École polytechnique — X2011 — INF423 Fondements de l'informatique

Logique, modèles, calculs: PC notée

(corrigé)

L'énoncé est intentionnellement trop long, ceci afin de pouvoir noter en barème débordant et de ne pas trop pénaliser ceux qui bloqueraient sur une question. Les parties sont indépendantes et pourront être traitées dans n'importe quel ordre. Il est possible d'admettre le résultat d'une question pour traiter les questions suivantes.

Nous attacherons de l'importance à la **clarté** de la rédaction et vous saurons gré d'**écrire lisiblement**. Les raisonnements mathématiques du corrigé des parties 1 et 3 peuvent s'exprimer synthétiquement en quelques lignes : il est probable que vous faites fausse route si vous cherchez quelque chose de compliqué.

1 Preuves et décidabilité

Question 1.1. *Supposons que nous sommes sur un langage et un système d'axiomes complet (toute formule close est soit prouvable, soit de contraire prouvable), avec un ensemble d'axiomes décidable (décider si une formule close est un axiome est un problème décidable ; autrement dit il y a une machine de Turing qui, si on lui met en entrée un texte sur un alphabet de symboles mathématiques, termine toujours, et accepte si et seulement si le texte est l'écriture d'une formule close qui est un axiome de ce système). Montrez qu'il existe une fonction calculable qui, au vu d'une formule close T, répond elle est démontrable ou de contraire démontrable.*

Solution : Soit T une formule close. La machine énumère par longueur croissante toutes les preuves bien construites ; il suffit pour cela, pour chaque chaîne, de vérifier si elle constitue une preuve bien formée (utilisation des règles de déduction) à partir des axiomes (il suffit de tester si les axiomes invoqués sont bien dans le système à l'aide de la fonction calculable prévue). Par hypothèse, soit elle finira par trouver une preuve de T soit une preuve de $\neg T$. □

Question 1.2. *Étendez au cas où l'ensemble des axiomes est récursivement énumérable (autrement dit il y a une machine de Turing qui, si on lui met en entrée un texte sur un alphabet de symboles mathématiques, peut terminer* ou ne pas terminer*, et accepte si et seulement si le texte est l'écriture d'une formule close qui est un axiome de ce système).*

Solution : Soit g la fonction récursive totale des entiers naturels dans les formules telle que A est un axiome si et seulement si il existe k tel que $g(k) = A$.

On va énumérer par n croissant toutes les preuves bien formées et ne mettant en jeu que des axiomes A tels qu'il existe k plus petit que n tel que $g(k) = A$. □

Nous fixons la signature à celle de l'arithmétique de Peano (0, successeur noté S, $+$, $-$, $\times$, $\leq$ usuels, avec les notations usuelles telles que $<$). Si n est un entier naturel, nous notons $\overline{n}$ le terme $\underbrace{S \dots S}_{n \text{ fois}} 0$; pour tout modèle de l'arithmétique, on appellera l'interprétation de ce terme « entier standard n ».

Une formule est dite Σ_1^0 si elle commence par des quantificateurs existentiels, puis se poursuit par une formule Σ_0^0. Une formule Σ_0^0 commence éventuellement par des quantificateurs bornés $\forall k < e\ F$ ou $\exists k < e\ F$ où e est une expression arithmétique en les variables libres ou introduites

par les quantificateurs précédents, et se poursuit par une formule sans quantificateurs. Un quantificateur borné $\forall k < e\ F$ (F est une formule) est une notation pour $\forall k\ (k < e \Rightarrow F)$. Ainsi, $\exists n \forall a < n \forall b < n\ n \neq ab$ est une formule Σ_1^0 (qui veut dire « il existe un nombre premier »), mais $\exists n \forall a \forall b\ (n \neq ab \vee a = 1 \vee b = 1)$ (qui veut dire la même chose) n'est pas Σ_1^0.

