N° 15

M. MARGENSTERN

Introduction à l'analyse constructive selon A.A. Markov

Seminaire donné à Orsay

en fevrier 1972

(Publications Mathématiques d'Orsay)

Introduction à l'analyse constructive selon A. A. Markov

par M. Margenstern

0. Introduction

1. La mathématique constructive, telle que la présente l'école scientifique de A. A. Markov se définit comme l'étude des objets constructifs, fondée sur l'abstraction de la réalité potentielle. Cette abstraction consiste à s'affranchir, dans les considérations mathématiques, de conditions d'ordre matériel relatives à la réalisation concrète des processus représentés abstraitement. Pour donner une idée de ce que recouvre cette abstraction et de ses limites, considérons l'exemple des entiers naturels : on connait un procédé permettant de construire es entiers les uns après les autres ; on peut à partir de là construire des entiers aussi grands que l'on veut, et en aussi grand nombre que l'on veut, mais on ne peut les construire tous ensemble.

2. Rester dans le cadre de la réalisabilité potentielle conduit à rejeter un usage universel de certaines règles de la logique classique (qui ne sont valables de façon générale qu'aux ensembles finis). Dans le développement des théories mathématiques, ce point de départ différent conduit à des résultats parfois très différents de ceux auxquels on parvient dans les théories classiques. Ainsi, nombre de théorèmes de l'analyse réelle classique sont constructivement faux. Ainsi, dans le cadre de cette théorie, on peut construire une suite croissante bornée de rationnels qui ne soit pas convergente. Il n'est pas vrai qu'on puisse trouver un procédé permettant de distinguer un réel d'un autre. De même, on peut construire

une suite de réunions finies de segments à extrémités rationnelles, non vides, emboités et dont l'intersection est vide. Enfin, on peut construire une fonction continue sur le segment fermé $[0,1]$ et qui ne soit pas bornée. Cependant, l'ensemble des réels constructifs n'est pas constructivement dénombrable. On pourrait citer encore de nombreux résultats indiquant les différences entre l'analyse constructive au sens de A. A. Markov et l'analyse classique.

3. En dehors de l'école de Markov, et, historiquement avant elle, il existe d'autres courants constructivistes dans les mathématiques. Leur point de départ commun est l'intuitionnisme, dont le développement se poursuit. Plusieurs des résultats indiqués ci-dessus sont d'ailleurs d'origine intuitionniste. Cependant, le constructivisme de Markov ne dérive pas de la construction intuitionniste : la notion de construction selon A. A. Markov est étroitement liée à la notion d'algorithme, alors que l'intuitionnisme met en oeuvre des notions plus complexes, comme la notion de suite de choix. En outre, la logique constructive au sens de A. A. Markov admet un axiome dont l'énoncé est faux pour les suites de choix. Il en résulte des différences sensibles dans les résultats obtenus en analyse : d'une part l'interprétation des énoncés n'est évidemment pas la même, d'autre part les démonstrations sont souvents différentes. Enfin, il existe des énoncés vrais dans une théorie et faux dans l'autre. Ainsi, en analyse intuitionniste, les fonctions réelles de variable réelle sur un segment sont uniformément continues (encore que cette assertion soit intuitionnistiquement indémontrable pour la notion intuitionniste de réel correspondant à la notion constructive de Markov de nombre réel). Mais on démontre, constructivement, au sens de Markov que toute fonction sur un segment est continue.

4. La théorie des algorithmes s'est constituée à partir du milieu des années trente.

L'intuitionnisme date du premier quart de ce siècle. Les divers courants constructivistes se sont développés à partir de l'intuitionnisme et des fondements de la théorie des algorithmes. L'école de A. A. Markov est postérieure à la seconde guerre mondiale ; les travaux présentés dans cet exposé se situent approximativement pendant les années cinquante, certains datent du début des années soixante.

1. La théorie des algorithmes normaux de Markov

1. Par définition, les algorithmes normaux agissent sur des mots d'un certain alphabet pour les transformer en des mots, éventuellement d'un autre alphabet.

Un alphabet est une collection finie de lettres distinctes deux à deux. On suppose qu'une lettre est entièrement définie par sa représentation graphique et qu'on sait apprécier des différences significatives entre des lettres. On désigne par la lettre Λ l'absence de lettres.

On définit les mots dans un alphabet A par les règles suivantes :

(i) Λ est un mot (appelé le mot vide)

(ii) si α est une lettre de A (on note $\alpha \in A$), alors α est un mot dans A

(iii) si P est un mot dans A et α est une lettre de A, l'assemblage $P\alpha$ est un mot dans A.

Remarques. a) L'ordre des lettres est pertinent dans les mots, et une lettre peut être répétée dans un même mot.

b) L'abstraction de la réalisabilité potentielle implique qu'on peut écrire un mot à la droite d'un autre. Elle se traduit aussi par le fait qu'on peut toujours ajouter une lettre nouvelle à un alphabet.

On peut aisément définir des opérations sur les mots :

- la réunion de deux mots

- le retournement d'un mot

- l'égalité (graphique) entre les mots

- la longueur d'un mot (elle est définie par exemple par :

$$(i)\ \ell(\Lambda) = 0$$

$$(ii)\ \ell(\xi) = 1 \quad (\xi \in A)$$

$$(iii)\ \ell(PQ) = \ell(P) + \ell(Q),\ P \text{ et } Q \text{ mots dans } A).$$

Pour plus de détails, voir [10] (Chapitre I).

2. A partir de ces opérations, on définit la notion d'occurence d'un mot dans un autre.

Soit A un alphabet, P et Q deux mots dans A. On dira que P commence par Q (resp. finit par Q) si P peut se mettre sous la forme $P \equiv QR$ où R est un mot dans A (resp. $P \equiv RQ$) et où le symbole $\equiv$ désigne l'égalité graphique. On dit alors que Q est un début (resp. une fin) de P.

On dira alors que P contient une occurence de Q, si P contient un début dont Q est une fin (resp. une fin dont Q est un début). C'est-à-dire si on peut construire des mots R et S dans A tels que l'on ait $P \equiv RQS$.

On peut maintenant définir ce qu'on appellera par la suite la première occurence d'un mot Q (dans A) dans un autre mot P (dans A) : on appelle première occurence

à gauche (resp. à droite) de Q dans P (on suppose que P contient une occurence de Q) la représentation de P sous la forme RQS possédant la propriété suivante : si R n'est pas le mot vide et $R = \xi_1 \ldots \xi_k$ (ξ_i $1 \leqslant i \leqslant k$ lettres de A) alors aucun des mots $\xi_2 \ldots \xi_k Q, \ldots \xi_k Q$ ne commence par Q (resp. si S n'est pas le mot vide et $S = \eta_1 \ldots \eta_\ell$ alors aucun des mots $Q\eta_1 \ldots \eta_{\ell-1}, \ldots Q\eta_1$ ne finit par Q). On conviendra dans la suite de considérer toujours la première occurence à gauche. On désignera cette première occurence par le mot $R * Q * S$ où $*$ est une lettre ne figurant pas dans A. Par exemple, la première occurence du mot mat dans le mot mathématique sera représentée par *mat*hématique.

3. Nous allons définir maintenant la notion d'algorithme normal. Il nous faut pour cela définir des <u>formules de substitution</u>.

Soit donc A un alphabet, P et Q des mots dans A, $\longrightarrow$ et $\cdot$ des lettres ne figurant pas dans A. Alors les mots $P \longrightarrow Q$ et $P \longrightarrow \cdot Q$ sont appelés formules de substitution dans A. P est appelé <u>partie gauche</u> de la formule, Q est appelé <u>partie droite</u>. Une formule de la forme $P \longrightarrow Q$ est dite <u>simple</u>, une formule de la forme $P \longrightarrow \cdot Q$ est dite <u>concluante</u>.

Considérons une formule de substitution $P \longrightarrow \gamma Q$ (γ désigne Λ ou $\cdot$) et un mot R dans A. On dira que la formule $P \longrightarrow \gamma Q$ s'applique à R si R contient une occurence de P. Si la formule s'applique à R, considérons $U * P * V$ la première occurence de P dans R. Le <u>résultat de l'application</u> de la formule

$P \longrightarrow \gamma Q$ au mot R est alors le mot UQV.

Un algorithme normal dans A est constitué par la donnée d'une suite finie de formules de substitution dans A appelée schème de l'algorithme, et d'une prescription précise définissant le processus d'application de l'algorithme considéré à un mot dans A que l'on peut énoncer ainsi : soit $F_1, \ldots, F_n$ le schème de l'algorithme (chaque F_i désigne une formule de substitution dans A) et P un mot dans A.

R.1. Le processus commence avec $i = 1$, le mot traité est le mot initialement donné P.

R.2. On regarde si la formule F_i s'applique au mot traité.

R.3. Si oui, on applique F_i au mot traité et on note P_i le résultat.

Si F_i est concluante, le processus s'arrête et le résultat final est le mot P_i.

Si F_i n'est pas concluante, on revient en R.2. avec $i = 1$ et pour mot traité P_i.

R.4. Si F_i ne s'applique pas au mot traité, on regarde si i est le dernier indice des formules entrant dans le schème de l'algorithme.

Si oui le processus s'arrête. Le résultat final est le mot traité à cette étape.

Si non, on revient à R.2. en remplaçant i par $i+1$ et en considérant le même mot comme mot traité.

Il y a donc deux types d'arrêt du processus :

- à une étape donnée, aucune formule ne s'applique au mot traité à cette étape : c'est ce qu'on appelle une rupture naturelle du processus. Si la rupture naturelle a lieu pour le mot initial P, on dit alors que P n'entre pas dans l'algorithme.

- ou bien P entre dans l'algorithme et à une certaine étape une formule concluante s'applique.

On dit que l'algorithme s'applique au mot P si le processus décrit ci-dessus a une fin.

Enfin, on introduit la notion d'algorithme sur un alphabet donné A : c'est un algorithme normal dans un alphabet B qui contient toutes les lettres que contient A.

4. Voici des exemples d'algorithmes normaux :

$$\mathcal{A}_1 : \left\{ \longrightarrow .P \quad \text{où} \quad P \quad \text{est un mot donné dans} \quad A. \right.$$

Si Q est un mot dans A on voit que $\mathcal{A}_1(Q) = PQ$ (le signe $\doteq$ signifie que le processus d'application de $\mathcal{A}_1$ à Q s'arrête et que son résultat final est graphiquement égal à PQ).

L'algorithme $\mathcal{A}_2 : \left\{ \longrightarrow P \right.$ où P est un mot donné dans A est un exemple d'algorithme qui ne s'applique à aucun mot dans A.

L'algorithme $$\mathcal{A}_3 : \begin{cases} \alpha\xi \longrightarrow \xi\alpha & (\xi \in A) \\ \alpha \longrightarrow .P & \text{où } P \text{ est un mot donné dans } A \\ \longrightarrow \alpha & \text{et } \alpha \notin A \end{cases}$$

($\alpha\xi \longrightarrow \xi\alpha \quad (\xi \in A)$ est une expression condensée du schème :

$$\begin{cases} \alpha\xi_1 \longrightarrow \xi_1\alpha \\ \cdots\cdots\cdots \\ \alpha\xi_n \longrightarrow \xi_n\alpha \end{cases} \qquad \text{où} \quad A = \{\xi_1, \ldots, \xi_n\}$$

consiste à écrire le mot P à la droite de tout mot dans A.

Algorithmes correspondant aux opérations arithmétiques :

On définit les entiers naturels comme étant les mots suivants de l'alphabet $\{0,1\}$:

(i) 0 est un entier

(ii) si N est un entier, N| est un entier.

On notera $0|^n$ pour $0\underbrace{|\ldots|}_{n \text{ fois}}$ $(n \geqslant 1)$ et on notera les entiers 0, 0|, ..., 0|||||, ... par : 0, 1, ..., 5,... (on peut donner un algorithme de conversion des entiers dans le système décimal).

L'addition est donnée par l'algorithme $\dotplus$: $\{*0 \longrightarrow .$ (on note $0|^n * 0|^m$ le couple (n,m)).

La multiplication, par l'algorithme :

$$\divideontimes \quad \begin{cases} b| \longrightarrow |b \\ a| \longrightarrow |ba \\ a \longrightarrow 0 \\ 0*0| \longrightarrow 0*0 \\ 0*0 \longrightarrow 0 \\ 0|* \longrightarrow \\ |*0 \longrightarrow *0a \\ b \longrightarrow |. \end{cases}$$

Cet algorithme donne le produit d'un nombre quelconque d'entiers :

$$\divideontimes(0|^n * 0|^m) \doteq 0|^{n \times m}$$

$$\divideontimes(0|^{n_1} * 0|^{n_2} * \ldots * 0|^{n_p} = 0|^{n_1 \times n_2 \times \ldots \times n_p}.$$

La soustraction entière, $\left(a \dot{-} b = \begin{cases} a-b & \text{si } a \geqslant b \\ 0 & \text{si } a < b \end{cases}\right)$, par

$$\overset{\circ}{\sqcup} : \begin{cases} |* 0| \longrightarrow * 0 \\ 0 * 0| \longrightarrow 0 * 0 \\ 0 * 0 \longrightarrow 0 \\ | * 0 \longrightarrow | \end{cases}$$

La division, par :

$$\overset{\circ}{0} : \begin{cases} a| \longrightarrow |a \\ |* 0| \longrightarrow * 0\ a \\ 0 * 0a \longrightarrow 0|aba \\ abaa \longrightarrow |aba \\ abab \longrightarrow *b \\ ba \longrightarrow |b \\ 0a \longrightarrow 0|b \\ 0 * 0| \longrightarrow 0 * 0 \\ b \longrightarrow | \ . \end{cases}$$

5. On définit un certain nombre d'opérations sur les algorithmes normaux dont les résultats sont toujours des algorithmes normaux.

5.1. L'extension

Soit $\mathbb{A}$ un algorithme normal dans A et B une extension de A, c'est-à-dire un alphabet contenant A.

On cherche un algorithme normal $\mathbb{B}$ dans B pour lequel

$$\mathbb{A}(P) \simeq \mathbb{B}(P) \qquad (P \text{ mot dans } A)$$

($\simeq$ signifie : si l'un des deux termes a un sens, c. à d. si l'algorithme envisagé est applicable à P, l'autre a également un sens et les deux termes sont alors graphiquement égaux).

Deux solutions sont possibles :

On considère comme algorithme $\mathbb{B}$, l'algorithme normal dans B ayant même schème que $\mathbb{A}$. C'est ce qu'on appelle l'extension naturelle. Il est alors clair que si P est un mot dans $B \setminus A$ (les lettres de B qui ne sont pas lettres de A), alors $\mathbb{B}(P) \doteq P$ (on peut voir qu'alors P n'entre pas dans $\mathbb{B}$).

On peut aussi vouloir que $\mathbb{B}$ ne s'applique qu'aux mots dans A auxquels l'algorithme $\mathbb{A}$ lui-même s'applique. Une solution consiste à prendre l'algorithme $\mathbb{B}$ de schème :

$$\begin{cases} \mathbb{A} & \text{où } \mathbb{A} \text{ symbolise le schème de } \mathbb{A}. \\ \xi \longrightarrow \xi \quad (\xi \in B \setminus A) \end{cases}$$

C'est ce qu'on appelle l'extension formelle de $\mathbb{A}$ à B.

5.2. Fermeture d'un algorithme.

On dit qu'un algorithme $\mathbb{A}$ dans A est fermé si le processus qu'il décrit ne peut s'arrêter qu'au moyen de formules concluantes. On appelle fermeture de $\mathbb{A}$, tout algorithme fermé dans A, $\mathbb{B}$, pour lequel

$$\mathbb{B}(P) \simeq \mathbb{A}(P) \qquad (P \text{ mot dans } A).$$

On peut prendre l'algorithme de schème suivant :

$$\begin{cases} \mathbb{A} \\ \rightarrow. \end{cases}$$

qu'on note $\mathbb{A}^{\bullet}$. Il est clair que tout mot dans A entre dans $\mathbb{A}$.

