



On l'appellera A+B et on pourra la réutiliser comme sous-routine sous la forme 

.

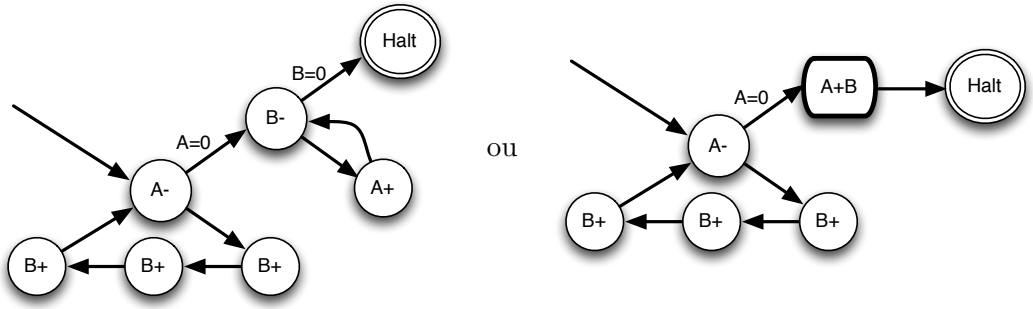
Dans une telle utilisation, l'état Halt correspond à une *branche de sortie* qui pointe sur l'état suivant du calcul.

Le cours montre que les M2C peuvent simuler toute machine de Turing. L'objet de cet exercice est d'approfondir la question de la puissance de calcul de ces machines, et de montrer le résultat, paradoxal au premier abord, que ces machines ne peuvent pas calculer la fonction $N \mapsto 2^N$.

3.1 Premières machines

Question 3.1. *Écrire une M2C* Mult_3 *effectuant l'opération* $(a, 0) \mapsto (3a, 0)$.

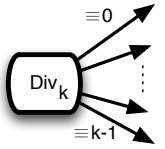
On admet pour la suite que pour tout $k \in \mathbb{N}$, on peut ainsi écrire une M2C Mult_k.
Solution : La machine commence par mettre $3a$ dans B puis échange B et A. De manière équivalente, cette dernière opération est un A+B (version de droite).



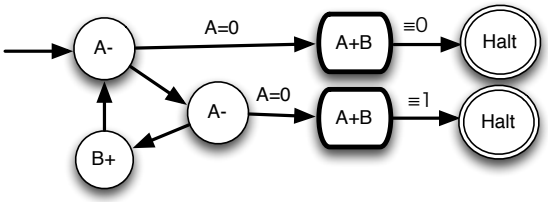
□

Question 3.2. *Écrire une M2C* Div_2*, qui prend en entrée* $(a, 0)$*, possède deux états* Halt *et donc deux branches de sortie selon que* a *est pair ou impair, et termine avec* $(\lfloor a/2 \rfloor, 0)$ *dans ses compteurs.*

On admet pour la suite que pour tout $k \in \mathbb{N}^\star$, on peut ainsi écrire une M2C Div_k, avec k branches de sortie selon la valeur de a mod k. On notera cette machine



Solution : Cette machine soustrait 1 à a k fois (ici $k = 2$), chaque branche $= 0$ de cette phase correspondant à une classe modulo k. Ensuite, on ajoute 1 au quotient b. En sortie d'une des branches $A = 0$, il n'y a plus qu'à échanger A et B, ce qui est effectué par A+B.



□

3.2 Machines à 1 compteur puissant (M1CP)

On considère maintenant des machines à un seul compteur, A, à valeur dans $\mathbb{N}$, avec un nombre fini d'états, et dont les instructions sont

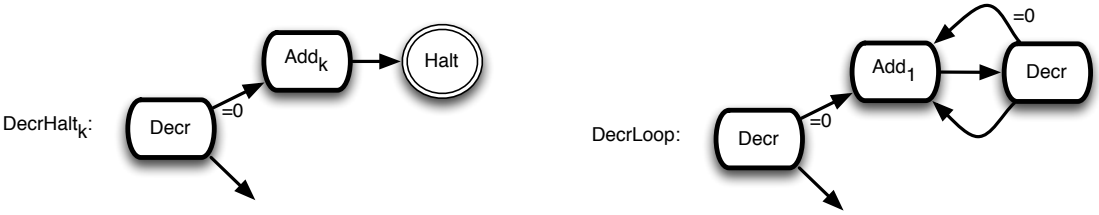
1. Add_k, qui ajoute k au compteur, k étant un entier positif fixé ;
2. Mul_k, Div_k, comme ci-dessus ;
3. Decr, Halt comme pour les machines à 2 compteurs.