Question 1.3. *Montrez que quel que soit le modèle $\mathcal{M}$ de l'arithmétique tel que $\mathcal{M} \models x < y$, si y a même interprétation que $\overline{n}$, alors les seules interprétations possibles de x sont celles des entiers naturels standards $m < n$.*

(Nous ne demandons pas de détailler les raisonnements arithmétiques simples : autrement dit, vous pouvez dire sans le justifier à l'aide des axiomes de Peano que, quel que soit le modèle $\mathcal{M}$ considéré, $\mathcal{M} \models \forall x \forall y\ x < y \Rightarrow x + 1 < y + 2$ ou autre formule évidente du même genre.)

Solution : Par récurrence sur n. Pour $n = 0$, le problème est trivial : il n'existe aucun modèle $\mathcal{M} \models x < 0$ et on conclut. Supposons maintenant l'hypothèse vraie pour $n - 1$. $\mathcal{M} \models x \leq \overline{n-1}$ donc $\mathcal{M} \models x = \overline{n-1} \vee x < \overline{n-1}$ par les axiomes de l'arithmétique. Le premier cas du $\vee$ permet de conclure directement, et le second pas permet d'appliquer l'hypothèse de récurrence. □

Question 1.4. *Montrez que l'interprétation d'une formule Σ_0^0 close est la même dans tout modèle.*

Solution : Par récurrence sur le nombre de quantificateurs : une telle formule se simplifie en une expression constante.

Le premier quantificateur est de la forme $\forall k < e\ R$ (resp. $\exists k < e\ R$) où e est une expression arithmétique en les variables libres et R est le reste de la formule... mais la formule est close donc e est constante. On peut donc remplacer par une conjonction de e termes $R[0/k] \wedge \cdots \wedge R[e-1/k]$ (resp. $R[0/k] \vee \cdots \vee R[e-1/k]$) à l'aide de la question précédente.

On applique au rang inférieur. Quand il n'y a plus de quantificateurs, on a une expression constante. □

Question 1.5. *Soit F une formule Σ_1^0 close, de la forme $\exists x_1 \ldots \exists x_n\ B$, où tous les quantificateurs de B sont bornés (les variables libres de B sont donc incluses dans les $x_1, \ldots, x_n$). Supposons F indécidable (c'est-à-dire qu'il existe au moins un modèle où F est vraie et un modèle où F est fausse).*

Montrez que B est alors toujours fausse si on remplace toutes ses variables libres par des entier naturels standards.

Solution : Intuitivement, nous utilisons le fait que les entiers naturels standards existent dans tout modèle.

Comme F est indécidable, il existe au moins un modèle $\mathcal{M}$ tel que $\mathcal{M} \models \neg F$, c'est à dire que pour tout $x_1, \ldots, x_n$ pris dans le modèle, l'interprétation de B pour ces $x_1, \ldots, x_n$ est toujours fausse. C'est en particulier le cas pour des x_i entiers naturels standards. □

On admettra qu'il existe une formule H de l'arithmétique de Peano, à un « trou » noté x, de classe Σ_1^0, telle que $H[\overline{n}/x]$ (H dans laquelle on a remplacé x par $\underbrace{S \ldots S}_{n \text{ fois}} 0$) est vraie dans le modèle standard $\mathbb{N}$ si et seulement si la machine de Turing numéro x termine sur l'entrée x.

Question 1.6. *Soit f la fonction de l'ensemble des formules closes de l'arithmétique de Peano vers l'ensemble $\{vrai, faux, indécidable\}$ qui dit si une formule close est démontrable, de contraire démontrable, ou indécidable. Montrez que f n'est pas calculable.*

Solution : Supposons qu'elle le soit. Soit n le numéro d'une machine de Turing. Calculer $f(H[\overline{n}/x])$: si cela répond « vrai », c'est notamment que $H[\overline{n}/x]$ est vraie sur les naturels standard et donc que la machine s'arrête ; si cela répond « faux », c'est notamment que $H[\overline{n}/x]$ est fausse sur les naturels standard et donc que la machine ne s'arrête pas.