5.3. Composition des algorithmes normaux.

On démontre le :

Théorème. Soit $\mathcal{A}_i$ un algorithme normal dans A_i $(i = 1, 2)$ et $A = A_1 \cup A_2$ la réunion des alphabets (les lettres de A_1 et celles de A_2 sans répétition). On peut construire un algorithme normal $\mathbb{K}$ sur A tel que pour tout mot P dans A on ait :

$$\mathbb{K}(P) \simeq \mathcal{A}_2^f(\mathcal{A}_i^f(P)) \qquad (\text{on note } \mathbb{K} = \mathcal{A}_2 \circ \mathcal{A}_1)$$

où $\mathcal{A}_i^f$ est une extension formelle de $\mathcal{A}_i$ à A.

Preuve. On traduit en termes d'algorithmes normaux la règle suivante :

1. on applique $\mathcal{A}_1$

2. s'il y a un résultat, on lui applique $\mathcal{A}_2$.

Un schème possible de $\mathbb{K}$ est le suivant :

$$\left\{\begin{array}{ll}
\xi a\alpha \longrightarrow a\alpha a\xi & \\
\alpha\xi \longrightarrow a\alpha a\xi & \\
a\xi\eta \longrightarrow a\,\xi\,a\,\eta & \\
a\alpha\beta \longrightarrow \beta & (\xi\circ\eta \in A) \\
a\xi\beta \longrightarrow \beta a\,\xi & \\
\beta a\xi \longrightarrow \xi\beta & \\
\beta \longrightarrow \cdot & \\
{}^{a}\mathcal{A}_{2\,\alpha}^{\beta} & \\
\mathcal{A}_1^{\alpha} &
\end{array}\right.$$

où $\mathcal{A}_1^{\alpha}$ s'obtient à partir du schème d'une extension formelle et fermée de $\mathcal{A}_1$ en y remplaçant tous les points par α (on suppose $\alpha \notin A$), où ${}^{a}\mathcal{A}_{2\alpha}$ s'obtient à partir du schème d'une extension formelle fermée de $\mathcal{A}_2$ dans laquelle : les mots $\xi_1 \ldots \xi_n$ dans A entrant dans les formules de substitution sont remplacés par $a\xi_1 \ldots a\xi_n$, les points sont remplacés par β et la formule $\longrightarrow \beta$ par la formule $\alpha \longrightarrow \alpha\beta$. On suppose naturellement que a et β sont distinctes et ne sont pas des lettres de

$A \cup \{\alpha\}$. ∎

5.4. Réunion d'algorithmes.

Théorème. Soient $\mathbb{A}_1, \ldots, \mathbb{A}_n$ $(n \geqslant 2)$ des algorithmes normaux, A la réunion de leurs alphabets et γ, une lettre non dans A. On peut construire un algorithme $\mathbb{A}$ sur A tel que : pour tout mot P dans $A \cup \{\gamma\}$ on ait :

$$\mathbb{A}(P) \simeq \mathbb{A}_1(P)\gamma \ldots \gamma\mathbb{A}_n(P)$$ (il s'agit des extensions formelles à A).

Ce théorème découle de la

Proposition. Soient $\mathbb{A}_1$ et $\mathbb{A}_2$ des algorithmes normaux et A la réunion de leurs alphabets. On peut construire un algorithme $\mathbb{B}$ sur A tel que :

pour tout mot P dans A :

$$\mathbb{B}(P) \simeq \mathbb{A}_1(P)\mathbb{A}_2(P)$$ (on considère les extensions formelles à A).

Preuve du théorème.

De la proposition, il résulte aisèment qu'on peut construire un algorithme $\mathbb{B}$ tel que :

$$\mathbb{B}(P) \simeq \mathbb{A}_1(P)\mathbb{A}_2(P) \ldots \mathbb{A}_n(P)$$ (P mot dans A).

Posons alors $\mathbb{B}_{2i-1} = \mathbb{A}_i$ pour $i = 1, \ldots, n$

et $\mathbb{B}_{2i}$ l'algorithme de schème :

$$\begin{cases} \xi \longrightarrow & (\xi \in A \cup \{\gamma\}) \\ \longrightarrow \cdot\gamma \end{cases}$$

pour $i = 1, \ldots, n-1$.

Alors $\mathbb{A}(P) \simeq \mathbb{B}_1(P)\mathbb{B}_2(P) \ldots \mathbb{B}_{2n-1}(P)$ car pour tout mot P dans $A \cup \{\gamma\}$

$\mathbb{B}_{2i}(P) = \gamma$. ∎.

Preuve de la proposition.

L'idée est de dédoubler le mot P et de faire agir $\mathbb{A}_1$ sur "l'original" et $\mathbb{A}_2$ sur la "copie".

On construit l'algorithme $\mathbb{D}$ sur A, de schème :

$$\begin{cases} \alpha\, a\, \xi\, \eta \longrightarrow \eta\, \alpha\, a\, \xi \\ \alpha a\, \xi\, a\, \eta \longrightarrow a \eta \alpha a \xi \\ \alpha a\, \xi \longrightarrow a \xi \alpha \\ \alpha\, \xi \longrightarrow \xi \alpha \alpha a \xi \\ \alpha\alpha \longrightarrow \alpha \\ \alpha \longrightarrow \cdot \\ \quad \longrightarrow \alpha \quad . \end{cases} \qquad \begin{array}{l} (\xi, \eta \in A) \\ a \neq \alpha \quad a, \alpha \notin A \end{array}$$

Si on définit ${}^{a}P$ où P est un mot dans A par :

(i) ${}^{a}\Lambda = \Lambda$

(ii) ${}^{a}\xi = a\xi \quad (\xi \in A)$

(iii) ${}^{a}(P\,\xi) = {}^{a}P\, a\, \xi$ P mot dans A.

On voit aisément que $\mathbb{D}$ s'applique à tout mot dans A et que :

$$\mathbb{D}(P) = P\,{}^{a}P.$$

Soit $\mathbb{A}_1'$ l'algorithme dans $A \cup \{a\}$ de schème $\begin{cases} \mathbb{A}_1 \\ a \longrightarrow \cdot a \end{cases}$, $\mathbb{A}_2'$ l'algorithme ${}^{a}\mathbb{A}_2$ dans $A \cup \{a\}$ (cf. 5.3 pour la définition de ${}^{a}\mathbb{A}_2$) et $\mathbb{E}$ l'algorithme dans $A \cup \{a\}$ de schème $\{a \longrightarrow$, alors une solution cherchée est $\mathbb{B} = \mathbb{E} \circ \mathbb{A}_1' \circ \mathbb{A}_2' \circ \mathbb{D}$. ∎

5.5. Ramification des algorithmes.

Théorème. Soient $\mathbb{A}_1$, $\mathbb{A}_2$ et $\mathbb{C}$ des algorithmes normaux, A la réunion de leurs alphabets.

On peut construire un algorithme $\mathbb{D}$ sur A tel que : pour tout mot P dans A, $\mathbb{D}$ n'est applicable qu'aux mots dans A auxquels $\mathbb{C}$ l'est (il s'agit de l'extension formelle de $\mathbb{C}$ à A) et :

$$\mathbb{D}(P) \simeq \mathbb{A}_1(P) \quad \text{si} \quad \mathbb{C}(P) \doteq \Lambda$$

$$\mathbb{D}(P) \simeq \mathbb{A}_2(P) \quad \text{si} \quad \mathbb{C}(P) \neq \Lambda$$

($\neq$ indique que $\mathbb{C}$ est applicable à P, mais que $\mathbb{C}(P)$ n'est pas graphiquement égal au mot vide).

On peut donner un corollaire plus général de ce théorème :

Soient $\mathbb{A}_0, \ldots, \mathbb{A}_n, \mathbb{C}_1, \ldots, \mathbb{C}_n$ $(n > 0)$ des algorithmes normaux. On peut construire un algorithme $\mathbb{D}$ sur la réunion A de leurs alphabets tel que :

$$\mathbb{D}(P) = \begin{cases} \mathbb{A}_0(P) & (P \text{ mot dans } A \text{ et } \mathbb{C}_1(P) = \Lambda, \ldots, \mathbb{C}_n(P) = \Lambda) \\ \mathbb{A}_1(P) & (P \text{ mot dans } A \text{ et } \mathbb{C}_1(P) = \Lambda, \mathbb{C}_2(\ ) = \Lambda, \ldots, \mathbb{C}_n(\ ) = \Lambda) \\ \mathbb{A}_2(P) & (P \text{ mot dans } A \text{ et } \mathbb{C}_2(P) \neq \Lambda, \ldots, \mathbb{C}_n(P) = \Lambda) \\ \ldots\ldots & \\ \mathbb{A}_n(P) & (P \text{ mot dans } A \text{ et } \mathbb{C}_n(P) \neq \Lambda). \end{cases}$$

Preuve. On construit un algorithme $\mathbb{F}$ sur $A \cup \{\alpha\}$ tel que :

$$\mathbb{F}(P) \simeq \alpha \quad \text{si} \quad \mathbb{C}(P) \doteq \Lambda$$
$$\mathbb{F}(P) \simeq \Lambda \quad \text{si} \quad \mathbb{C}(P) \neq \Lambda$$

et ne s'appliquant qu'aux mots P dans A auxquels $\mathbb{C}$ s'applique. On peut procéder ainsi : soit $\mathbb{F}_0$ l'algorithme dans $A \cup \{\alpha, \beta\}$ de schème :

$$\begin{cases} \alpha\xi \longrightarrow \beta \\ \beta\xi \longrightarrow \beta \\ \beta \longrightarrow \cdot \\ \alpha \longrightarrow \cdot\alpha \\ \quad \longrightarrow \alpha \end{cases} \qquad (\xi \in A)$$

On pose alors $\mathbb{F} = \mathbb{F}_o \circ \mathbb{C}$.

D'après le théorème du point 5.4 on peut construire un algorithme $\mathbb{E}$ sur A tel que $\mathbb{E}(P) \simeq \mathbb{F}(P)P$.

Soit alors $\mathbb{H}$ l'algorithme de schème :

$$\begin{cases} a\xi\eta \longrightarrow a\xi a\eta \\ \alpha\xi \longrightarrow \alpha a\xi \\ \beta a\xi \longrightarrow \xi\beta \\ \xi\beta \longrightarrow \beta\xi \\ a\beta \longrightarrow \beta \\ \alpha\beta \longrightarrow \cdot \\ {}^{a}\mathbb{A}_1'^{\beta}_{\alpha} \\ \mathbb{A}_2 \end{cases} \qquad (\xi, \eta \in A)$$

où ${}^{a}\mathbb{A}_1'^{\beta}_{\alpha}$ désigne le schème obtenu à partir de celui de $\mathbb{A}_1'$ en remplaçant les mots dans A, $\xi_1 \cdots \xi_n$, par $a\xi_1 \cdots a\xi_n$, les points par β et les formules de la forme $\longrightarrow \xi_1 \cdots \xi_p$ par $\alpha \longrightarrow \alpha\, \xi_1 \cdots \xi_p$.

On pose alors $\mathbb{D} = \mathbb{H} \circ \mathbb{E}$. □.

5.6. Répétition d'un algorithme

On a deux théorèmes de répétition d'un algorithme donné : un théorème d'itération jusqu'à l'ordre n et un théorème de répétition conditionnelle, c'est-à-dire de répétition jusqu'à ce qu'une condition donnée soit vérifiée.

Théorème 1. Soit $\mathbb{A}$ un algorithme normal dans A. On peut construire un algorithme normal $\mathbb{B}$ sur $A \cup \{0, |, *\}$ tel que :

$$\mathbb{B}(n * P) \fallingdotseq Q$$

si et seulement si P étant un mot dans A et n un entier naturel, on peut trouver des mots $P_o, \ldots, P_n$ dans A tels que :

$$P_o = P \qquad P_i = \mathbb{A}(P_{i-1}) \qquad i = 1, \ldots, n$$

$$P_n = Q$$

Et pour tout mot P dans A, $\mathbb{B}(0 * P) = P$.

Preuve. On prend comme schème de $\mathbb{B}$:

$$\begin{cases} \alpha\xi\eta \longrightarrow \alpha\xi\alpha\eta \\ \beta\alpha\xi \longrightarrow \xi\beta \\ \xi\beta \longrightarrow \beta\xi \\ \alpha\beta \longrightarrow \beta \\ *\beta \longrightarrow * \\ {}^{\alpha}\mathbb{A}_{\alpha}^{\beta} \\ |* \longrightarrow * \alpha \\ 0* \longrightarrow . \end{cases} \qquad \begin{array}{l} (\xi,\eta) \in A \\ \alpha \neq \beta \text{ et } \alpha.\beta \notin A \end{array}$$

où ${}^{\alpha}\mathbb{A}_{\alpha}^{\beta}$ s'obtient à partir du schème de $\mathbb{A}^{\circ}$ en remplaçant les mots P dans A figurant dans ce schème par ${}^{\alpha}P$, en remplaçant les $\cdot$ par β et les formules du type $\longrightarrow {}^{\alpha}P$ par $\alpha \longrightarrow {}^{\alpha}P$. ▭.

Théorème 2. Soient $\mathbb{A}$ et $\mathbb{C}$ des algorithmes normaux, A la réunion de leurs alphabets. On peut construire un algorithme $\mathbb{B}$ sur A tel que :

$\mathbb{B}$ transforme un mot P dans A en un mot Q si et seulement si on peut construire $P_0, \ldots, P_n$ $(n \geqslant 1)$ tels que :

$$P_o = P \qquad P_i = \mathbb{A}(P_{i-1}) \qquad i = 1, \ldots, n$$

$$P_n = Q$$

et

$$\mathbb{C}(P_i) \neq \Lambda \quad \text{pour} \quad 0 \leqslant i < n$$

$$\mathbb{C}(P_n) = \Lambda.$$

Preuve. On considère $\mathbb{F}_1$ l'algorithme construit comme au point 5.5 tel que :

$$\mathbb{F}_1(P) = \alpha \quad \text{si} \quad \mathbb{C}(P) = \Lambda$$
$$\mathbb{F}_1(P) = \beta \quad \text{si} \quad \mathbb{C}(P) \neq \Lambda.$$

On construit $\mathbb{D}$ algorithme sur A tel que

pour tout mot P dans A $\quad \mathbb{D}(P) \simeq \mathbb{F}_1(P)\mathbb{A}(P)$

(on prend l'extension formelle de $\mathbb{A}$ à A).

On peut alors prendre pour algorithme cherché l'algorithme $\mathbb{B}$ de schème :

$$\begin{cases} \alpha \longrightarrow \cdot \\ \beta \longrightarrow \\ \mathbb{D} \end{cases}$$

5.7. Traduction et réduction.

Il s'agit ici de théorèmes permettant le transport de propriétés d'algorithmes sur un alphabet à d'autres alphabets et de faire l'opération inverse de l'extension.

5.7.1. Il nous faut d'abord définir la traduction d'un alphabet dans un autre :

soit $A = B \cup \{\alpha, \beta\}$

$\quad\quad B = B \cup \{\gamma_1, \ldots, \gamma_k\}$

où B est un alphabet éventuellement vide, α et β deux lettres distinctes ne figurant pas dans B, $\gamma_1, \ldots, \gamma_k$ des lettres distinctes deux à deux et ne figurant pas non plus dans B.

On définit la traduction dans A d'un mot dans B par les règles suivantes :

(i) si $\xi \in B$, alors ${}^{\tau}\xi = \xi$

(ii) si $\xi = \gamma_j$, alors ${}^{\tau}\xi = \alpha\beta^j\alpha \qquad 1 \leqslant j \leqslant k$

(iii) ${}^{\tau}\Lambda =$ et $\beta^j = \underbrace{\beta \ldots \beta}_{j \text{ fois}}$

(iv) ${}^{\tau}(P\xi) = {}^{\tau}P\,{}^{\tau}\xi \qquad P$ mot dans B et $\xi \in B$.