Une telle machine *calcule* une fonction totale F de $\mathbb{N}$ dans $\mathbb{N}$ si pour tout $n \in \mathbb{N}$, l'exécution de la machine sur le compteur initialisé à n atteint un état Halt avec $F(n)$ dans le compteur.

Les questions de la section précédente montrent que ces machines peuvent être simulées par des machines à 2 compteurs. La limitation à un seul compteur facilite l'analyse du fonctionnement de ces machines et en particulier des branches $A = 0$ de leurs instructions Decr. Pour voir cela, on peut également ajouter aux machines à 1 compteur :

— pour tout entier k, une instruction $\mathsf{DecrHalt}_k$, qui décrémente le compteur si sa valeur est non-nulle, et s'arrête avec k dans le compteur sinon.
— une instruction $\mathsf{DecrLoop}$, qui décrémente le compteur si sa valeur est non-nulle, et déclenche une boucle infinie sinon.

Ces instructions peuvent s'encoder avec Add_k, Decr, et Halt de la manière suivante :



Question 3.3. *Montrez que, réciproquement, toute machine à 1 compteur peut être simulée par une machine à 1 compteur qui utilise $\mathsf{DecrHalt}_k$ et $\mathsf{DecrLoop}$ mais qui n'utilise pas d'instructions Decr.*

Solution : La machine n'ayant qu'un compteur, les branches $A = 0$ des instructions Decr sont déterminées indépendamment de la valeur du compteur au début du calcul. La suite du calcul peut soit ne pas terminer, donc entrer dans une boucle infinie, soit terminer, mais avec une valeur k qui ne dépend que de la machine, pas de son entrée. Il suffit donc de remplacer les Decr par des $\mathsf{DecrLoop}$ dans le premier cas et par des $\mathsf{DecrHalt}_k$ dans le second. □

Pour une telle machine $\mathcal{M}$, on définit $S_{\mathcal{M}}$ comme l'ensemble des entiers k de ses instructions $\mathsf{DecrHalt}_k$.

Question 3.4. *Soit $\mathcal{M}$ une telle machine qui calcule une fonction totale $N \mapsto F(N)$. En suivant le chemin effectué lors d'un calcul, montrer que pour tout N vérifiant $F(N) \notin S_{\mathcal{M}}$, il existe deux entiers M et D strictement positifs tels que*

$$F(N + pD) = F(N) + pM, \qquad \forall p \in \mathbb{N}.$$

Solution : Pour un N tel que $F(N) \notin S_{\mathcal{M}}$, le calcul utilise donc une suite d'instruction $I_1, \dots, I_n$ où I_i est de l'un des types

$$\mathsf{Add}_k, \mathsf{Mul}_k, \mathsf{Div}_k, \mathsf{DecrHalt}_k, \mathsf{DecrLoop},$$

ces deux derniers sans prendre la branche $= 0$, et $I_n = \mathsf{Halt}$. On pose $m_i = 1$ sauf si $I_i = \mathsf{Mul}_k$, auquel cas on pose $m_i = k$. De même on pose $d_i = 1$ sauf si $I_i = \mathsf{Div}_k$, auquel cas on pose $d_i = k$.