Reste le cas où elle répond « indécidable ». Alors par la question précédente, $H[\overline{n}/x]$ est fausse sur les naturels standard, ce qui veut dire que la machine ne s'arrête pas (c'est le cas où la machine ne s'arrête pas, mais on n'arrive pas à le prouver dans l'arithmétique de Peano). On a donc une procédure de décision pour le problème de l'arrêt, ce qui est absurde. □

Nous considérons des preuves avec le langage et le système d'axiomes de l'arithmétique de Peano, un système de règles de déduction fixé (par exemple celui de Hilbert), et une façon fixée d'écrire les énoncés et preuves de ce système à l'aide d'un alphabet fini. Nous nous intéressons à la fonction qui à chaque théorème T de ce système associe la longueur $l(T)$ de la plus courte preuve de T (nous rappelons qu'un théorème est une formule close prouvable dans le système).

Question 1.7. *Montrez qu'il n'y a pas de fonction récursive totale f de l'ensemble des formules closes de l'arithmétique de Peano telle que pour tout théorème T, $l(T) < f(T)$ (il n'y a aucune contrainte sur les valeurs de $f(T)$ ailleurs que sur les théorèmes).*

Solution : Supposons qu'il existe une telle fonction. Considérons l'algorithme suivant : étant donné une formule close T, calculer $n = \max(f(T), f(\neg T))$ et énumérer toutes les preuves de longueur inférieure à n. Si l'on tombe sur une preuve de T, on répond « vrai », sur une de $\neg T$ on répond « faux », et si on les a toutes énumérées on répond « indécidable ». On se ramène aux questions précédentes. □

On en conclut donc qu'il est impossible de borner *a priori* de façon effective (calculable) la longueur de preuve nécessaire pour prouver un théorème.

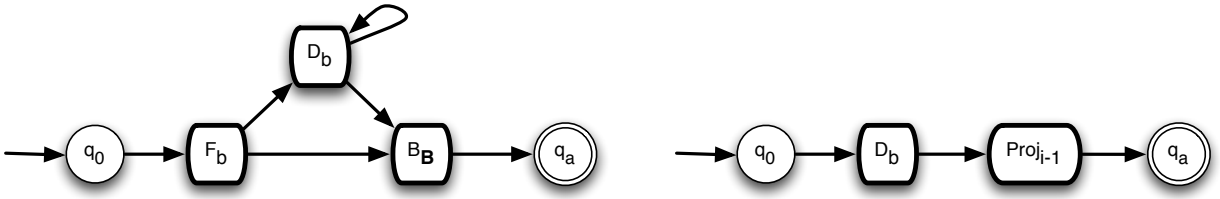
2 Des machines de Turing pour les fonctions récursives primitives

L'objectif de cet exercice est de montrer que les machines de Turing ont une expressivité suffisante pour calculer des fonctions récursives primitives. Les notations de la dernière PC sont conservées, en particulier le k-uplet d'entiers $(n_1, \ldots, n_k)$ sera représenté par le mot $a^{n_1+1}ba^{n_2+1}b\cdots a^{n_k+1}$ sur l'alphabet $\Sigma = \{a, b\}$; l'alphabet de travail Γ contient Σ et $\mathbf{B} \in \Gamma \setminus \Sigma$ est le caractère blanc. On réutilisera une partie des machines qui ont été construites, notamment

- F_ℓ (pour *Forward*) qui avance la tête de lecture sur le ruban jusqu'à la première occurrence de la lettre $\ell \in \Gamma$ et recule alors sur le caractère précédent ;
- B_ℓ (pour *Backward*) qui recule et termine sur le premier caractère suivant l'occurrence de ℓ précédente ;
- D_ℓ et D'_ℓ (pour *Delete*) qui prennent en entrée un mot ℓw où $\ell \in \Sigma$ et w est un mot sur Σ, et s'arrêtent avec le mot w sur leur ruban, décalé d'une lettre, la tête de lecture sur le premier caractère de w pour D_ℓ, sur le premier caractère suivant w pour D'_ℓ (on pourra aussi noter ces machines simplement D et D' si la lettre ℓ n'importe pas) ;
- I_ℓ, I'_ℓ et I''_ℓ (pour *insert*) qui prennent en entrée un mot w sur l'alphabet $\{a, b\}$, et s'arrêtent avec respectivement les mots ℓw, $w\ell$, $w\ell$ sur leurs rubans. La différence entre I'_ℓ et I''_ℓ est que cette dernière laisse sa tête de lecture sur le caractère suivant $w\ell$;
- Copy^m_ℓ qui prend un mot de la forme $w_1\ell w_2$ où w_1 et w_2 sont des mots sur Σ, ℓ et m sont des lettres différentes de $\mathbf{B}$ dans $\Gamma \setminus \Sigma$, et s'arrête avec le mot $w_1\ell w_2 m w_1$ sur son ruban.