Si δ est une lettre ne figurant pas dans $B \cup \{\alpha, \beta\}$, l'algorithme suivant $\mathbf{T}$ de schème :

$$\begin{cases} \delta\,\xi \longrightarrow {}^{\tau}\xi\,\delta & (\xi \in \mathcal{B}) \\ \delta \longrightarrow \cdot \\ \quad \longrightarrow \delta \end{cases}$$

est applicable à tous les mots dans $\mathcal{B}$ et pour tout mot P dans $\mathcal{B}$ on a

$$\mathbb{T}(P) = {}^{\tau}P.$$

On peut définir l'opération inverse.

Désignons par $\mathbb{T}^{-1}$ l'algorithme sur A de schème :

$$\begin{cases} \delta\,\xi \longrightarrow \xi\,\delta & \xi \in B \\ \delta\alpha\,\beta^{j}\alpha \longrightarrow \gamma_j\,\delta & 1 \leqslant j \leqslant k \\ \delta\alpha\alpha \longrightarrow \delta\alpha\alpha \\ \delta\beta \longrightarrow \delta\beta \\ \delta \longrightarrow \circ \\ \quad \longrightarrow \delta. \end{cases}$$

Cet algorithme est applicable à un mot dans A si et seulement si ce dernier est la traduction d'un mot dans $\mathcal{B}$. Si $\mathbb{T}^{-1}$ est applicable à P, mot dans A, on a donc $P = {}^{\tau}Q$ où Q est un mot dans $\mathcal{B}$; alors $\mathbb{T}^{-1}(P) = Q$ (il y a univocité de la traduction).

5.7.2. On définit alors la traduction d'un algorithme dans $\mathcal{B}$ en un algorithme dans A de la façon suivante : si $F_1, \ldots, F_n$ est le schème de $\mathbb{B}$ algorithme dans $\mathcal{B}$, la traduction de $\mathbb{B}$ dans A qu'on notera $\mathbb{A}$, est l'algorithme de schème ${}^{\tau}F_1, \ldots, {}^{\tau}F_n$ où ${}^{\tau}F_1$ est défini de la même façon que ${}^{\tau}P$ pour un mot P dans A en convenant que ${}^{\tau}\longrightarrow = \longrightarrow$ et ${}^{\tau}\cdot = \circ$. On a alors :

Théorème. Soit $\mathbb{B}$ un algorithme normal dans $\mathcal{B}$ et $\mathbb{A}$ sa traduction dans A. On a alors : pour tout mot P dans $\mathcal{B}$ $\quad \mathbb{A}({}^{\tau}(P)) \simeq {}^{\tau}(\mathbb{B}(P))$.

La démonstration est évidente.

Corollaire 1. Soit un algorithme normal $\mathbb{A}$ sur un alphabet A. On peut alors construire un algorithme normal dans $A \cup \{\alpha, \beta\}$, $\mathbb{A}$, (avec $\alpha \neq \beta$ et $\alpha . \beta \notin A$) tel que l'on ait :

pour tout mot P dans A, si $\mathbb{A}$ est applicable à P et $\mathbb{A}(P)$ est dans A, alors $\mathbb{B}$ est applicable à P et $\mathbb{B}(P) = \mathbb{A}(P)$ et réciproquement.

On met l'alphabet de $\mathbb{A}$ sous la forme $A \cup \{\gamma_1, \ldots, \gamma_k\}$ et on applique le théorème.

En fait on peut faire mieux :

Corollaire 2 (Nagorny). Soit $\mathbb{A}$ un algorithme normal sur un alphabet A et α une lettre non dans $\mathbb{A}$. On peut alors construire un algorithme normal $\mathbb{B}$ dans $A \cup \{\alpha\}$ tel que :

pour tout mot P dans A, si $\mathbb{A}$ est applicable à P et si $\mathbb{A}(P)$ est dans A alors $\mathbb{B}(P) \doteq \mathbb{A}(P)$ et réciproquement.

L'idée de la démonstration est de faire jouer le rôle de β à une lettre fixée de A et de traduire toutes les lettres de A de la forme $\alpha \xi_i \alpha$ où ξ_i est une lettre de A. On définit alors une notion correspondante de traduction d'un algorithme voir [11]. Dans cet article, N. M. Nagorny montre qu'on a là le meilleur résultat possible.

6. Nous allons maintenant démontrer le théorème fondamental de la théorie des algorithmes : le théorème de l'algorithme universel.

Si $\mathbb{A}$ est un algorithme dans A, son schème est de la forme

$$\begin{cases} F_1 \\ \vdots \\ F_n \end{cases}$$

où chaque F_i est un mot du type $P \longrightarrow \gamma Q$ ($\gamma = \cdot$ ou $\gamma = \Lambda$). Nous allons "linéariser" ce schème de façon à l'écrire comme un mot dans un alphabet. Comme $\longrightarrow$ et $\cdot$ sont fixés une fois pour toutes pour décrire des algorithmes dans des alphabets quelconques, dans cette transcription nous devrons utiliser d'autres lettres pour désigner $\longrightarrow$ et $\circ$.

$\mathbb{A}$ étant donné, on appellera image de $\mathbb{A}$, notée $\mathbb{A}^{\mathrm{H}}$, le mot $F'_1\beta \ldots F'_n\beta$ où

si $F_i = P_i \longrightarrow Q_i$, alors

$F'_i = P_i \alpha Q_i$

et si $F_i = P_i \longrightarrow \cdot Q_i$ alors

$F'_i = P_i \alpha \gamma Q_i$.

On démontre alors :

Théorème 1. Soit A un alphabet, a, β, γ, δ des lettres distinctes deux à deux ne figurant pas dans A et soit $B = A \cup \{a, \beta, \gamma, \delta\}$.

On peut construire un algorithme normal sur B tel que l'on ait :

pour tout mot P dans A et pour tout algorithme normal dans A, $\mathbb{A}$,

$$\mathbb{U}(\mathbb{A}^{\mathrm{H}} \delta P) \simeq \mathbb{A}(P).$$

L'idée de la démonstration est de "normaliser" les règles R.1 à R.5 données au point 3. Voir par exemple [13].

En utilisant les notions de traduction, on peut traduire le mot $\mathcal{A}^{\mathcal{U}}$ dans $\{0,|\}$. On peut fixer une fois pour toutes les mots $0|0$, $0||0$ et $0|||0$ pour désigner les lettres α, β, γ figurant dans $A^{\mathcal{U}}$. On notera alors $\mathcal{E}\, A\, 3$ cette traduction qu'on appellera code de $\mathcal{A}$ ou transcription du schème de $\mathcal{A}$.

L'alphabet A étant fixé, en utilisant un algorithme de traduction, le théorème de réunion des algorithmes, le théorème de composition, le corollaire du théorème de traduction et le théorème précédent, on peut énoncer :

Théorème 1'. Soit A un alphabet. On peut construire un algorithme $\mathcal{A}$ sur $A \cup \{0,|,\delta\}$ tel que pour tout algorithme $\mathcal{B}$ dans A et tout mot P dans A on ait :

$$\mathcal{A}(\mathcal{E}\,\mathcal{B}\,3\,\delta\,P) \simeq \mathcal{B}(P) \qquad (\delta \notin A).$$

Remarque. Il n'est pas nécessaire ici de supposer que 0 et $|$ n'appartiennent pas à A, car $\mathcal{E}\mathcal{B}3$ est de la forme $P_1 \ldots P_k$ où chaque P_j est de la forme $0\,|^{n_j}\,0$ (puisque les lettres de A sont traduites toutes sous cette forme, la première lettre étant traduite par $0||||0$, etc...).

On démontra alors le :

Théorème 2. On peut construire un algorithme normal $\mathcal{H}$ dans $\{0,|\}$ tel qu'il soit impossible de construire un algorithme $\mathcal{K}$ sur $\{0,|\}$, applicable à tous les mots dans $\{0,|\}$ et transformant en le mot vide ceux et eux seuls de ces mots auxquels $\mathcal{H}$ s'applique (resp. ne s'applique pas).

Preuve. Il est aisé de voir qu'il suffit de montrer le théorème dans le cas où on prend en conclusion la non applicabilité.

Soit $\mathbb{A}$ l'algorithme du théorème 1', avec pour alphabet $A = \{0,|,\varepsilon,\eta\}$ où $\varepsilon \neq \eta$ et ε et η distincts de 0 et $|$. (Cet alphabet permet de considérer les algorithmes sur $\{0,|\}$). A l'aide du théorème de l'union des algorithmes et celui de la composition, on peut construire un algorithme $\mathbb{H}_o$ sur $\{0,|\}$ tel que : pour tout mot P dans $\{0,|\}$ $\mathbb{H}_o(P) \simeq \mathbb{A}(P\ \delta\ P)$.

Soit $\mathbb{H}$ la traduction dans $\{0,|\}$ de l'algorithme $\mathbb{H}_o$ (on applique le théorème de traduction à $A = \Lambda \cup \{0,|\}$ et $B = \Lambda \cup \{0,|,\delta,\sigma_1,\ldots,\sigma_k\}$ d'alphabet de $\mathbb{H}_o$).

Supposons qu'il existe un algorithme $\mathbb{K}$ sur $\{0,|\}$, applicable à tous les mots dans $\{0,|\}$ et tel que $\mathbb{K}(P) \doteq \Lambda$ si et seulement si $\mathbb{H}$ ne s'applique pas à P. Désignons par $\mathfrak{F}$ l'algorithme dans $\{0,|\}$ de schème

$$\begin{cases} 0 \longrightarrow 0 \\ | \longrightarrow | \\ \longrightarrow \circ \end{cases}$$

Soit $\mathbb{L} = \mathfrak{F}_o\mathbb{K}$. Alors $\mathbb{L}$ est un algorithme sur $\{0,|\}$ tel que : $\mathbb{H}$ ne s'applique pas à P si et seulement si $\mathbb{L}(P) \doteq \Lambda$ et si $\mathbb{H}$ s'applique à P, $\mathbb{L}$ ne s'applique pas à P.

Soit $\mathbb{M} = \mathbb{L}_o\mathbb{T}$ où $\mathbb{T}$ est l'algorithme de traduction des mots dans $\{0,|\}$. C'est un algorithme sur $\{0,|\}$ et on peut le supposer dans $\{0,|,\varepsilon,\eta\}$. On a alors que si $\mathbb{H}_o$ ne s'applique pas à P, $\mathbb{H}$ ne s'applique pas à ${}^{\tau}P = \mathbb{T}(P)$ et donc $\mathbb{M}(P) = \Lambda$ et réciproquement. De même, on aura que si $\mathbb{H}_o$ s'applique à P, $\mathbb{H}$ s'applique à ${}^{\tau}P$ et donc $\mathbb{M}$ ne s'applique pas à P.

Considérons $\mathbb{Q} \rightarrow \mathfrak{E}\mathbb{M}3$. Alors : (1) $\mathbb{H}_o(\mathbb{Q}) \simeq \mathbb{M}(\mathbb{Q})$.

Si $\mathbb{H}_o$ s'applique à $\mathbb{Q}$, $\mathbb{M}$ ne s'y applique pas, ce qui est contradictoire avec (1). Si $\mathbb{H}_o$ ne s'applique pas, alors $\mathbb{M}(\mathbb{Q}) \neq 1$ ce qui contredit encore (1). Donc $\mathbb{M}$ ne pouvant exister, et l'existence de $\mathbb{M}$ découlant de celle de $\mathbb{K}$, il est clair que $\mathbb{K}$ ne saurait exister. □.

7. Il faut dire un mot du principe de normalisation qui correspond, dans la théorie des algorithmes à ce qu'on appelle la thèse de Church dans la théorie des fonctions récursives.

La notion d'algorithme normal est une notion mathématique. Celle d'algorithme ne l'est pas. Un algorithme dans $\mathbb{A}$ (cf. [10]) est "une prescription précise, universellement compréhensible, définissant un processus potentiellement réalisable de transformations successives de mots dans $\mathbb{A}$, processus admettant n'importe quel mot dans $\mathbb{A}$ en qualité de mot initial."

Le principe de normalisation dit que pour tout algorithme $\mathfrak{A}$ on peut construire un algorithme normal $\mathbb{A}$ sur $\mathbb{A}$ tel que pour tout mot P dans $\mathbb{A}$ on ait $\mathfrak{A}(P) \simeq \mathbb{A}(P)$.

Dans ce qui suit, on ne fera pas appel au principe de normalisation. C'est pourquoi dans un énoncé où figure le terme d'"algorithme", on peut aussi bien le lire comme une abréviation de l'expression "algorithme normal", que comme une application du principe de normalisation.

Dans le cas d'un théorème énonçant l'impossibilité de construire un algorithme qui puisse résoudre un problème donné, cette distinction a une certaine importance.

En fait, le principe de normalisation est plus qu'une simplification. Il s'appuie sur l'expérience, sur le fait que toutes les théories des algorithmes actuelles sont équivalentes, sur l'impossibilité de construire un algorithme non normalisable par les méthodes constructives actuellement connues (c'est le sens des théorèmes de composition de répétition conditionnelle, etc...). C'est ce que confirme par exemple un article de Nagorny [12] où se trouve développées des généralisations de la notion d'algorithme normal : les algorithmes de type σ, σ' et σ''. Un σ- algorithme est un algorithme dont chaque pas consiste en l'application d'un schème fini de formules de transformation indiquant le "code" du pas suivant ou l'arrêt éventuel du processus, le nombre de ces schèmes étant fini. Dans un σ'-algorithme, le nombre de ces schèmes devient "infini" en ce sens que le pas est commandé par un algorithme normal définissant une suite indexée de schèmes (qui peuvent être transcrits selon le processus de construction de l'image d'un algorithme). Dans un σ''-algorithme, les schèmes eux-mêmes sont "infinis" (c'est-à-dire définis par un algorithme construisant une suite indexée de formules de substitution). Or toutes ces définitions sont équivalentes à celle des algorithmes normaux.

En conclusion, un théorème énonçant l'impossibilité de construire un algorithme réclamant tel problème exclut la possibilité de trouver une méthode <u>générale</u>, de type algorithmique qui permette de résoudre simultanément toute <u>la classe</u> de problèmes posés ;

cela ne veut pas dire que si on prend un problème concret, individuel dans cette classe on ne puisse décider de la possibilité de le résoudre.

§ 2. Quelques éléments de logique constructive.

1. Désignons par $\mathfrak{F}$. une formule exprimant une assertion dont on cherche à démontrer la vérité.

On dispose de deux méthodes pour établir $\mathfrak{F}$. constructivement. La première consiste à démontrer $\mathfrak{F}$. à partir des axiomes de la logique constructive, des règles de déduction qu'elle admet et des résultats déjà établis (par cette méthode ou par l'autre). La seconde méthode consiste en deux étapes. En premier lieu, on examine si $\mathfrak{F}$. contient un problème constructif ce qui consiste à chercher si $\mathfrak{F}$. peut être réduite, selon un processus algorithmique bien défini, en une formule équivalente d'un type particulier ou non, auquel cas on dira que $\mathfrak{F}$. contient ou ne contient pas de problème constructif. Si $\mathfrak{F}$. ne contient pas de tel problème, $\mathfrak{F}$. est équivalente à une formule $\mathfrak{N}$ dite normale. Démontrer $\mathfrak{F}$. se ramène alors à démontrer $\mathfrak{N}$ en utilisant la partie de la logique constructive concernant ce type de formules. On verra qu'il s'agit de tous les axiomes de la logique classique en remplaçant les symboles $\exists$ et $\vee$ respectivement par $\dot{\exists}$ et $\dot{\vee}$ où $\dot{\exists}x\,A(x)$ signifie $\neg\forall x \neg A(x)$ et $(A \dot{\vee} B)$ signifie $\neg(\neg A \,\&\, \neg B)$. Si $\mathfrak{F}$. contient un problème constructif, démontrer $\mathfrak{F}$. consiste à résoudre de problème ce qui consiste à : 1. Construire des objets d'un certain type devant

FACULTÉ DES SCIENCES · BIBLIOTHÈQUE · ORSAY · MATHÉMATIQUE

satisfaire à une condition (celle qu'on met en évidence dans la reformulation de $\mathfrak{S}$ permettant de dégager le problème constructif qu'elle contient) exprimée par une formule normale, 2. démontrer que les objets constructifs ainsi construits satisfont à cette condition, cette démonstration se faisant dans le cadre de la partie de la logique constructive concernant les formules normales.