On choisit alors $D = d_1 \cdots d_n$. En notant F_i^p la valeur du compteur en sortie de l'instruction I_i, initialisé avec $F_0^p = N + pD$, on montre par récurrence que $F_i^p = F_i^0 + pm_1 \cdots m_i D/(d_1 \cdots d_i)$. Pour $i = 0$ la propriété est vraie. Ensuite, on la suppose vraie jusqu'à l'instruction $i-1$. Lorsque $I_i = \mathsf{Add}_k$, la propriété est claire. Il en va de même pour les instructions $\mathsf{DecrHalt}_k$ et $\mathsf{DecrLoop}$, puisque $F_i^p \geq F_i^0$: la branche $= 0$ n'est pas prise. Si $I_i = \mathsf{Mul}_k$, alors $m_i = k$, $d_i = 1$ et la propriété découle de

$$m_i(F_{i-1}^0 + pm_1 \cdots m_{i-1} D/(d_1 \cdots d_{i-1}) = F_i^0 + pm_1 \cdots m_i D/(d_1 \cdots d_i).$$

De même, si $I_i = \mathsf{Div}_k$, alors $d_i = k$, $m_i = 1$ et

$$F_{i-1}^0 + pm_1 \cdots m_{i-1} D/(d_1 \cdots d_{i-1}) \equiv F_{i-1}^0 \bmod d_i$$

ce qui montre que le calcul prend la même branche de sortie de l'instruction Div. Enfin, à l'issue de l'opération, on obtient

$$\underbrace{(F_{i-1}^0 - (F_{i-1}^0 \bmod d_i))/d_i}_{F_i^0} + pm_1 \cdots m_i D/(d_1 \cdots d_i),$$

ce qui conclut la récurrence. Le choix $(D, M) = (d_1 \cdots d_n, m_1 \dots m_n)$ répond donc à la question. □

Question 3.5. *En déduire que les M1CPs ne peuvent pas calculer $N \mapsto 2^N$.*

Solution : Par l'absurde : si une machine calcule la fonction, alors elle est simulée par une M1CP $\mathcal{M}$ préparée comme à la question 3.3. Soit alors N tel que 2^N n'est pas dans l'ensemble $S_{\mathcal{M}}$, et soient M et D les entiers associés à N par la question précédente. Alors pour p suffisamment grand,

$$2^N \times 2^{pD} > 2^N + pM,$$

en contradiction avec la question précédente. □

La suite de l'exercice consiste à montrer que les M1CPs peuvent néanmoins simuler les M2Cs.

3.3 Boucles fortes et normalisation des machines à 2 compteurs

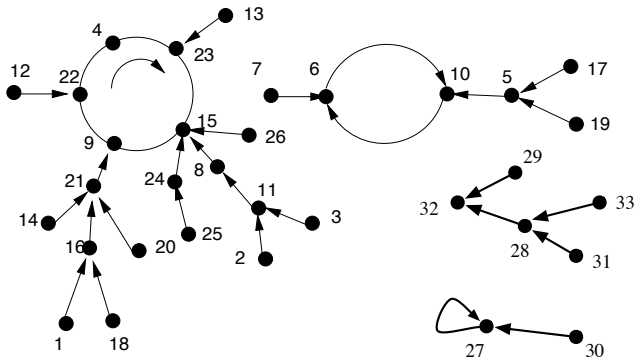


FIGURE 1 – Un graphe orienté à 33 sommets ayant chacun au plus une arête sortante