Question 2.1. *Écrire une machine de Turing Proj_i qui calcule la fonction prenant en entrée un k-uplet d'entiers et renvoyant le i^e, avec $1 \leq i \leq k$.*

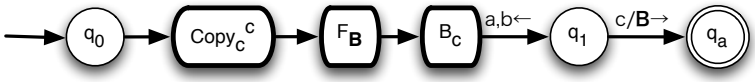
Solution : On distingue le cas de Proj_1 des autres, qui se définissent récursivement :



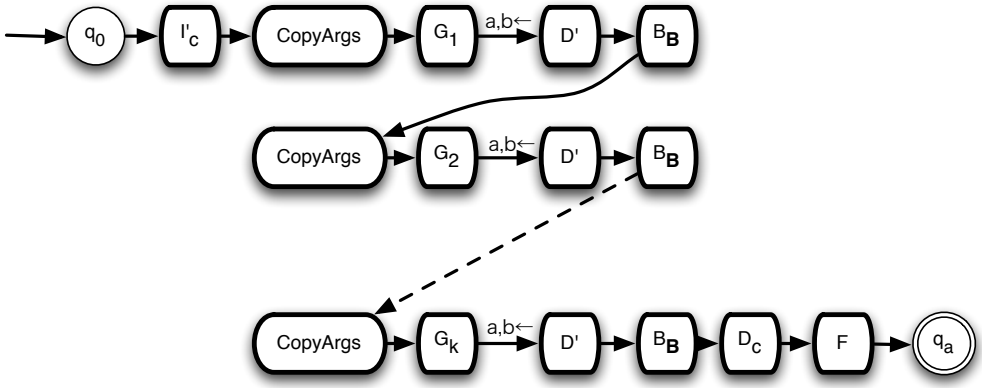
□

Question 2.2. *À partir de k machines $\mathsf{G}_1, \ldots, \mathsf{G}_k$ calculant des fonctions $g_1, \ldots, g_k$ de $\mathbb{N}^n$ dans $\mathbb{N}$ et d'une machine F calculant une fonction f de $\mathbb{N}^k$ dans $\mathbb{N}$, construire une machine calculant l'application $(x_1, \ldots, x_n) \mapsto f(g_1(x_1, \ldots, x_n), \ldots, g_k(x_1, \ldots, x_n))$.*

Solution : L'idée est d'aller calculer les fonctions $g_1, \ldots, g_k$ dans cet ordre, de plus en plus à droite sur le ruban, en insérant un blanc pour que les machines G_i n'effacent pas ce qui a déjà été calculé. On utilise une machine auxiliaire $\mathsf{CopyArgs}$ pour recopier les arguments $(x_1, \ldots, x_n)$ derrière un blanc :



Cette machine prend en entrée un mot de la forme $w_1 c w_2$, où w_1 codera les arguments $x_1, \ldots, x_n$, elle le transforme d'abord en $w_1 c w_2 c w_1$, puis en $w_1 c w_2 \mathbf{B} w_1$ et laisse la tête de lecture positionnée sur le premier caractère du w_1 final.
La composition se résume alors ainsi :