2. Les formules sont définies par rapport à un alphabet, ou une collection finie d'alphabets appelé alphabet de langage logico-mathématique qu'on utilise et servant à énoncer les assertions, etc..

On définit les formules comme des mots d'un certain type, à partir des formules élémentaires et selon les règles génératives suivantes :

(i) toute formule élémentaire est une formule

(ii) si $A_1, \ldots, A_n$ sont des formules, les mots $(A_1 \& \ldots \& A_n)$ et $(A_1 \vee \ldots \vee A_n)$ sont également des formules

(iii) si A et B sont des formules, les mots $(A \supset B)$ et $\neg A$ sont des formules

(iv) si A et B sont des formules, $x_1, \ldots, x_n$ des variables deux à deux distinctes, les mots $\forall x_1 \ldots x_n\, A$ et $\exists x_1 \ldots x_n\, B$ sont des formules.

Les variables sont des mots d'un type particulier servant à désigner des mots dans un alphabet donné ou des algorithmes sur cet alphabet. Par exemple, si un alphabet A est fixé contenant $\{0, |\}$, si A^{ac} désigne une extension de A permettant de définir les algorithmes sur A et des groupes de mots dans A, fixons t et λ des lettres n'appartenant pas à A^{ac}. Désignons par μ une des lettres t et λ

(toujours la même). On appelle variable du genre μ les mots définis par les règles génératives suivantes :

(i) le mot (μ) est une variable du genre μ.

(ii) si X est une variable du genre μ, le mot $\mathbb{V}_\mu(X)$ est une variable du genre μ où $\mathbb{V}_\mu$ est l'algorithme de schème :

$$\{\quad) \longrightarrow \circ\, \mu).$$

Suivant que μ désigne t ou λ, on appellera les variables correspondantes variables d'objet ou variables fonctorielles. On appellera alors valeurs admissibles d'une variable de genre μ, tout mot de genre μ où : si $\mu = t$ on désigne par mot de genre t, tout mot dans A ; si $\mu = \lambda$ on désigne par mot de genre λ, tout mot de la forme $\langle H \rangle$ où H est un mot dans A et ayant la signification suivante : si H est un mot dans $\{0, \}$ et est le code d'un algorithme sur A, $\langle H \rangle$ désignera cet algorithme, si H n'est pas un mot dans $\{0, \}$ ou, étant dans $\{0, \}$ n'est pas le code d'un algorithme normal sur A, on convient que $\langle H \rangle$ désigne l'algorithme de schème $\{ \longrightarrow$.

Il nous reste à expliciter ce qu'on entend par formule élémentaire. On introduit à cet effet la notion de terme.

On appellera terme correspondant à un alphabet, tout mot ou groupe de mots dans cet alphabet ou tout algorithme sur cet alphabet, toute variable d'objet ou fonctorielle relative à cet alphabet et les mots suivants :

$\langle \theta \rangle$, qui est un terme fonctoriel si θ est un terme d'objet,

$\Phi(R)$ où Φ est un terme fonctoriel et R un système de terme d'objets (si T est un terme, c'est un système de termes et si T_1 et T_2 sont des systèmes de termes, $T_1 \circ T_2$ est un système de termes).

$\varepsilon\Phi\beta$ qui est un terme d'objet si Φ est un terme fonctoriel.

Dans le cas où on considère plusieurs alphabets, qu'on supposera vérifier la condition : si A_i et A_j sont deux alphabets de notre système, $A_i \cup A_j$ figure dans la liste et le dernier alphabet de la liste contient tous les autres, on définit aussi une relation entre termes correspondant à des alphabets différents définie par la relation $A_i \subseteq A_j$ (cf. [3]).

Les formules élémentaires sont alors ainsi définies :

(i) si θ est un terme d'objet, le mot $!\theta$ est une formule élémentaire.

(ii) si θ_1 et θ_2 sont deux termes d'objet, le mot $(\theta_1 \simeq \theta_2)$ est une formule élémentaire.

$!\theta$ se lit : "θ a un sens".

$(\theta_1 \simeq \theta_2)$ se lit : "si l'un des deux termes θ_1 et θ_2 a un sens, l'autre terme a également un sens et les valeurs de ces deux termes sont graphiquement égales".

L'interprétation de ces formules est la suivante :

On ne considère dans cette interprétation que les formules constantes, c. à d. sans variables (si $!\theta$ est une formule contenant une variable, elle énonce une condition portant sur cette variable dont l'interprétation se ramène à l'interprétation indiquée ci-dessous quand on remplace les occurences de la variable par des valeurs admissibles).

$!\theta$ a un sens dans les cas suivants :

a) θ ne contient pas de terme de la forme $\langle H\rangle(R)$. Dans ce cas la valeur de $!\theta$ est le mot que figure θ.

b) θ contient des termes de la forme $\langle H\rangle(R)$, chacun d'eux ayant un sens ; on dit que le terme $\langle H\rangle(R)$ a un sens si l'algorithme $\langle H\rangle$ est applicable à R et si le résultat est un mot dans l'alphabet sur lequel $\langle H\rangle$ opère. (cf. [3] et la remarque p. 247).

Lorsque $!\theta$ a un sens, sa valeur se calcule ainsi : on remplace toute occurence dans θ d'un terme de la forme $\langle H\rangle(R)$ par la valeur de ce dernier qui est le résultat de l'application de $\langle H\rangle$ à R, et toute occurence des termes de la forme $\varepsilon\Phi 3$ par le mot dans $\{0,|\}$ qu'ils représentent.

Dans la suite, on utilisera les abréviations :

$\theta_1 \doteq \theta_2$ pour $(!\theta \,\&\, \theta_1 \simeq \theta_2)$

$\theta_1 \not\doteq \theta_2$ pour $(!\theta_1 \,\&\, \theta_2 \,\&\, \neg(\theta_1 \simeq \theta_2))$.

Lorsqu'une formule contient des variables, on classe les occurences de ces variables en occurences libres et occurences liées.

Définissons une sous-formule d'une formule donnée comme tout mot de cette formule qui est lui-même une formule.

Nous définirons alors les chainons graphiques d'une formule donnée comme les sous-formules de longueur maximale (c. à d. qui ne sont contenues dans aucune autre formule de la formules).

Les occurences libres (resp. liées) d'une variable γ dans une formule F proviennent des occurences libres (resp. liées) de γ dans les chaînons graphiques de F.

Enfin, on posera que : toutes les occurences d'une variable dans une formule élémentaire sont libres ; si F est de la forme $Kx_1 \ldots x_n A$ où K est un des quantificateurs $\forall$ et $\exists$ et γ est une des variables $x_1, \ldots, x_n$, alors toutes les occurences de γ dans F sont liées.

On dira enfin qu'une variable γ figure librement dans F si on peut trouver au moins une occurence libre de γ dans F.

Les variables libres d'une formule sont appelées les paramètres de cette formule.

3. Les formules suivantes (voir par exemple [4]) (et qui sont admises par la logique intuitionniste sont considérées comme vraies :

1. $(A \supset A)$
2. $(A \supset (B \supset A))$
3. $((A \supset (A \supset B)) \supset (A \supset B))$
4. $((A \supset (B \supset C)) \supset (B \supset (A \supset C)))$
5. $((A \supset B) \supset ((B \supset C) \supset (A \supset C)))$
6. $((A \& B) \supset A)$
7. $((A \& B) \supset B)$
8. $((A \supset B) \supset ((A \supset C) \supset (A \supset (B \& C))))$
9. $(A \supset (A \vee B))$
10. $(B \supset (A \vee B))$
11. $((A \supset C) \supset ((B \supset C) \supset ((A \vee B) \supset C)))$
12. $((A \supset B) \supset ((A \supset \neg B) \supset \neg A))$
13. $(\neg A \supset (A \supset B))$
14. $(\forall x\ F(x) \supset F(a))$
15. $(F(a) \supset \exists x\ F(x))$

où A, B et C désignent des formules quelconques ; dans les formules 14 et 15, x

est une variable liée, a est une variable liée, F une formule à un paramètre.

Lorsqu'on a des formules vraies, on établit de nouvelles formules vraies au moyen du schéma :

$$\frac{A(A \supset B)}{B} \qquad \frac{(A \supset F(a))}{(A \supset \forall x\ F(x))} \qquad \frac{(F(a) \supset A)}{(\exists x\ F(x) \supset A)}\ .$$

Dans les deux règles de droite, on suppose que la formule A ne contient aucune occurrence libre de la variable x.

De ces formules, et en utilisant les règles ci-dessus on peut déduire (voir par exemple [5] ou [1]) :

(1) $(\forall x\ (A \& B) \equiv (\forall x\ A \& \forall x\ B))$
(2) $(\exists x\ (A \vee B) \equiv (\exists x\ A \vee \exists x\ B))$
(3) $(\neg \exists x A \equiv \forall x\ \neg A)$
(4) $((A \supset B) \supset (\neg B \supset \neg A))$
(5) $((A \supset \neg B) \equiv (B \supset \neg A))$
(6) $(\neg (A \vee B) \equiv (\neg A \& \neg B))$
(7) $((A \supset \neg B) \equiv \neg (A \& B))$
(8) $(\neg \neg (A \& B) \equiv (\neg \neg A \& \neg \neg B)$
(9) $\neg (A \& \neg A)$.

où $(A \equiv B)$ est une abréviation de $((A \supset B) \& (B \supset A))$.

Les formules (1), (2) et (3) sont encore vraies quand on considère des formules à plusieurs paramètres on a :

(1') $(\forall x_1 \ldots x_n\ (A \& B) \equiv (\forall x_1 \ldots x_n A \& \forall x_1 \ldots x_n B)$
(2') $(\exists x_1 \ldots x_n\ (A \cup B) \equiv (\exists x_1 \circ \ldots x_n A \vee \exists x_1 \ldots x_n\ B))$
(3') $(\neg\ \ni x_1 \circ \ldots x_n\ A \equiv\ \forall x_1 \ldots x_n\ \neg A)$.

A toutes ces formules, on ajoute la formule suivante (qu'on ne peut déduire de la logique intuitionniste, cf. [9]) :

16. $(\forall x (A \vee \neg A) \supset (\neg \neg \exists x\ A \supset \exists x\ A))$

qu'on désignera par la suite par principe de Markov.

Il est équivalent à l'énoncé suivant :

"si le processus d'application de l'algorithme $\mathcal{A}$ au mot donné P n'est pas indéfiniment prolongeable, l'algorithme $\mathcal{A}$ est applicable au mot P."

4. Nous avons dit, au point 1 de ce paragraphe, qu'on dispose d'une autre méthode pour démontrer les formules vraies.

On dit qu'une formule F est <u>normale</u> si elle ne contient aucun des signes $\exists$ ou $\forall$. Toute formule normale ou toute formule du type $\exists x_1 \ldots x_n A$ où A est une formule normale est dite <u>complètement régulière</u>.

Dans [2], N. A. Chanine décrit différents algorithmes normalisables permettant de transformer toute formule en une formule complètement régulière qui par définition lui est équivalente et qui constitue la reformulation constructive de la formule montrant s'il y a ou non un problème constructif à résoudre pour démontrer cette formule.

La construction de cet algorithme est fondée sur les considérations suivantes :

(i) la démonstration d'une formule du type $(A_1 \vee \ldots \vee A_n)$ supposée vraie doit toujours indiquer au moins une des formules $A_1, \ldots, A_n$ exprimant une assertion vraie.

(ii) une formule du type $\exists x A$ n'est vraie que si on peut donner une construction de x pour lequel la formule A' obtenue à partir de A en remplaçant les occurences de x par l'objet construit est vraie.

(iii) dans le même ordre d'idées $\forall x \exists y A$ suppose qu'étant donné chaque x, on a une construction d'un y à partir de x pour lequel A est vrai ; c'est-à-dire, qu'on a une méthode générale, (un algorithme) donnant pour chaque mot P du type de la variable que représente x, un mot Q (du type de la variable représentée par y)

tel que la formule obtenue à partir de A en y remplaçant les occurences de x et y respectivement par P et Q soit vraie.

Nous indiquons seulement ici, sur un exemple, comment fonctionne cet algorithme.

Considérons la formule

(1) $\forall\alpha\ (A \vee B)$ où A et B sont des formules normales à un paramètre, du même type que la variable α.

On considère la sous-formule $(A \vee B)$. L'explicitation de cette formule, selon le principe (i) donné plus haut donne :

$$\exists\delta((<W_2>(\delta) \underset{\circ}{=} \wedge) \& ((\delta \underset{\circ}{=} 0|) \supset A) \& ((\delta \underset{\circ}{=} 0||) \supset B))$$

où $<W_2>$ est l'algorithme de schème :

$$\begin{cases} 0|| \longrightarrow \circ \\ 0| \longrightarrow \cdot \\ \longrightarrow \circ| \end{cases}$$ et δ variable d'objet dans $\{0,|\}$.

La formule (1) devient donc

(2) $$\forall\alpha\,\exists\delta\ ((<W_2>(\delta) \underset{\circ}{=} \wedge) \& ((\delta \underset{\circ}{=} 0|) \supset A) \& ((\delta \underset{\cdot}{=} 0||) \supset B))$$

qui est donc de la forme $\forall\alpha\,\exists\delta C$ où C est une formule normale.

En application du principe (iii), on introduit une variable fonctorielle sur l'alphabet dont α représente les mots et $\forall\alpha\,\exists\delta C$ devient alors :

(3) $$\exists\sigma\forall\alpha((<I_o>(\sigma(\alpha)) \underset{\circ}{=} \wedge)\ F^{\delta}_{\sigma(\alpha)}\,\llcorner C\lrcorner)$$

où I_o est l'algorithme de reconnaissance des mots dans l'alphabet $\{0,|\}$ (c. à d. I_o transforme en le mot vide ceux des mots dans A qui sont des mots dans $\{0,|\}$ et eux seulement, où A est un alphabet donné contenant $\{0,|\}$) et où $F^{\delta}_{\sigma(\alpha)}\,\llcorner C\lrcorner$ repré-

sente la formule (normale) obtenue à partir de C en y remplaçant toutes les occurences de δ par $\sigma(\alpha)$.

L'interprétation constructive de (1) est donc en fin de compte :

(4) $\exists\sigma\forall\alpha((\langle I_o\rangle(\sigma(\alpha)) \stackrel{.}{=} \Lambda) \& (\langle W_2\rangle(\sigma(\alpha)) \stackrel{.}{=} \Lambda) \& ((\sigma(\alpha) = 0|) \supset A) \& ((\sigma(\alpha) \stackrel{.}{=} 0||) \supset B))$.

Etant donné l'importance des formules normales, nous démontrons la :

Proposition. Une formule A est équivalente à une formule normale si et seulement si elle est équivalente à sa double négation (c. à d. $(\neg\neg A \equiv A)$).

Preuve. Supposons que l'on ait

$(A \equiv B)$ où B est une formule normale. Alors

$(\neg\neg A \equiv \neg\neg B)$ qui découle aisément de 3.(4). Comme B est normale, on a donc $(\neg\neg B \equiv B)$ (voir [3]). Et donc $(\neg\neg A \equiv A)$.