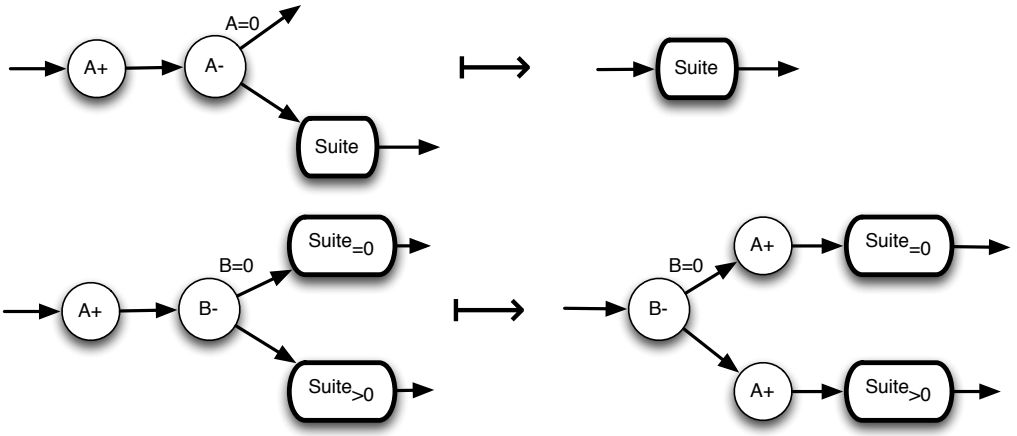
Si on efface temporairement les branches $= 0$ des instructions Decr_A et Decr_B d'une M2C, il reste un graphe orienté où chaque instruction a au plus une arête sortante. Dans un tel graphe, suivre les flèches à partir d'un sommet arbitraire mène soit à un sommet sans arête sortante (Halt), soit à un sommet déjà visité. Un exemple de tel graphe est présenté en Figure 1. Tout graphe de ce type se décompose donc en un ensemble de cycles orientés (éventuellement réduits à un sommet), sur lesquels viennent pointer des arbres. Les arêtes formant ces cycles sont appelées *boucles fortes* de la machine, et les sommets de ces cycles sur lesquels pointent des arêtes autres que celles du cycle sont appelés *sommets d'entrée* de ces boucles fortes. Dans la suite, on raisonne sur l'ensemble de la machine, cette construction ayant permis d'identifier ses boucles fortes et leurs sommets d'entrée.

Question 3.6. *Montrer qu'on peut simuler toute M2C par une M2C où les boucles fortes n'ont qu'un sommet d'entrée.*

Solution : Chaque boucle forte est simplement dupliquée autant de fois qu'elle a de sommets d'entrée, et on choisit une copie distincte pour chaque arête menant à un sommet d'entrée. En ce qui concerne les autres arêtes entrantes (qui ne correspondent pas à des sommets d'entrée parce qu'elles faisaient partie des arêtes initialement effacées), il suffit de les laisser pointer sur l'une des copies. □

Question 3.7. *Montrer qu'on peut récrire toute M2C préparée comme à la question précédente de sorte que dans le parcours des boucles fortes à partir de leur sommet d'entrée, les instructions* Decr *précèdent les instructions* Incr.

Solution : On applique tant que c'est nécessaire les transformations ci-dessous qui échangent un Incr et un Decr :



Chaque application d'une telle transformation fait décroître le nombre de Decr précédé par un Incr, d'où la terminaison de cette construction. □

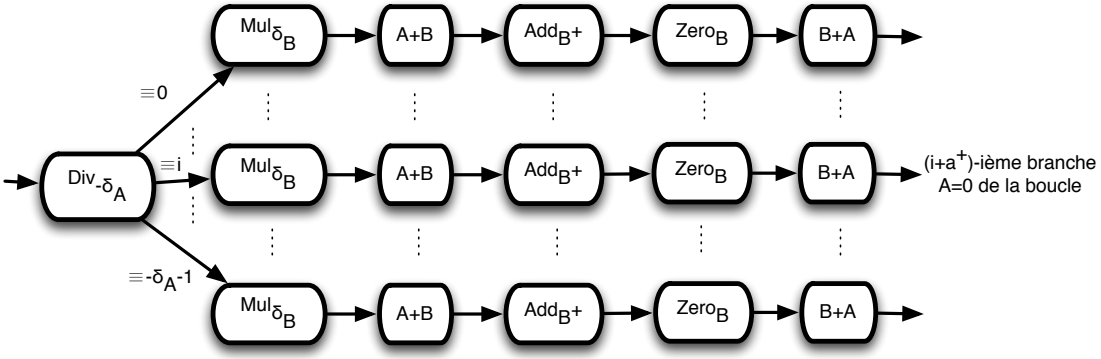
Question 3.8. *Soit une machine qui contient une boucle forte préparée comme à la question précédente. Soit a^+ le nombre de Incr_A de la boucle, b^+ le nombre de Incr_B, a^- le nombre de Decr_A et b^- le nombre de Decr_B. Écrire une machine qui simule cette boucle et telle que, à tout moment d'une exécution sur une entrée (a, b),*

- *soit la valeur du compteur B est inférieure à $\max(b, b^+ - b^-)$;*
- *soit la valeur du compteur A est 0.*

Dans cette machine, on pourra utiliser des sous-routines Div_k, Mul_k, A+B, et à l'intérieur de ces sous-routines, la valeur du compteur B pourra devenir arbitrairement grande.