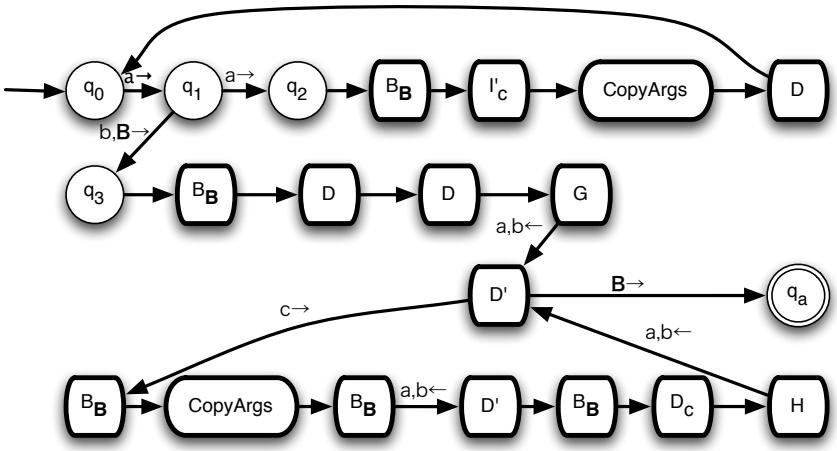


En tout début de calcul, la machine transforme l'argument w en wc, puis appelle la machine précédente pour obtenir $wc\mathbf{B}w$ avant d'appeler G_1 qui termine avec le mot $wc\mathbf{B}g_1(w)$, ensuite la tête de lecture est ramenée sur le blanc, qui est effacé par D' avant de ramener la tête de lecture tout au début du mot $wcg_1(w)$. La deuxième étape commence par transformer ce mot en $wcg_1(w)\mathbf{B}w$ avant de lancer l'appel à G_2. En dernière ligne, on a donc obtenu le mot $wcg_1(w) \cdots g_k(w)$. La machine D_c efface tout le début du mot jusqu'à c, et il ne reste plus qu'à appeler F. □

Question 2.3. *À partir d'une machine G calculant une fonction g de $\mathbb{N}^{n-1}$ dans $\mathbb{N}$ et d'une machine H calculant une fonction h de $\mathbb{N}^{n+1}$ dans $\mathbb{N}$, construire une machine calculant la fonction de $\mathbb{N}^n$ dans $\mathbb{N}$ définie récursivement par*

$$f(0, x_2, \ldots, x_n) = g(x_2, \ldots, x_n),$$
$$f(x_1 + 1, x_2, \ldots, x_n) = h(f(x_1, \ldots, x_n), x_1, \ldots, x_n).$$

Solution :



Pour mémoriser l'état antérieur, cette machine procède comme la machine Comp en allant travailler plus à droite sur le ruban. La première opération consiste à tester si $x_1 = 0$. La première transition lit un premier a, et le branchement est effectué par q_1. La première ligne de la machine traite le cas où $x_1 \neq 0$, en transformant le mot w qui code $(x_1, \ldots, x_n)$ en $wc\mathbf{B}w'$ où w' code $(x_1 - 1, x_2, \ldots, x_n)$ grâce au D final. La machine repart alors en q_0. Lorsque la transition vers q_2 a finalement lieu, le ruban contient le mot

$$w_{x_1} c\mathbf{B} w_{x_1-1} c\mathbf{B} \cdots c\mathbf{B} w_0 c,$$

où w_i code $(i, x_2, \ldots, x_n)$, et la tête de lecture est sur le premier caractère du x_2 de w_0 (ou sur **B** si $n = 1$). La deuxième ligne de la machine calcule alors $g(x_2, \ldots, x_n)$ après avoir effacé $x_1 = 0$.

Ensuite, il faut aller dépiler les valeurs précédentes de x_n. Lorsque la machine arrive à la sortie de la machine D′ de la 3e ligne et que le calcul n'est pas terminé, le ruban contient le mot

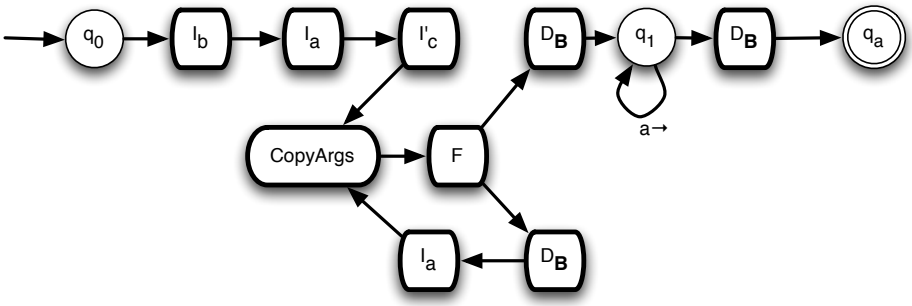
$$w_{x_1} c\mathbf{B} w_{x_1-1} c\mathbf{B} \cdots c\mathbf{B} w_i c v,$$