Si on a $(\neg\neg A \equiv A)$, on sait, d'après l'interprétation constructive des formules que $(A \equiv \pi\lfloor A\rfloor)$ où π est l'algorithme transformant toute formule en une formule complètement régulière. Si $\pi\lfloor A\rfloor$ est normale, c'est terminé. Sinon, on a alors $\pi\lfloor A\rfloor = \exists x_1 \ldots x_n\, B$ où B est une formule normale. Donc : $(A \equiv \exists x_1 \ldots x_n B)$. Alors de 3.(4) on a

$$(\neg A \equiv \neg\exists\, x_1 \ldots x_n\, B)$$

d'où :

$$(\neg A \equiv \forall x_1 \ldots x_n\, \neg B) \qquad \text{d'après 3.(3)}$$

d'où 3.(4) donne

$$(A \equiv \neg\forall x_1 \ldots x_n\, \neg B)$$

la formule de droite étant normale puisque B l'est. □

5. Sur la base de la notion de formule, on peut introduire la notion constructive d'ensemble.

Comme précédemment, considérons $\mathfrak{A}$ un alphabet, désignons par t et λ les lettres génériques des variables respectivement d'objet et fonctorielles. Soit A une formule définie dans un langage logico-mathématique dont la liste des alphabets comprend $\mathfrak{A}$. Si A est une formule à un paramètre de genre μ (t ou λ) correspondant aux variables associées à $\mathfrak{A}$, on dira également que A est un ensemble de genre μ. Si P est un mot de genre μ on écrira $P \in A$ (on dira "P appartient à l'ensemble A") si P vérifie A c'est-à-dire, si la formule $F_P^\alpha \lfloor A \rfloor$ est vraie (où α désigne le paramètre de la formule A). Si A est un ensemble de genre t, on dira que A est un ensemble de mots dans $\mathfrak{A}$.

Si A et B sont deux ensembles de genre μ et si α et β désigne leurs paramètres (du même genre), on dit que A est contenu dans B, et on notera $A \subseteq B$, si la formule

$$\forall \alpha \ (A \supset F_\alpha^\beta \lfloor B \rfloor)$$

est vraie. L'égalité de A et B $(A = B)$ est définie par $A \subseteq B$ et $B \subseteq A$ c'est-à-dire :

$$\forall \alpha \ (A \equiv F_\alpha^\beta \lfloor B \rfloor).$$

Parmi les différents types possibles d'ensemble on distingue ceux qui sont définis par des formules assez simples faissant intervenir des conditions algorithmiques.

On définit principalement : les ensembles résolubles et les ensembles énumérables.

Un ensemble A de mots dans $\mathfrak{A}$ est dit algorithmiquement résolubles (en abrégé

résoluble) si on peut construire un mot H dans $\{0,|\}$ tel que

$$\forall \alpha \; ! \; \langle H \rangle (\alpha)$$

où α est une variable du genre du paramètre de A, et si B désigne la formule

$$(\langle H \rangle (\beta) \doteq \Lambda)$$

alors A est égal à B.

Un ensemble A de mots dans $\mathfrak{A}$ est dit algorithmiquement énumérable (en abrégé énumérable) si on peut construire un mot H dans $\{0,|\}$ tel que si B est la formule $\exists \gamma (\langle H \rangle (\gamma) \doteq \beta)$ (γ et β sont des variables distinctes, du même genre que le paramètre de A), alors A est égal à B.

On peut énoncer :

Proposition 1. Un ensemble A de mots dans $\mathfrak{A}$ est résoluble si et seulement si, α étant le paramètre de A on a :

$$\forall \alpha \; (A \vee \neg A).$$

Preuve. Si A est résoluble, on peut donc construire un mot H dans $\{0,|\}$ tel que : $\forall \alpha \; ! \; \langle H \rangle (\alpha)$

et tel que $(\langle H \rangle (\beta) \doteq \Lambda)$ définit un ensemble égal à A.

On voit donc qu'un ensemble récursif est défini par une formule normale. Donc, d'après l'étude faite au point 4 il nous faut trouver un algorithme de code K tel que la formule $F_K^{\sigma} \lfloor C \rfloor$ soit vraie où C désigne la formule :

$$\exists \sigma \forall \alpha ((\langle I_0 \rangle (\sigma(\alpha)) \doteq \Lambda) \& (\langle W_2 \rangle (\sigma(\alpha)) \doteq \Lambda) \& ((\sigma(\alpha) \doteq 0|) \supset A) \& ((\sigma(\alpha) \doteq 0||) \supset \neg A)).$$

Prenons pour K le code de l'algorithme $\mathbb{F}_0 \langle H \rangle$ où $\mathbb{F}$ est l'algorithme de schème

$$\begin{cases} \xi \longrightarrow \cdot\, 0|| & (\xi \in A) \\ \longrightarrow \cdot\, 0| \end{cases}$$

On vérifie aisément qu'il répond au problème posé (car on a pour tout P dans A $(\langle W_2\rangle(\mathbb{F}(P)) \underset{\circ}{=} \Lambda)$ et $\mathbb{F}(P) \underset{\cdot}{=} 0|$ si et seulement si P est vide).

Supposons maintenant que l'on ait

$$\forall\alpha(A \vee \neg A).$$

Si A est une formule normale, on sait qu'on a (d'après ce qui a été fait au point 4) :

$$\exists\sigma\forall\alpha((\langle I_o\rangle(\sigma(\alpha)) \underset{\cdot}{=} \Lambda) \& (\langle W_2\rangle(\sigma(\alpha)) \underset{\circ}{=} \Lambda) \& ((\sigma(\alpha) \underset{\cdot}{=} 0|) \supset A) \& ((\sigma(\alpha) \underset{\circ}{=} 0||) \supset \neg A)).$$

On a donc un algorithme, qu'on peut écrire sous la forme $\langle H_o\rangle$ (plus exactement on peut construire un tel algorithme) vérifiant les propriétés indiquées plus haut.

Remarquons que d'après 3.(1) et l'axiome 3.6 on a

$$\forall\alpha(\langle I_o\rangle(\langle H_o\rangle(\alpha)) \underset{\circ}{=} \Lambda).$$

D'après le théorème de composition des algorithmes normaux on peut donc écrire :

$$\forall\alpha\ !\langle H_o\rangle(\alpha).$$

La même remarque nous permet d'écrire :

(*) $\forall\alpha\ (\langle W_2\rangle(\langle H_o\rangle(\alpha)) \underset{\circ}{=} \Lambda)$

(**) $\forall\alpha\ ((\langle H_o\rangle(\alpha) \underset{\cdot}{=} 0|) \supset A)$

(***) $\forall\alpha\ ((\langle H_o\rangle(\alpha) \underset{\circ}{=} 0||) \supset \neg A)$.

Considérons l'algorithme II de schème :

$$\begin{cases} \delta\ 0| \longrightarrow 0|\varepsilon \\ 0|\varepsilon\,\xi \longrightarrow \circ\,\xi & (\xi \in A) \\ 0|\varepsilon \longrightarrow \cdot \\ \delta \longrightarrow \circ \\ \longrightarrow \delta \end{cases}$$

et soit H le code de l'algorithme $\Pi_o\langle H_o\rangle$ (plus exactement le code de la traduction de cet algorithme dans $\mathfrak{A}^{ac}$). Alors on a encore

$$\forall\alpha\ !\langle H\rangle(\alpha)$$

car Π est applicable à tous les mots dans $\mathfrak{A}$. De plus (**) implique que $B \subseteq A$ où B désigne l'ensemble défini par :

$$(\langle H\rangle(\beta) =_\circ \Lambda).$$

Mais inversement on a :

$$\forall\alpha\ \ (A \supset (\langle H_o\rangle(\alpha) =_\circ 0||)$$

car : $\langle W_2\rangle(\langle H_o\rangle(\alpha)) =_\circ \Lambda$ et A et $\langle H_o\rangle(\alpha) =_\circ 0||$ sont incompatibles (voir [5] où on établit $((\neg A \,\&\, (A \vee B)) \supset B)$ en n'utilisant que les axiomes donnés ici).

Considérons maintenant le cas où A n'est pas une formule normale (mais A peut être équivalente à une formule normale). A est équivalente à priori à une formule de la forme $\exists\beta B$ où B est une formule normale (ici à deux paramètres α et β).

La reformulation constructive de

$$\forall\alpha\ (\exists\beta B \vee \neg\exists\beta B)$$

conduit à la formule :

$$(\square)\qquad \exists\varphi_1\varphi_2\,\forall\alpha(\langle I_o\rangle(\varphi_1(\alpha)) =_\circ \Lambda) \,\&\, (\langle I_o\rangle(\varphi_2(\alpha)) =_\circ \Lambda) \,\&\, (\langle W_2\rangle(\varphi_1(\alpha)) =_\circ \Lambda)$$

$$\&\,((\varphi_1(\alpha) =_\circ 0|) \supset (!\langle\varphi_2(\alpha)\rangle(\Lambda) \,\&\, F^{\beta}_{\langle\varphi_2(\alpha)\rangle(\Lambda)}\llcorner B\lrcorner))$$

$$\&\,((\varphi_1(\alpha) =_\circ 0||) \supset \forall\beta\neg B)$$

voir [3] (la reformulation de $(\exists\beta B \vee \neg\exists\beta B)$ passe par $(\exists\beta B \vee \forall\beta\neg B)$ dont la

reformulation donne (voir principe 4.(i))

(a) $\exists\delta((\langle W_2\rangle(\delta) \doteq \Lambda) \& ((\delta \doteq 0|) \supset \exists\beta B) \& ((\delta \doteq 0||) \supset \forall\beta\neg B))$

δ étant une variable d'objet pour $\{0,|\}$ n'entrant pas dans $(\exists\beta B \vee \forall\beta\neg B)$.

Or $((\delta \doteq 0|) \supset \exists\beta B)$ se reformule en $\exists\sigma((\delta = 0|) \supset (! \sigma(\Lambda) \& F^{\beta}_{\sigma(\Lambda)} \lfloor B \rfloor)$

où σ est une variable de genre λ (variables fonctorielles sur A). Et (a) devient

(b) $\exists\delta\sigma((\langle W_2\rangle(\delta) \doteq \Lambda) \& ((\delta \doteq 0|) \supset (! \sigma(\Lambda) \& F^{\beta}_{\sigma(\Lambda)} \lfloor B \rfloor)) \& ((\delta \doteq 0||) \supset \forall\beta\neg B))$.

On aboutit à la formule donnée en appliquant alors le principe 4.(iii), φ_1 et φ_2 sont évidemment des algorithmes sur A, transformant tout mot dans A en un mot dans $\{0,1\}$.

Soit H_1, H_2 mots dans $\{0,|\}$ tel que $\langle H_1\rangle$ et $\langle H_2\rangle$ vérifient la formule (□) quand ils remplacent φ_1 et φ_2 (de tels algorithmes peuvent être construits par hypothèse).

L'algorithme $\langle H\rangle = I_o \langle H_1\rangle$ répond à notre question : il est clair que $\forall\alpha\ !\langle H\rangle(\alpha)$ (on a $\forall\alpha\ (\langle I_o\rangle(\langle H_1\rangle(\alpha)) \doteq \Lambda)$).

Si P est un mot dans A tel que $\langle H\rangle(P) \doteq \Lambda$, alors $(\langle H_1\rangle(P) \doteq 0|)$ et donc $\langle H_2(P)\rangle$ est applicable à Λ et donne un mot tel que $F^{\alpha,\beta}_{P,\langle\langle H_2\rangle(P)\rangle(\Lambda)} \lfloor B \rfloor$ soit vraie.

Donc on a $F^{\alpha}_{P} \lfloor A \rfloor$ est vraie. Donc A contient l'ensemble $(\langle H\rangle(\alpha) \doteq \Lambda)$. Réciproquement, si $P \in A$ comme alors on a $\exists\beta B$ c.à d. qu'on peut construire un mot Q tel que $F^{\alpha,\beta}_{P,Q} \lfloor B \rfloor$ soit vraie, $P \in A$ est incompatible avec $(\langle H_1\rangle(\alpha) \doteq 0|)$. Donc, comme précédemment on a que P appartient à l'ensemble défini par $(\langle H\rangle(\alpha) \doteq \Lambda)$.

Ceci constitue donc une justification du raisonnement informel suivant :

"Supposons $\forall\alpha\ (A \vee \neg A)$. En vertu du principe 4.(i) on peut construire un algorithme $\mathbb{A}$ applicable à tous les mots dans $\mathfrak{A}$, les transformant en $0|$ ou en $0||$ et tel que l'on ait pour tout mot P dans :

$$((\mathbb{A}(P) \underset{\circ}{=} 0|) \supset A)$$
$$((\mathbb{A}(P) \underset{\circ}{=} 0||) \supset \neg A).$$

Si $\mathbb{H}$ est l'algorithme de schème indiqué plus haut, l'algorithme $\mathbb{H}_o/\mathbb{A}$ de code H vérifie :

$$\forall\alpha\ !\langle H\rangle(\alpha)$$
$$\forall\alpha((\langle H\rangle(\alpha) \underset{\circ}{=} \Lambda) \equiv F^{\gamma}_{\alpha}\lfloor A\rfloor)$$

où γ désigne le paramètre de A".

Nous aurons recours à ce type de raisonnement par la suite.

On établit aussi :

Proposition 2. Tout ensemble résoluble de mots dans $\mathfrak{A}$ est énumérable.

Preuve. Soit A un tel ensemble. On peut construire un mot H dans $\{0,|\}$ tel que :

$$\forall\alpha\ !\langle H\rangle(\alpha)$$
$$\forall\alpha((\langle H\rangle(\alpha) \underset{\circ}{=} \Lambda) \equiv A).$$

Soit $/A^{o}$ l'algorithme de schème :

$$\left\{\begin{array}{l}\xi \longrightarrow \xi\ (\xi \in B)\\ \underset{\circ}{\longrightarrow}\end{array}\right.$$

où B est l'alphabet de $\langle H\rangle$, et désignons par $/A$ l'algorithme $/A^{o}_{o}\langle H\rangle$.

Désignons par $/A^{*}$ l'algorithme dont le schème d'obtient à partir de celui de $/A^{\circ}$ en remplaçant les points par la lettre $\square$ qu'on suppose ne pas appartenir à l'alphabet de $/A$), en remplaçant dans les formules simples la lettre $\longrightarrow$ par le mot $\underset{\circ}{\longrightarrow}$ et en mettant comme premières formules du schème de $/A^{*}$, et dans cet ordre les formules

$\xi\square \longrightarrow \square\xi$ (ξ lettre de l'alphabet de $\mathcal{A}$) et $\square \longrightarrow \cdot\square$. Soit $\mathbb{C}$ l'algorithme donnant la $n^{\text{ième}}$ itérée de A^* (voir § 1. 5.6 théorème 1). Soit $\mathbb{D}$ l'algorithme de schème :

$$\begin{cases} \square\xi \longrightarrow \square & (\xi \in \text{alphabet de } \mathcal{A}) \\ \square \longrightarrow \circ & \\ \longrightarrow \cdot| & \end{cases}$$

On considère alors $\mathbb{B}(P\square n) \simeq \mathbb{D}(\mathbb{C}(\Lambda * P))$ où P est un mot dans $\mathcal{A}$ et n un entier.

L'algorithme $\mathbb{B}$ donne le résultat de $n^{\text{ième}}$ pas du processus de l'application de $\mathcal{A}$ au mot P. On vérifie aisément que $\mathbb{B}$ possède les propriétés :

(i) pour tout mot P dans $\mathcal{A}$ et pour tout entier n, $\mathbb{B}(P\square n)$ a un sens.