Solution : Soit (a, b) les valeurs des compteurs A et B avant le premier Incr de la boucle forte préparée, soient $\delta_A = a^+ - a^-$ et $\delta_B = b^+ - b^-$. Trois cas peuvent se produire selon les signes de δ_A et δ_B.

- Si $\delta_B < 0$ alors b reste toujours inférieur à b_0 ;
- Si $\delta_B \geq 0$ et $\delta_A \geq 0$, alors le calcul ne termine jamais et on peut remplacer la suite de la boucle forte par une boucle infinie (par exemple un Incr_A suivi d'un Decr_A qui repointe sur lui) ;
- Si $\delta_B \geq 0$ et $\delta_A < 0$, la boucle est parcourue $\lfloor a/(-\delta_A) \rfloor$ fois et B y est augmenté à chaque fois de δ_B. Au dernier passage par ce point de la boucle, les valeurs des compteurs sont donc $(a \bmod (-\delta_A), b + \lfloor a/(-\delta_A) \rfloor \delta_B)$. Les valeurs atteintes au début de la boucle suivante sont donc $(a \bmod (-\delta_A) + a^+, b + \lfloor a/(-\delta_A) \rfloor \delta_B + b^+)$. Pour éviter d'augmenter B au-delà de la valeur souhaitée, on stocke ces valeurs de B dans A jusqu'à la fin de la boucle, pour les y remettre ensuite. La fin de la boucle devient donc



où Zero_B est une petite boucle qui met B à 0 et B+A effectue le même calcul que A+B, avec les rôles de A et B inversés. Seule cette dernière opération ne respecte pas la

contrainte de borne de B ; elle est effectuée lorsque $A = 0$.

□

On dira qu'une M2C est *normalisée* si elle vérifie les trois propriétés spécifiées aux questions 3.6, 3.7 et 3.8.

3.4 Simulations

On introduit enfin une troisième sorte de machines : des machines à 1 compteur et 1 compteur borné (M1.5C), toujours avec un nombre fini d'états. Ces machines disposent d'un compteur C contenant un entier naturel arbitraire, et d'un compteur D contenant un entier limité à un intervalle $\{0, \ldots, K\}$. On utilise une paire (c, d) pour représenter la valeur des compteurs à un instant donné, c (resp. d) étant la valeur de C (resp. D). Ces machines disposent en outre d'un drapeau F à valeur dans $\{0, 1\}$. Lorsqu'on les utilisera pour simuler une M2C, F indiquera si C contient la valeur de A (F est à 1) ou celle de B (F est à 0). L'ensemble d'instructions est le suivant :