où v code $f(i-1, x_2, \ldots, x_n)$. La dernière ligne transforme alors la portion $w_i cv$ en $w_i cv w_i$ puis $v w_i$, ce qui code $f(i-1, x_2, \ldots, x_n), x_2, \ldots, x_n$ et permet d'appeler H, de sorte qu'à la sortie de la ligne, $w_i cv$ est remplacé par $f(i, x_2, \ldots, x_n)$. Ceci permet de recommencer tant qu'il reste un c à gauche, c'est-à-dire tant que la pile n'est pas vide. Ensuite, il y a un blanc à gauche et le calcul est terminé. □

Les machines de Turing ont donc au moins l'expressivité des fonctions récursives primitives. Le sujet de la première PC se concluait en mentionnant que ce qui manquait aux fonctions primitives récursives pour avoir toute l'expressivité des machines de Turing était la minimisation non-bornée (correspondant au 'while' des langages de programmation). C'est par contre une opération très facile à implanter sur une machine de Turing.

Question 2.4. *À partir d'une machine* F *calculant une fonction* $f : \mathbb{N}^n \to \{0, 1\}$*, construire une machine, qui prend en entrée le codage de* $(x_2, \ldots, x_n)$ *et s'arrête avec sur son ruban le codage du plus petit* $y \in \mathbb{N}$ *tel que* $f(y, x_2, \ldots, x_n) = 1$ *s'il en existe 1, et boucle sinon.*

Solution :



Cette machine commence par insérer le codage de 0 en tête de ruban, puis elle ajoute un c à la fin de son entrée ainsi modifiée. Ensuite, elle utilise CopyArgs pour recopier le codage de $(0, x_2, \ldots, x_n)$ après un espace au-delà de ce c. La machine F est alors invoquée sur cet argument. Si elle réussit, alors le résultat est effacé, puis la tête de lecture est amenée juste après la fin de y et le reste du ruban est effacé. Si elle échoue, alors le résultat est effacé aussi, puis y est incrémenté (par I_a) et le calcul reprend. □

3 Théorème de complétude

Nous nous proposons ici de démontrer le théorème de complétude de la logique du premier ordre par construction d'un modèle de Herbrand. Le résultat final auquel nous arriverons est :

Théorème 1. *Tout système d'axiomes (formules closes) sur une signature au plus dénombrable soit permet d'obtenir une preuve de l'absurde, soit admet un modèle au plus démontrable.*

Quelques points de vocabulaire et de notations :
- Au plus dénombrable = fini ou dénombrable.
- Preuve de l'absurde : preuve de la formule « faux » ou, de façon équivalente, preuve de F et $\neg F$ pour une formule F quelconque.
- Nous notons $F[t/x]$ (resp. $\tau[t/x]$) la formule (resp. le terme) où la variable libre x est remplacée par le terme t, avec les précautions usuelles pour éviter les captures de variables (renommage des variables quantifiées).

Nous ne préciserons pas le système de preuves utilisé, mais il vérifiera les propriétés usuelles, dont :
- Si $F_1, \ldots, F_n, G$ sont des formules propositionnelles et que pour toute valuation des variables propositionnelles qui satisfait $F_1, \ldots, F_n$, alors cette valuation satisfait G, alors $F_1, \ldots, F_n \vdash G$ (« le système de preuve permet de déduire G à partir de $F_1, \ldots, F_n$ »).
- De toute formule quantifiée universellement $\forall x\ F$ on peut déduire $F[t/x]$ où t est un terme quelconque.

3.1 Lemmes sur des formules propositionnelles

Avant de nous attaquer à la logique du premier ordre, nous avons besoin de lemmes sur des formules propositionnelles.