(ii) pour tout entier n, si $\mathbb{B}(P\square n) = \Lambda$, alors pour tout entier m, $m \geqslant n$ on a aussi $\mathbb{B}(P\square m) \underset{\circ}{=} \Lambda$.

(iii) $\mathcal{A}$ est applicable au mot P si et seulement si il existe un entier n tel que $\mathbb{B}(P\square n) \underset{\cdot}{=} \Lambda$.

On appellera B le développement canonique de $\mathcal{A}$.

Soit $\langle I_{\underline{nat}} \rangle$ un algorithme de reconnaissance des entiers parmi les mots de $\{0,|\}$.

Soit $\mathfrak{B}$ algorithme tel que pour tout mot P dans $\mathcal{A}$ et tout mot N dans $\{0,|\}$ on ait $\mathfrak{B}(P\square N) \simeq \langle I_{\underline{nat}} \rangle (N)\mathbb{B}(P\square N)$. Alors, par construction de $\mathcal{A}$ on a :

$\forall\alpha(!\mathcal{A}(\alpha) \equiv (\langle H \rangle(\alpha) \underset{\circ}{=} \Lambda))$ et donc : $\forall\alpha((\langle H \rangle(\alpha) = \Lambda) \equiv \exists\delta(\mathfrak{B}(\alpha\square\delta) = \Lambda))$ où δ est une variable d'objet pour les mots dans $\{0,|\}$.

On veut mettre la condition $\exists\delta(\mathfrak{B}(\alpha\square\delta) \underset{\circ}{=} \Lambda)$ sous la forme $\exists\delta(\mathbb{E}(\delta) = \alpha)$ où $\mathbb{E}$ est un algorithme dont nous allons indiquer la construction.

On peut construire un algorithme $\mathbb{N}$ sur $\{0,|\}$ applicable à tous les entiers et énumérant sans répétition (c. à d. $n \neq m$ entraine $\mathbb{N}(n) \neq \mathbb{N}(m)$ tous les mots dans

A, ainsi que son inverse $\mathbb{N}^{-1}$. A étant fixé et se fondant sur l'ordre lexicographique on peut donner $\mathbb{N}^{-1}$ par la formule :

$$\mathbb{N}^{-1}(P) \underset{\cdot}{=} \left(\sum_{i=1}^{\ell(P)} \mathbb{N}^{-1} p_i\right) n^{i-1}) \quad \text{et} \quad \mathbb{N}^{-1}(\Lambda) \underset{\cdot}{=} 0$$

où n est le nombre de lettres de A, p_i la ième lettre du mot P. On fixe $\xi_1, \ldots, \xi_n$ les lettres de A en posant $\mathbb{N}^{-1}(\xi_i) = i$ $(i = 1, \ldots, n)$. Alors l'inverse $\mathbb{N}$ peut être donné par le schème ($\mathbb{N}$ est applicable à tous les mots) :

$$\begin{cases} a| \longrightarrow b| \\ a0| \longrightarrow b \\ a0 \longrightarrow \cdot \\ a \longrightarrow \cdot \\ b|^{n} \longrightarrow cb \\ b|^{i-1} \longrightarrow \xi_i \quad (i = 2, \ldots, n) \\ b \longrightarrow \xi_1 \\ c \longrightarrow | \\ \longrightarrow a \end{cases}$$

c, b, a sont supposés ne pas appartenir à $A \cup \{0,|\}$ et on suppose ici $0,| \notin A$ pour la commodité de la construction de $\mathbb{N}$.

On construit en même temps une bijection constructive entre les entiers et les couples d'entiers. On pose

$$\mathbb{K}(n \square m) \underset{\cdot}{=} 2^m(2n + 1) - 1 \quad \text{où } n \text{ et } m \text{ sont des entiers.}$$

(dans [15] . S. Tseïtine donne un schème possible pour $\mathbb{K}$).

On sait construire des algorithmes $\mathbb{I}_1$ et $\mathbb{I}_2$ applicables à tous mots dans $\{0,|\}$ (cf. [15]), tels que pour tout entiers n et m on ait :

$$(\mathbb{K}(\mathbb{I}_1(n) \square \mathbb{I}_2(n)) \underset{\cdot}{=} n)$$

$$(\mathbb{I}_1(\mathbb{K}(n \square m)) \underset{\cdot}{=} n) \;\&\; (\mathbb{I}_2(\mathbb{K}(n \square m)) \underset{\cdot}{=} m)).$$

A partir du développement canonique de $\mathfrak{A}$, c. à d. $\mathfrak{B}$ et de l'algorithme $\mathfrak{N}$ et des théorèmes du § 1, on peut construire un algorithme $\mathfrak{B}_1$ applicable à tous les mots $n \square m$ où n et m sont des entiers et tels que l'on ait pour tous les entiers n et m

$$((\mathfrak{B}(\mathfrak{N}(n) \square m) \doteq \Lambda) \equiv (\mathfrak{B}_1(n \square m) \doteq \Lambda))$$

avec $$((\mathfrak{B}_1(n \square m) \neq \Lambda) \supset (\mathfrak{B}_1(n \square m) \doteq \square)).$$

En utilisant le théorème de réunion des algorithmes, on peut construire un algorithme $\mathfrak{E}_o$ tel que pour tout entier n on ait :

$$\mathfrak{E}_o(n) \doteq \mathfrak{N}(\Pi_1(n))\mathfrak{B}_1(\Pi_1(n) \square \Pi_2(n)).$$

Comme $\square \notin \mathfrak{A}$, on constate aisément que

$$\forall \alpha (\exists \delta (\mathfrak{F}(\delta)\mathfrak{E}_o(\delta) \doteq \alpha) \equiv \exists \gamma (\mathfrak{B}(\alpha \square \gamma) \doteq \Lambda))$$

où $\mathfrak{F}$, construit à partir de $\langle I_{\underline{nat}} \rangle$ transforme en le mot vide les mots dans $\{0,|\}$ qui sont des entiers et eux seuls, et transforme en $\square$ les autres mots. Si on pose $\mathfrak{E}(N) \simeq \mathfrak{F}(N)\mathfrak{E}_o(N)$ pour tout mot dans $\{0,|\}$ (théorème de réunion des algorithmes) on a le résultat annoncé. $\square$

Simultanément, on a la

Proposition 3. On peut construire un ensemble énumérable de mots dans $\{0,|\}$ qui ne soit pas résoluble.

Nous ne démontrons pas ce résultat ici. Nous l'avons en fait établi en construisant un algorithme dont on ne peut déterminer algorithmiquement l'applicabilité, car on peut établir (cf. [15]) que le domaine de définition d'un algorithme est énumérable : il

suffit d'utiliser l'algorithme de développement canonique construit précédemment.

Enfin, concernant les ensembles énumérables on peut énoncer :

Proposition 4. Soit A un ensemble énumérable de mots dans $\mathcal{A}$. Alors :

$$\forall\alpha(\neg\neg A \supset A)$$

où α désigne le paramètre de A.

La démonstration repose sur le

Lemme : Soit A un ensemble énumérable de mots dans $\mathcal{A}$. On peut construire un mot H dans $\{0,|\}$ tel que :

(i) $\forall\alpha\ !\langle H\rangle(\alpha)$

(ii) $\forall\alpha\ (A \equiv \exists\gamma(\langle H\rangle(\gamma) \doteq \alpha))$ (γ et α du même genre).

Preuve du lemme.

Désignons par $\mathcal{A}$ l'algorithme $\langle H\rangle$ et par $\mathcal{B}$ le développement canonique de $\mathcal{A}$. Soit $\mathcal{B}_1$ tel que pour tout couple d'entiers n et m

$$\mathcal{B}_1(n \square m) \simeq \mathcal{B}(\mathcal{N}(m) \square n)$$ où $\mathcal{N}$ énumère les mots dans $\mathcal{A}$ à l'aide des entiers.

Soit $\mathcal{C}(n) \simeq \mathcal{B}_1(\mathcal{I}_1(n) \square \mathcal{I}_2(n))$ (cf. démonstration de la proposition 2). Alors soit

$$\mathcal{D}(N) \simeq \mathcal{F}(N)\mathcal{N}(\mathcal{I}_1(N))\mathcal{B}_1(\mathcal{I}_1(N) \square \mathcal{I}_2(N)).$$

On peut construire un algorithme $\mathcal{E}_o$ sur $\mathcal{A}$ (contenant $\{0,|\}$) tel que pour tout mot P dans $\mathcal{A}$, $!\mathcal{E}_o(P)$ et $\mathcal{E}_o(P)$ est un mot dans $\{0,|\}$ (on prend par exemple $\mathcal{E}_o$ de schème :

$$\{\xi \longrightarrow (\xi \in \mathcal{A} \setminus \{0,|\})).$$

Soit alors, pour tout mot P dans $\mathcal{A}$, $\mathcal{E}(P) \simeq \mathcal{D}(\mathcal{E}_o(P))$.

Il est clair que $\forall\alpha\ !\mathcal{E}(\alpha)$

et : $$(\exists \gamma(\langle H\rangle(\gamma) \doteq \alpha) \equiv \exists \beta(\mathbb{E}(\beta) \doteq \alpha))$$

où β est du même genre que γ et α. □

Preuve de la proposition.

On considère H mot dans $\{0,|\}$ tel que : $\forall \alpha\ !\langle H\rangle(\alpha)$ et $\forall \alpha(A \equiv \exists \gamma(\langle H\rangle(\gamma) \doteq \alpha))$. Soit P un mot dans A. Comme $\forall \gamma\ !\langle H\rangle(\gamma)$ on peut construire un algorithme testant si $\langle H\rangle(\gamma) \doteq P$ ou non (et cet algorithme peut être construit indépendamment de P). D'après le principe 4.(i) on a donc :

$$\forall \gamma((\langle H\rangle(\gamma) \doteq P) \vee \neg(\langle H\rangle(\gamma) \doteq P)).$$

Par le principe de Markov, on a donc :

$$(\neg\neg \exists \gamma\ (\langle H\rangle(\gamma) \doteq P) \supset \exists \gamma\ (\langle H\rangle(\gamma) \doteq P)).$$

On a ceci quel que soit P.

Donc la règle

$$\frac{A \supset \mathfrak{F}(a)}{A \supset \forall \alpha\, \mathfrak{F}(x)}$$

permet d'écrire : $$\forall \alpha(\neg\neg \exists \gamma(\langle H\rangle(\gamma) \doteq \alpha) \supset \exists \gamma(\langle H\rangle(\gamma) \doteq \alpha))$$

c'est-à-dire $$\forall \alpha(\neg\neg A \supset A).$$ □

5.2. Il faut mentionner qu'il existe une autre notion constructive d'ensemble : la notion d'ensemble génératif.

Un exemple est donné par la définition des entiers donnée au début du point 4 du premier paragraphe.

Cette notion est basée sur celle de calcul. Un calcul est constitué par la donnée d'un alphabet, l'alphabet du calcul, et une liste (finie) de règles de déduction indiquant comment on obtient les mots dans l'alphabet du calcul, déductibles dans ce

calcul (on peut donner ou ne pas donner des mots initiaux du calcul (jouant le rôle d'axiomes dans le calcul logique évoqué plus haut) qu'on suppose déjà déduits. Un couple d'objets constitués d'un alphabet et d'un calcul sur cet alphabet (cf. définition des algorithmes normaux) est appelé ensemble génératif. Un mot P dans $\mathfrak{A}$ est dit appartenir à cet ensemble, si le mot P est déductible dans le calcul donné.

On définit l'égalité entre un ensemble génératif de mots dans $\mathfrak{A}$ et un ensemble de mots dans $\mathfrak{A}$ par l'équivalence de l'appartenance à chacun de ces ensembles.

On démontre qu'il y a équivalence entre la notion d'ensemble génératif et celle d'ensemble énumérable.

6. Dans ce qui a été fait jusqu'ici, les variables représentaient toujours des mots quelconques dans des alphabets donnés. Dans ce qui précède on a déjà formulé des énoncés relatifs seulement aux entiers naturels. Par la suite on aura souvent à raisonner sur une catégorie délimitée d'objets, par exemple, les rationnels ou les réels, etc...

On est alors conduit à introduire de nouvelles variables appelées variables restreintes.

Une variable restreinte (d'objet ou fonctorielle) est définie comme au point 2 par une lettre générique $\mathfrak{v}$. Les valeurs admissibles seront de nouveaux des mots du genre $\mathfrak{v}$. Mais ces mots ne sont plus les mêmes que précédemment. Soit $\mathfrak{A}$ l'alphabet correspondant à la variable restreinte envisagée. Notons $\mathfrak{d}$ la lettre générique des variables restreintes d'objet, $\mathfrak{p}$ celle des variables restreintes fonctorielles. La restriction apportée est qu'un mot du genre $\mathfrak{d}$ est un élément d'un ensemble de mot dans $\mathfrak{A}$ fixé

à l'avance par une formule A ; un mot du genre ρ est un mot appartenant à un ensemble B du genre λ (c. à d. le paramètre de B est une variable fonctorielle sur $\mathfrak{A}$).

Les formules A et B sont dites formules caractéristiques des lettres d et ρ respectivement.

Pour l'interprétation des formules contenant des variables restreintes, on passe par l'intermédiaire des formules dites de base (c'est-à-dire ne contenant pas de variables restreintes). Dans la monographie [3] déjà citée N. A. Chanine décrit un algorithme normalisable transformant toute formule $\mathfrak{F}$ en une formule de base qui lui est équivalente qu'on note $c\lfloor\mathfrak{F}\rfloor$. L'interprétation de $\mathfrak{F}$ est alors l'interprétation de la formule $c\lfloor\mathfrak{F}\rfloor$.

Dans [1] N. A. Chanine énonce des théorèmes sur l'interprétation constructives des formules dans le cas où les formules caractéristiques des variables restreintes sont normales, et qui permettent d'éviter le passage par la formule de base.

Le principe du passage à la formule de base associée à une formule est d'expliciter les conditions liées à toutes les variables restreintes entrant dans la formule. Les théorèmes indiquent alors qu'on peut appliquer le principe (iii) aux formules contenant des variables restreintes.

On peut également démontrer de telles formules en utilisant les axiomes 3.1 à 3.15 et les règles de déduction. L'axiome 3.16 est valable pour les variables de base ou les variables restreintes dont l'ensemble caractéristique est énumérable (et non vide).

Reprénons le cas de la formule

(1) $\forall\alpha(A \vee B)$

où α est une variable restreinte, A et B des formules normales à un paramètre du même genre que celui de α. Soit C la formule caractéristique de α. La formule de base équivalente à (1) est :

(1') $$\forall_\beta \ (F^\alpha_\beta \ulcorner C \urcorner \supset F^\alpha_\beta \ulcorner (A \vee B) \urcorner)$$

β variable de base correspondant à l'alphabet de définition de C.

Désignons par C_1 la formule $F^\alpha_\beta \ulcorner C \urcorner$

A_1 la formule $F^\alpha_\beta \ulcorner A \urcorner$

B_1 la formule $F^\alpha_\beta \ulcorner B \urcorner$.

Il est clair que $F^\alpha_\beta \ulcorner (A \vee B) \urcorner = (F^\alpha_\beta \ulcorner A \urcorner \vee F^\alpha_\beta \ulcorner B \urcorner)$.

Supposons C normale.