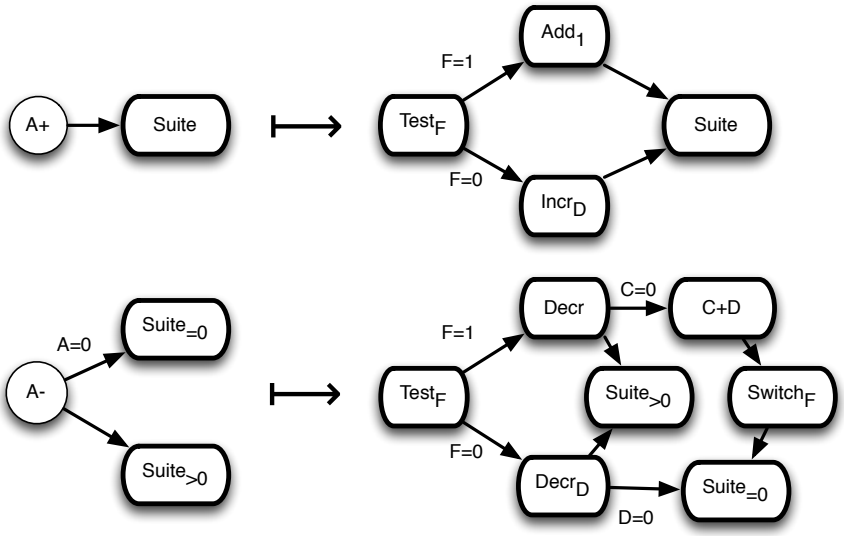
— les instructions de la M1CP, n'agissant que sur C ;
— les instructions Decr_D et Incr_D, cette dernière transformant (c, d) en $(c, d + 1)$ si $d < K$, et échouant si $d = K$;
— une instruction Test_F avec deux sorties selon que F est à 0 ou à 1 ;
— une instruction Switch_F qui change la valeur du drapeau F (de 0 à 1 ou de 1 à 0).

La notion de calcul d'une fonction totale est la même que pour les machines à deux compteurs, le drapeau F étant initialement à 1.

Il est facile de voir qu'une M2C peut simuler une telle machine. L'inverse est plus intéressant.

Question 3.9. *Montrer qu'on peut simuler une M2C normalisée (et donc une M2C quelconque) par une M1.5C. Précisez une borne sur la valeur de K pour cette machine.*

Solution : La M2C est d'abord transformée en une M2C normalisée équivalente par les opérations de la section précédente. Par ailleurs, en dehors des boucles fortes, les transformations suivantes sont effectuées



et leurs analogues pour Incr_B et Decr_B. Dans ces machines, l'instruction C+D utilisée lorsque $c = 0$ est la machine A+B du début de l'exercice, en remplaçant A par C et B par D. Elle n'utilise que des Incr_C et Decr_D.

On obtient le résultat souhaité en choisissant pour K le nombre d'états de la machine normalisée. En effet, initialement le compteur B vaut 0 et la propriété de borne est satisfaite. En sortie de chaque branche = 0 ou de chaque boucle forte, le compteur D vaut 0. En dehors des boucles

fortes, il ne peut se produire plus de K incréments depuis la dernière remise à 0. En ce qui concerne les boucles fortes, dans la transformation de la question 3.8, l'instruction A+B devient C+D (qui n'utilise que des Incr_C et Decr_D), et le dernier B+A devient Switch_F. Le compteur D n'est donc jamais incrémenté dans cette boucle et la propriété de borne est vérifiée. □

Question 3.10. *Montrer qu'on peut simuler une M1.5C dont le compteur D est borné par K et ayant M états par une M1CP ayant au plus $2M(K+1)$ états.*

Solution : Il suffit de remplacer chaque état par $2(K+1)$ états, un par valeur possible de (F, D). Les instructions Test_F et Switch_F sont exploitées dans la construction des transitions de la nouvelle machine ; les instructions Incr_D et Decr_D deviennent des transitions également. On obtient ainsi une machine équivalente, mais qui n'utilise plus qu'un compteur. □

Question 3.11. *Conclure que les machines à 2 compteurs ne peuvent pas calculer la fonction $N \mapsto 2^N$. Expliquer pourquoi ce résultat n'est pas contradictoire avec le fait que ces machines peuvent simuler des machines de Turing.*

Solution : Les machines à 2 compteurs sont simulées par des M1.5C, qui peuvent elles-mêmes être simulées par des M1CP équivalentes, qui sont donc au moins aussi puissantes. Or on a montré que ces machines ne peuvent calculer la fonction $N \mapsto 2^N$, d'où la conclusion. Il n'y a pas contradiction avec la simulation des machines de Turing, puisque celle-ci utilise des machines dont les entrées sont codées sur des entiers $2^a 3^b 5^c$, et qu'on n'impose rien sur les valeurs que calcule cette machine sur des entiers qui ne sont pas de cette forme. □