Question 3.1. *Démontrez qu'un ensemble E (éventuellement infini) de formules propositionnelles sur un ensemble fini de variables soit est incohérent (et alors il existe une preuve de l'absurde à partir de ces formules), soit admet un modèle (qui satisfait chaque formule de E).*

Solution : Soient $v_1, \ldots, v_n$ les variables propositionnelles. Chaque formule $\phi \in E$ se traduit en une fonction $[\![\phi]\!] : \{0,1\}^n \to \{0,1\}$; deux formules sont équivalentes si elles induisent la même fonction. La conjonction de toutes les formules de E est équivalente à la conjonction d'une formule par classe d'équivalence, qui sont en nombre fini $F_1 \wedge \cdots \wedge F_m$. Soit on a un modèle de $F_1, \ldots, F_m$, soit il n'y en a pas et donc tout modèle de ces formules satisfait la formule « faux ». Alors, par l'hypothèse de l'énoncé sur le système de preuve, on peut démontrer « faux » à partir de $\{F_1, \ldots, F_m\}$ et donc a fortiori à partir de E. □

Question 3.2. *Démontrez qu'un ensemble E (éventuellement infini) de formules propositionnelles sur un ensemble au plus dénombrable de variables soit est incohérent (et alors il existe une preuve de l'absurde à partir de ces formules), soit admet un modèle.*

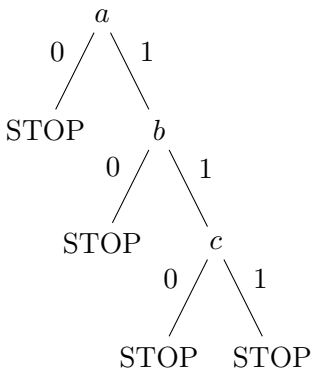
Indice : On ne s'occupera que du cas où l'ensemble de variables est dénombrable, le cas fini ayant été traité à la question précédente. On note E_n l'ensemble des formules de E ne portant que sur les variables $v_1, \ldots, v_n$.

On construit un arbre binaire A des choix successifs de $v_1, v_2, \ldots$, dont chaque nœud de profondeur p est donc associé à un chemin d'accès correspondant aux choix de $v_1, \ldots, v_{p-1}$, la profondeur de la racine étant de 1 (et son chemin d'accès est donc le mot vide) :

- les nœuds internes sont étiquetés par v_k où k est la profondeur dans l'arbre, et leurs arêtes sortantes sont étiquetées par 0 et 1 (signifiant respectivement que le sous-arbre sur lequel l'arête point décrit le cas où $v_k = 0$ ou $v_k = 1$) ;
- on met une feuille STOP à la position étiquetée par le chemin d'accès $b_1, \ldots, b_n$ dès que l'ensemble de formules $E_n[b_1, \ldots, b_n/v_1, \ldots, v_n]$ devient incohérent.

Toute branche de l'arbre est donc soit infinie, so it terminée par une feuille STOP. Nous vous rappelons d'ailleurs le lemme de Koenig : un arbre binaire est infini si et seulement si il a une branche infinie. [1]

Prenons par exemple les variables propositionnelles $a, b, c, \ldots$ et l'ensemble de formules $\{a \implies c, a \implies \neg c, \neg a \implies c, \neg a \implies \neg c, a \wedge b\}$. $E_1 = \emptyset$, $E_2 = \{a \wedge b\}$, $E_3 = E$, et l'arbre est :



Solution : Soit A est fini, soit il a une branche infinie, d'après le lemme de Koenig. Si A est fini, cela veut dire qu'à une certaine profondeur p, on n'a obtenu des feuilles STOP dans toutes les branches, donc que E_p n'a pas de solution quelles que soient les valeurs de $v_1, \ldots, v_p$, autrement dit que E_p est incohérent. On conclut par la question précédente.