Alors on a à reformuler :

(1") $$\forall_\beta \ (C_1 \supset (A_1 \vee B_1))$$

ou

(2) $$\forall_\beta \ (C_1 \supset \exists \delta \mathfrak{F})$$

où $\mathfrak{F}$ est la formule :

$$((<W_2>(\delta) \underset{\circ}{=} \Lambda) \ \& \ ((\delta \underset{\circ}{=} 0|) \supset A_1) \ \& \ ((\delta \underset{\circ}{=} 0||) \supset B_1)).$$

On reformule $(C_1 \supset \exists \delta \mathfrak{F})$ qui, du fait que δ ne figure pas dans C_1 donne :

$$\exists (C_1 \supset ! \sigma(\Lambda) \ \& \ F^\delta_{\sigma(\Lambda)} \ulcorner \mathfrak{F} \urcorner)$$

σ étant une variable fonctorielle correspondant à l'alphabet de δ (c'est-à-dire $\{0,|\}$)

Si on note $\mathfrak{F}_1$ la formule $(C_1 \supset !\sigma(\Lambda) \& F^\delta_{\sigma(\Lambda)} \ulcorner \mathfrak{F} \urcorner)$ on arrive à :

(3) $$\forall \beta \exists \sigma \ \mathfrak{F}_1 .$$

On applique alors le principe 4.(iii) d'où :

(4) $\exists\tau\,\forall\beta((\langle I_{\langle\rangle\beta}\rangle(\tau(\beta)) = \Lambda)\,\&\,F^{\sigma}_{\langle\tau(\beta)\rangle\ulcorner\mathcal{C}_1\urcorner})$

τ étant une variable fonctorielle sur l'alphabet de β transformant tout β en un code d'algorithme sur $\{0,1\}$ et $\langle I_{\langle\rangle\beta}\rangle$ est un algorithme reconnaissant les mots de $\{0,1\}$ qui sont codes d'un algorithme sur l'alphabet de β.

Donc la reformulation constructive de (1) est :

(5) $\exists\tau\,\forall\beta((\langle I_{\langle\rangle\beta}\rangle(\tau(\beta)) = \Lambda)\,\&\,(F^{\alpha}_{\beta\ulcorner C\urcorner} \supset (!\langle\tau(\beta)\rangle(\Lambda)\,\&\,(\langle W_{\tau}\rangle\langle\tau(\beta)\rangle(\Lambda) = \Lambda)$

$\&\,((\langle\tau(\beta)\rangle(\Lambda) = 0|) \supset F^{\alpha}_{\beta\ulcorner A\urcorner})\,\&\,((\langle\tau(\beta)\rangle(\Lambda) = 0||) \supset F^{\alpha}_{\beta\ulcorner B\urcorner}))))$.

Or celle de (5) est une interprétation constructive de :

(6) $\exists\tau\,\forall\alpha((\langle I_0\rangle(\tau(\alpha)) = \Lambda)\,\&\,(\langle W_{\tau}\rangle(\tau(\alpha)) = \Lambda)\,\&\,((\tau(\alpha) = 0|) \supset A)\,\&\,((\tau(\alpha) = 0||) \supset B))$.

Il reste à préciser quelques notations :

Si $\mathbb{A}$ est un algorithme applicable à tout mot du genre $\mu_1\ldots\mu_k$ (où $\mu_1,\ldots,\mu_k$ sont des lettres génériques de variables (restreintes ou non) et un mot de genre $\mu_1\ldots\mu_k$ est de la forme $P_1\square\ldots\square P_k$ où P_i est de genre μ_i) et transforme tout mot de ce genre en un mot de genre μ_{k+1} on écrira :

$$\mathbb{A} \in (\mu_1\ldots\mu_k \to \mu_{k+1}).$$

Si $\mathbb{A}$ transforme tout mot de genre $\mu_1\ldots\mu_k$ auquel il est applicable en un mot de genre μ_{k+1}, on écrira :

$$\mathbb{A} \in (\mu_1\ldots\mu_k \overset{*}{\to} \mu_{k+1}).$$

Enfin, en établissant une convention sur les symboles $\&$, $\vee$, $\supset$ et les symboles opératoires qu'on introduira par la suite, on peut réduire le nombre de parenthèses dans une formule. On donne un rang à chaque symbole (c-à-d on lui

associe un entier). Un tel symbole est dit marqué. Le rang indique alors le degré de liaison du symbole en fonction duquel on peut ôter des parenthèses. Dans [1] N.A. Chamine donne un algorithme partant d'expressions avec ou sans parenthèses permettant de reconnaître si ces expressions sont ou non des abréviations de formules.

§3. Les nombres réels constructifs.

1. Définition des nombres réels constructifs.

La notion de nombre réel la plus importante en analyse constructive est celle qui traduit l'idée d'approximation à un degré de précision connu et fixé à l'avance.

Dans ce qui suit nous ne parlerons pas d'autres notions de nombres réels. Nous renvoyons cette question en appendice.

La définition des nombres réels s'appuie sur celle des nombres rationnels, s'appuyant elle même sur celle des entiers relatifs.

Rappelons que les entiers naturels sont les mots suivants dans l'alphabet $\{0,1\}$ (qu'on désignera par N_o)

(i) 0 est un entier naturel

(ii) si N est un entier naturel, le mot $N|$ est un entier naturel.

On vérifie aisément que les entiers naturels sont les mots dans N_o qui vérifient la formule $(\langle H\rangle(\alpha) = \Lambda)$ où $\langle H\rangle$ est l'algorithme de schème :

$$\begin{cases} 0| \to 0 \\ 0 \to \circ \\ \to \cdot| \end{cases}$$

Les entiers relatifs (en abrégé entiers) sont les mots dans l'alphabet $N_1 \leftrightarrows \{0,|,-\}$ (le symbole $\leftrightarrows$ signifie : "est une notation de") définis de façon générative comme suit :

(i) les entiers naturels sont des entiers

(ii) si N et M sont des entiers naturels, le mot N-M est un entier.

On peut aisément trouver un algorithme permettant de reconnaître parmi les mots dans N_1 ceux qui sont des entiers.

Les nombres rationnels (on dira "rationnel" en abrégé pour "nombre rationnel") sont les mots dans l'alphabet $N_2 \leftrightarrows \{0,|,-,/\}$ de la forme suivante :

(i) si P est un entier, P est un rationnel

(ii) si P est un entier et Q un entier naturel différent de 0 alors le mot P/Q est un rationnel.

A l'aide d'un algorithme calculant le p.g.c.d. de deux entiers on peut définir aisément la notion de nombre rationnel irréductible. Les lettres génériques des variables restreintes d'entiers naturels, d'entiers et de rationnels sont respectivement : H , U , P .

Considérons maintenant l'alphabet $N_3 \leftrightarrows \{0,|,-,/,\Diamond\}$. On appelle nombres réels (en abrégé "réels") les rationnels et les mots dans N_3 de la forme $P \Diamond Q$ où :

FACULTE DES SCIENCES ORSAY BIBLIOTHEQUE MATHEMATIQUE

(i) P est le code dans $\{0,|\}$ d'un algorithme du type $(H \to P)$

(ii) Q est le code dans $\{0,|\}$ d'un algorithme du type $(H \to H)$

tel que :

$$\forall k\ell m \quad (\ell,m \geqslant Q(k) \supset |P(m)-P(\ell)| < 2^{-k})$$

où k,ℓ,m désignent les variables $(H),(HH)$ et (HHH) et où on écrit en abrégé $P(H)$ au lieu de $\langle P\rangle(H)$ et $Q(H)$ au lieu de $\langle Q\rangle(H)$.

De plus $|P(m)-P(\ell)|$ désigne la valeur absolue du rationnel $P(m)-P(Q)$ (qu'on définit sans difficulté).

Remarque 1 : On peut définir une représentation canonique des rationnels de la forme $P \diamond Q$.

On peut construire un algorithme τ tel que pour tout rationnel a, $\tau(a)$ est le code dans $\{0,|\}$ d'un algorithme du type $(H \to p)$ vérifiant : $\forall H(\langle\tau(a)\rangle(H) = a)$ (voir [1]).

Si $\mathfrak{b}$ est le code de l'algorithme de schème :

$$\begin{cases} \xi \to & (\xi \in N_0) \\ \to .0 \end{cases}$$

il est évident que $\tau(a) \diamond \mathfrak{b}$ est un réel, dont on verra plus loin qu'il est égal à a.

Remarque 2 : La propriété "P est un mot du type H (resp. μ, p)" est algorithmiquement vérifiable. Dans N_3 on peut vérifier par un procédé algorithmique la propriété pour un mot d'être de la forme $P \diamond Q$ où P et Q sont des mots dans N_0. Mais, comme nous le verrons plus loin, on ne peut pas vérifier algorithmiquement la propriété pour un mot dans N_3 d'être un réel.

Par la suite on notera $\mathfrak{A}$ le genre de la variable restreinte correspondant aux nombres réels.

On réservera les premières lettres de l'alphabet des lettres latines miniscules pour désigner les rationnels, et les dernières pour représenter les réels.

Si x désigne donc une variable de nombre réel on introduit les notations $\underline{x}$ pour désigner l'algorithme de type $(H \rightarrowtail p)$ dont le code figure dans x et $\overline{x}$ pour désigner l'algorithme de type $(H \rightarrow H)$. On appellera $\underline{x}$ base du réel x et $\overline{x}$ régulateur de convergence de x.

2. Egalité et inégalité entre les réels.

2.1. On définit l'égalité sur les entiers naturels par l'identité graphique. On définit les inégalités $\leqslant$ et $\geqslant$ par :

$$(n \leqslant m) \equiv (\mathbb{A}(n*m) = \Lambda)$$

où $\mathbb{A}$ est l'algorithme de schème :

$$\begin{cases} |*0| \rightarrow *0 \\ |*0 \rightarrow .| \\ 0*0| \rightarrow 0*0 \\ 0*0 \rightarrow . \end{cases}$$

On étend aisément ces définitions aux entiers puis aux rationnels.

De même on définit les relations $<$ et $>$.

On vérifie aisément que :

$$\forall ab \quad (a \leqslant b \equiv a < b \vee a = b)$$

$$\forall a \qquad (a > 0 \vee a \leqslant 0)$$

On a d'ailleurs :

Lemme technique 1. Pour tout rationnel a on a :

$$\forall \ell \quad (a < 2^{-\ell}) \supset (a \leqslant 0)$$

Preuve : Si a est un rationnel positif (c.à.d. $a > 0$) on peut trouver (au moyen d'un algorithme ne dépendant pas de a) un entier naturel ℓ tel que $a \geqslant 2^{-\ell}$ (soit a un rationnel donné avec $a > 0$, alors si P/Q est la forme irréductible de a, on a $a \geqslant 0|/Q$ et de là on tire aisément ℓ).

Donc : $\qquad a > 0 \supset \exists \ell \quad (a \geqslant 2^{-\ell})$

Donc : $\qquad \neg \exists \ell \ (a \geqslant 2^{-\ell}) \supset \neg(a > 0)$ $\qquad$ (on applique 3.(4))

Ou : $\qquad \forall \ell \quad (a \geqslant 2\ell) \supset (a \leqslant 0)$

C.à.d. $\qquad \forall \ell \ (a < 2^{-\ell}) \supset (a \leqslant 0)$

car on vérifie aisément que l'on a $\forall ab(\neg(a > b) \equiv (a \leqslant b))$.

Enfin, il est clair que la relation d'égalité est une relation d'équivalence sur les rationnels et que les relations $\leqslant$ et $\geqslant$ sont des relations d'ordre.

2.2. Définition de l'égalité et de l'inégalité de deux nombres réels.

Si x et y sont des réels, on pose les relations suivantes :

(1) $x = y \leftrightarrows \forall k \exists \ell \forall m \ (m \geqslant \ell \supset |\underline{x}(m) - \underline{y}(m)| < 2^{-k})$
et on lit x égale y

(2) $x \leqslant y \leftrightarrows \forall k \exists \ell \forall m \ (m \geqslant \ell \supset \underline{x}(m) - \underline{y}(m) < 2^{-k})$
et on lit x minore y

(3) $x \geqslant y \leftrightarrows \forall k \exists \ell \forall m \ (m \geqslant \ell \supset \underline{y}(m) - \underline{x}(m) < 2^{-k})$
et on lit x majore y

(4) $x \neq y \leftrightarrows \exists k\ell\, \forall m\, (m \geqslant \ell \supset |\underline{x}(m) - \underline{y}(m)| \geqslant 2^{-k})$

et on lit x est différent de y

(5) $x < y \leftrightarrows \exists k\ell\, \forall m\, (m \geqslant \ell \supset \underline{y}(m) - \underline{x}(m) \geqslant 2^{-k})$

et on lit x est plus petit que y

(6) $x > y \leftrightarrows \exists k\ell\, \forall m\, (m \geqslant \ell \supset \underline{x}(m) - \underline{y}(m) \geqslant 2^{-k})$

et on lit x est plus grand que y .

Désignons par σ le genre des variables restreintes désignant les algorithmes du type $(H \to H)$ et par f la variable restreinte (σ). La reformulation constructive de (1), (2) et (3) exprimée à l'aide des variables restreintes donne :

(1') $x = y \equiv \exists f\, \forall km\, (m \geqslant f(k) \supset |\underline{x}(m) - \underline{y}(m)| < 2^{-k})$

(2') $x \leqslant y \equiv \exists f\, \forall km\, (m \geqslant f(k) \supset \underline{x}(m) - \underline{y}(m) < 2^{-k})$

(3') $x \geqslant y \equiv \exists f\, \forall km\, (m \geqslant f(k) \supset \underline{y}(m) - \underline{x}(m) < 2^{-k})$.

A l'aide du lemme technique 1 on voit que ces relations coïncident avec celles que l'on a défini en 2.1 pour les rationnels. On trouve également que :

$$\forall a\, (\tau(a) \diamond \mathfrak{b} = a).$$

Nous utiliserons par la suite le

Lemme technique 2. Pour tous réels x et y on a :

(1a) $x = y \equiv \forall n(|\underline{y}(\overline{y}(n+1)) - \underline{x}(\overline{x}(n+1))| \leqslant 2^{-n})$

(2a) $x \leqslant y \equiv \forall n(\underline{x}(\overline{x}(n+1)) - \underline{y}(\overline{y}(n+1)) \leqslant 2^{-n})$

(3a) $x \geqslant y \equiv \forall n(\underline{y}(\overline{y}(n+1)) - \underline{x}(\overline{x}(n+1)) \leqslant 2^{-n})$

(4a) $x \neq y \equiv \exists n(|\underline{y}(\overline{y}(n+1)) - \underline{x}(\overline{x}(n+1))| > 2^{-n})$

(5a) $x < y \equiv \exists n(\underline{y}(\overline{y}(n+1)) - \underline{x}(n+1)) > 2^{-n})$

(6a) $x > y \equiv \exists n(\underline{x}(\overline{x}(n+1)) - \underline{y}(\overline{y}(n+1)) > 2^{-n})$.

Preuve : Pour simplifier l'écriture, nous adopterons la notation suivante dans la démonstration :

$$x_n = \underline{x}(\overline{x}(n+1)) \quad \text{pour tout réel } x \text{ .}$$

Démontrons simplement (1a) et (4a) :

Pour (1a), on a par définition :

$$x = y \supset \exists f \, \forall k m \; (m \geqslant f(k) \supset |\underline{x}(m) - \underline{y}(m)| < 2^{-k}).$$

Or : $|x_n - y_n| \leqslant |y_n - \underline{y}(m)| + |\underline{y}(m) - \underline{x}(m)| + |x(m) - x_n|$; prenons $m \geqslant \max$, $\overline{y}(n+1)$, $\overline{x}(n+1))$ (on peut aisément définir algorithmiquement le maximum de deux entiers).

Il vient alors : $|x_n - y_n| < 2^{-n-1} + |x_m - y_m| + 2^{-n-1}$ (par définition du régulateur de convergence), c.à.d.

$$|x_n - y_n| < 2^{-n} + |x_m - y_m| \text{ .}$$

Donnons-nous k entier naturel. Si on prend de plus $m \geqslant f(k)$ on a donc :

$$|x_n - y_n| < 2^{-n} + 2^{-k} \text{ .}$$

Donc $$\forall k \; (|x_n - y_n| < 2^{-n} + 2^{-k})$$

et par le lemme technique 1 on a donc $|x_n - y_n| \leqslant 2^{-n}$.