Si A est infini, alors il a (au moins) une branche infinie, étiquetée par les choix $b_1, b_2, \ldots$. Montrons que $\mathcal{M} : v_1 \mapsto b_1, v_2 \mapsto b_2, \ldots$ est un modèle de E. Si ce n'est pas un modèle, c'est qu'il y a une formule $\phi \in E$ telle que $\mathcal{M} \not\models \phi$. Cette formule ϕ ne met en jeu qu'un nombre fini de variables propositionnelles, donc elle appartient à un certain E_p ; mais alors, on aurait dû couper cette branche à une profondeur au plus p — contradiction, donc $\mathcal{M} \models E$. □

3.2 Retour au premier ordre

Nous considérons un système d'axiomes sur une signature comportant un ensemble au plus dénombrable de symboles de fonctions et un ensemble un peu plus dénombrable de prédicats.

Question 3.3. *Montrez qu'on peut se ramener au cas d'un système d'axiomes de la forme $\forall x_1 \ldots \forall x_n\ F$ où F est sans quantificateurs, quitte à rajouter des symboles de fonction.*

Solution : On met les axiomes sous forme prénexe (on déplace les quantificateurs en tête de formule, quitte à renommer des variables) et on skolémise : on transforme chaque variable existentiellement quantifiée en une fonction des variables introduites par des quantificateurs universels précédant le quantificateur existentiel.

1. En revanche, si un nœud peut avoir une infinité d'arêtes sortantes, alors ce résultat devient faux.

Par exemple : $\forall x \forall \alpha \exists \eta \ (|x-y| < \eta \Rightarrow |f(x)-f(y)| < \alpha)$ se traduit en $\forall x \forall \alpha \ (|x-y| < \eta(x,\alpha) \Rightarrow |f(x)-f(y)| < \alpha)$ □

Question 3.4. *Montrez qu'on peut se passer du prédicat d'égalité en introduisant un prédicat binaire supplémentaire $\equiv$, des axiomes faisant de $\equiv$ une relation d'équivalence, et des axiomes supplémentaires.*

Solution : On introduit un prédicat $\equiv$, des axiomes de réflexivité, transitivité, et symétrie, et des axiomes de congruence par rapport à tous les autres prédicats et symboles de fonction. Pour tout modèle de ce nouveau système, on obtient un modèle de l'ancien système en quotientant par l'interprétation de $\equiv$; réciproquement tout modèle de l'ancien système est un modèle du nouveau en interprétant $\equiv$ par l'égalité.

Par congruence nous entendons : pour un symbole n-aire de fonction f,

$$\forall x_1 \ldots \forall x_n \forall y_1 \ldots \forall y_n \ x_1 \equiv y_1 \wedge \cdots \wedge x_n \equiv y_n \implies f(x_1, \ldots, x_n) \equiv f(y_1, \ldots, y_n) \qquad (1)$$

et pour un prédicat n-aire P

$$\forall x_1 \ldots \forall x_n \forall y_1 \ldots \forall y_n \ x_1 \equiv y_1 \wedge \cdots \wedge x_n \equiv y_n \implies (P(x_1, \ldots, x_n) \Leftrightarrow P(y_1, \ldots, y_n)) \qquad (2)$$

□

Nous en arrivons à prouver le théorème de complétude pour la signature Σ et les axiomes $\mathcal{A}$ obtenus après les transformations des questions 3.3 et 3.4.

Question 3.5. *Prouvez, en utilisant les résultats des questions précédentes, que soit l'ensemble d'axiomes $\mathcal{A}$ produit une preuve de l'absurde, soit il admet un modèle $\mathcal{M}$, dit de Herbrand, dont l'ensemble de base est l'ensemble $T = \{t_1, t_2, \ldots\}$ (au plus dénombrable) des termes construits sur la signature Σ, les symboles de fonction étant interprétés comme eux-mêmes.*

Solution : On peut alors obtenir un système d'axiomes $\mathcal{A}'$ équivalent par rapport à la structure $\mathcal{M}$ en instanciant chaque axiome de $\mathcal{A}$ par toutes les combinaisons d'éléments de T : $\mathcal{M} \models \mathcal{A}'$ si et seulement si $\mathcal{M} \models \mathcal{A}$. Chaque formule de $\mathcal{A}'$ peut être considérée comme une formule propositionnelle non quantifiée dont les variables libres sont les instanciations des prédicats de Σ par des combinaisons d'éléments de T. On conclut par la question précédente. □