Supposons maintenant que :

$$\forall n \quad (|x_n - y_n| \leqslant 2^{-n}) \text{ .}$$

Si on prend $f(k) = \max(\overline{y}(k+2) , \overline{x}(k+2))$, par le même type de majoration on trouve (1).

Pour (4a), par hypothèse :

$$\exists k \ell \, \forall m \; (m \geqslant \ell \supset |\underline{x}(m) - \underline{y}(m)| \geqslant 2^{-k})$$

Donc :

$$2^{-k} \leqslant |\underline{x}(m)-\underline{y}(m)| \leqslant |\underline{x}(m)-x_n| + |x_n-y_n| + |y_n-\underline{y}(m)| < |x_n-y_n| + 2^{-n}$$

en prenant comme précédemment $m \geqslant \max(\overline{x}(n+1),\overline{y}(n+1))$ et en supposant n fixé.

En fixant donc $n > k+1$ on trouve :

$$|x_n-y_n| > 2^{-k} - 2^{-n} \geqslant 2^{-n} .$$

D'où $(x \neq y) \supset \exists n\ (|x_n-y_n| > 2^{-n})$. La réciproque s'établit de la même façon. □ .

En utilisant de la même façon les définitions, on démontre aisément le

Lemme technique 3. Soient x et y deux réels. Si :

$$\exists m\ \forall \ell\ (\ell \geqslant m \supset y \leqslant \underline{x}(\ell))$$

[resp. $\exists m\ \forall \ell\ (\ell \geqslant m \supset \underline{x}(\ell) \leqslant y)$] alors $y \leqslant x$ [resp. $x \leqslant y$].

2.3. Propriétés des relations d'égalité et d'inégalité entre nombres réels.

En revenant aux définitions, il est clair que la relation $=$ est une relation d'équivalence. Nous allons démontrer le théorème 1 (voir N.A. Chanine [1]).

Pour tous réels x et y on a :

(1)
$$\begin{cases} x = y \equiv x \leqslant y \quad x \geqslant y \\ x \neq y \equiv x < y \quad x > y \\ x < y \equiv x \leqslant y \quad x \neq y \\ x > y \equiv x \geqslant y \quad x \neq y \end{cases}$$

(2)
$$\begin{cases} \neg(x \neq y) \equiv x = y \\ \neg(x < y) \equiv x \geqslant y \\ \neg(x > y) \equiv x \leqslant y \\ \neg(x = y) \equiv x \neq y \\ \neg(x \geqslant y) \equiv x < y \\ \neg(x \leqslant y) \equiv x > y \end{cases}$$

(3)
$$\begin{cases} \neg\neg(x \neq y) \equiv x \neq y \\ \neg\neg(x = y) \equiv x = y \\ \neg\neg(x \geqslant y) \equiv x \geqslant y \\ \neg\neg(x \leqslant y) \equiv x \leqslant y \\ \neg\neg(x > y) \equiv x > y \\ \neg\neg(x < y) \equiv x < y) \end{cases}$$

(4)
$$\begin{cases} \neg(x \neq y \,\&\, x = y) \\ \neg(x < y \,\&\, x \geqslant y) \\ \neg(x > y \,\&\, x \leqslant y) \\ \neg(x \neq y \,\&\, x \geqslant y \quad x \leqslant y) \\ \neg(x < y \,\&\, x > y) \end{cases}$$

$$(5)\begin{cases} x > y \vee x = y \supset x \geqslant y \\ x < y \vee x = y \supset x \leqslant y \\ x \geqslant y \equiv \neg\neg(x > y \vee x = y) \\ x \leqslant y \equiv \neg\neg(x < y \vee x = y) \end{cases} \qquad (6)\begin{cases} \neg\neg(x = y \vee x \neq y) \\ \neg\neg(x \geqslant y \vee x < y) \\ \neg\neg(x \leqslant y \vee x > y) \\ \neg\neg(x = y \vee x < y \vee x > y) \\ \neg\neg(x \geqslant y \vee x \leqslant y) \end{cases}$$

La démonstration repose sur le

Lemme : Pour tout algorithme A du type $(H \to p)$ on a :

$$\forall n\,(\mathbb{A}(n) \geqslant 0) \equiv \neg\neg\forall n(\mathbb{A}(n) \geqslant 0)$$

$$\exists n\,(\mathbb{A}(n) > 0) \equiv \neg\neg\exists n(\mathbb{A}(n) > 0)$$

Preuve du lemme : Comme $\mathbb{A}(n) \geqslant 0$ est vérifiable par un algorithme (applicable à tous les réels, cette formule est équivalente à :

$\forall n(\mathbb{B}(n) = \Lambda)$ où $\mathbb{B}$ est un algorithme résultant de la composition de $\mathbb{A}$ avec un algorithme reconnaissant la propriété $a \geqslant 0$ pour les rationnels. C'est une formule normale qui est donc équivalente à sa double négation.

Considérons maintenant la formule $\exists n\,(\mathbb{A}(n) > 0)$. On sait que l'on peut construire un algorithme reconnaissant parmi les rationnels ceux qui possèdent la propriété $a > 0$. On peut donc savoir pour chaque n, si l'on est dans le cas $\mathbb{A}(n) > 0$ ou dans le cas $\mathbb{A}(n) \leqslant 0$ $(\equiv \neg(\mathbb{A}(n) > 0))$.

On a donc :

$$\forall n\,(\mathbb{A}(n) > 0 \vee \neg(\mathbb{A}(n) > 0)).$$

En vertu du principe de Markov et de la règle de déduction $\dfrac{A(A \supset B)}{B}$.

On peut énoncer :

$$\neg\neg\exists n\,(\mathbb{A}(n) > 0) \supset \exists n(A(n) > 0).$$

Donc, comme on a toujours $A \supset \neg\neg A$ pour toute formule A (cf. par exemple [8]) la seconde formule du lemme est donc vraie. □.

Preuve du théorème : Le lemme précédent et le lemme technique 2 et la formule de contraposition (§2 3(5)) donnent aussitôt les formules des groupes (2) et (3) du théorème.

Les deux premières formules du groupe (1) résultent respectivement du lemme technique 2 et des formules (§2 3 (1)) et (§2 3 (2)) qui sont

$\forall x\ (A \& B) \equiv \forall x\ A \& \forall x\ B$ et $\exists x(A \vee B) \equiv \exists x\ A \vee \exists x\ B$.

Pour les deux formules restantes de ce groupe, montrons par exemple :

$$x < y \equiv x \leqslant y \quad x \neq y .$$

Du lemme technique 2 et du fait que $\forall a\ (a > 0 \supset |a| = a)$ on démontre aisément $x < y \supset x \neq y$.

On a également $x < y \supset x \leqslant y$ car $x < y$ donne $\exists n\ (y_n - x_n > 2^{-n})$.

Or : $x_m - y_m = x_m - x_n + x_n - y_n + y_n - y_m$.

Or : $x_m - x_n \leqslant 2^{-\min(m+1,n+1)}$ (définition du régulateur de convergence).

Donc : $x_m - y_m \leqslant 2.2^{-\min(m+1,n+1)} + x_n - y_n \leqslant 2^{-\min(m,n)} - 2^{-n} \leqslant 2^{-m}$.

Et ceci pour chaque m .

Donc $\forall m\ (x_m - y_m \leqslant 2^{-m})$ d'où $x \leqslant y$ et donc $x < y \supset x \leqslant y$.

Donc en vertu de l'axiome (§2.3.8) $x < y \supset x \leqslant y \& x \neq y$.

On a vu que $x \neq y \equiv x < y \vee x > y$ et que $x < y \equiv \neg(x > y)$.

Or on sait (voir par exemple [5] p.50) que $((\neg A \& (A \vee B)) \supset B)$.

Donc $(\neg(x > y) \& (x < y \vee x > y)) \supset x < y$.

Les formules du groupe (4) se déduisent de celles du groupe (2) par la formule $\neg(A \& \neg A)$, la formule $\neg(x < y \quad x > y)$ se déduisent par contraposition de $(x < y \& x > y)$ puisque $(x < y \& x > y) \supset (x \leqslant y \& x > y)$ car $x < y \supset x \leqslant y$

d'après ce que nous venons de voir.

Les formules du groupe (6) se déduisent de celles du groupe (4) par la formule $\neg(A \vee B) \equiv (\neg A \& \neg B)$ (§2 3 (6)).

Il nous reste à établir les formules du groupe (5).

Or : $x > y \supset x \geqslant y$ et $x = y \supset x \geqslant y$ (voir définitions).

Donc par l'axiome §2.3.11 $[(A \supset C) \supset ((B \supset C) \supset (A \vee B \supset C))]$ on a :

$x > y \vee x = y \supset x \geqslant y$. De même $x < y \vee x = y \supset x \leqslant y$.

Mais : $x \geqslant y \equiv \neg(x < y)$

$\equiv \neg(x \leqslant y \& x \neq y)$ (groupe (1))

$\equiv \neg(\neg(x > y) \& \neg x = y))$ (groupe (2))

$\equiv \neg\neg(x > y \vee x = y)$ $(\neg(A \vee B) \equiv \neg A \& \neg B)$. □.

On déduit immédiatement des formules du groupe (1) et de la définition que la relation $\leqslant$ (resp. $\geqslant$) est une relation d'ordre.

En regardant les formules du groupe (5), on peut se demander si on a ou pas la formule $x \geqslant y \equiv x > y \vee x = y$.

On montre en fait que :

Théorème 2 (cf.[ ]).

$\neg\forall xy(x \geqslant y \vee x \leqslant y)$ $\neg\forall xy(x \leqslant y \vee x > y)$

$\neg\forall xy(x \geqslant y \vee x < y)$ $\neg\forall xy(x = y \vee x < y \vee x > y)$.

La démonstration de ce théorème résulte immédiatement par contraposition et par addition (voir plus loin) du

Lemme fondamental (6.5 Tseïtin [16])

(7) $\neg\forall x(x \geqslant 0 \vee x \leqslant 0)$.

Preuve du lemme fondamental : x étant défini par une formule normale, à savoir : $\forall k \ell m(\ell, m \geqslant \bar{x}(k) \supset |\underline{x}(m) - \underline{x}(\ell)| < 2^{-k})$

(ou $|\underline{x}(m) - \underline{x}(\ell)| < 2^{-k}$ peut se traduire par une formule élémentaire du type $(\langle H \rangle(m\ell k) = \Lambda)$ puisque $\underline{x}(m)$ et $\underline{x}(\ell)$ sont rationnels) une reformulation constructive de

(8) $\forall x \ (x \geqslant 0 \vee x \leqslant 0)$

est donnée par :

$\exists \tau \ \forall x((\langle I_0 \rangle(\tau(x)) = \Lambda) \& (\langle W_\tau \rangle(\tau(x)) = \Lambda) \& ((\tau(x) = 0|) \supset x \geqslant 0)$
$\& ((\tau(x) = 0||) \supset x \leqslant 0))$ (cf. §2.6).

Soit $\mathbb{A}$ un algorithme vérifiant cette propriété. Alors $\mathbb{A}$ est applicable à tout réel, et $\mathbb{A}$ transforme les $x > 0$ en $0|$, car $x > 0 \equiv \neg(x \leqslant 0) \supset \neg(\tau(x) = 0||)$, et pour la même raison les $x < 0$ en $0||$.

On peut donc construire un algorithme $\mathbb{K}$ sur N_3 applicable à tous les réels, transformant en le mot vide tous les réels positifs (c.à.d. tels que $x > 0$), mais ne transformant aucun réel négatif (c.à.d. tel que $x < 0$) en le mot vide.

On construit alors un algorithme $\mathfrak{A}$ sur N_1 tel que :

(i) pour tout mot H auquel $\mathfrak{A}$ est applicable, $\mathfrak{A}(H) = \Lambda$ ou $\mathfrak{A}(H) = -$

(ii) il est impossible de construire un algorithme $\mathfrak{B}$ applicable à tous les mots dans N_0 tel que si $\mathfrak{A}(H) = \Lambda$ alors $\mathfrak{B}(H) = \Lambda$ et si $\mathfrak{A}(H) = -$ alors $\mathfrak{B}(H) \neq \Lambda$..

En effet, on peut construire aisément un algorithme $\mathfrak{A}$ sur N_1 tel que

$$\mathfrak{A}(H) \simeq \begin{cases} \Lambda & \text{si} \quad (H*H) \neq \Lambda \\ - & \text{si} \quad (H*H) = \Lambda \end{cases}$$

et $!\mathfrak{A}(H)$ si et seulement si est applicable à H, étant un algorithme universel sur $\{0,1\}$ correspondant aux algorithmes définis par le théorème 1' du point 6 du §1.

Si $\mathfrak{B}$ est tel que pour tout H dans N_0 on ait $!\mathfrak{B}(H)$ et si $\mathfrak{A}(H) \neq \Lambda$ alors $\mathfrak{B}(H) \neq \Lambda$ et $\mathfrak{B}(H) \neq \Lambda$ si $\mathfrak{A}(H) = -$, considérons $\mathfrak{B}_1$, la traduction de $\mathfrak{B}$ dans l'extension standard de N_0. (C.à.d. les mots dans N_0 restent inchangés par la traduction considérée). Alors $\mathfrak{B}_1$ a les mêmes propriétés que $\mathfrak{B}$ par rapport à $\mathfrak{A}$. Soit alors $H_1 = \varepsilon B_1 \mathfrak{Z}$. On a donc $\mathfrak{B}_1(H) = (H_1 * H)$ puisque $!\mathfrak{B}_1(H)$. Donc $\mathfrak{A}(H_1)$ a un sens et $\mathfrak{A}(H_1) = \Lambda$ ou $\mathfrak{A}(h_1) = -$.

Si $\mathfrak{A}(H_1) = \Lambda$ alors $\mathfrak{B}_1(H_1) = \Lambda$ ce qui est impossible car $\mathfrak{A}(H_1) = \Lambda$ si $(H_1 * H_1) = \mathfrak{B}_1(H_1) \neq \Lambda$. On a la même contradiction si $\mathfrak{A}(H_1) = -$.

Considérons maintenant pour $\mathbb{B}$ l'algorithme de développement canonique de $\mathfrak{A}$ (cf. §2.5.1 démonstration de la proposition 2). On a : pour tout mot H dans N_0 et tout entier naturel n, $\mathbb{B}(H \square n)$ a un sens et si $\mathbb{B}(H \square n) = \Lambda$, alors pour tout $m \geqslant n$ $\mathbb{B}(H \square m) = \Lambda$ et $\mathfrak{A}$ est applicable à H si et seulement si il existe un entier naturel n tel que $\mathbb{B}(H \square n) = \Lambda$.

On construit alors un algorithme $\mathbb{C}$ défini par :

$$\mathbb{C}(H) \simeq \mu n(\mathbb{B}(H \square n) = \Lambda)$$

c.à.d. $\mathbb{C}(H)$ est applicable à H si et seulement si il existe un entier n tel que $\mathbb{B}(H \square n) = \Lambda$, auquel cas $\mathbb{C}(H)$ est le plus petit entier naturel m pour lequel $\mathbb{B}(H \square m) = \Lambda$. On peut construire un tel algorithme à l'aide du

théorème 2 de §1.5.6 (théorème de répétition conditionnelle des algorithmes).

On peut alors construire un algorithme $\mathbb{D}$ tel que :

$$\mathbb{D}(H \square n) \simeq \begin{cases} 0 & \text{si} \quad \mathbb{B}(H \square n) \neq \Lambda \\ \mathfrak{A}(H)\dot{0}|.2^{-\mathbb{C}(H)} & \text{si} \quad \mathbb{B}(H \square n) = \Lambda . \end{cases}$$

Notons . (cf. [ ]) $\tilde{\mathbb{D}}_{H_\square}$ l'algorithme obtenu à partir de $\mathbb{D}$ en fixant H . $\tilde{\mathbb{D}}_{H_\square}$ est du type $(H \to p)$, car $\mathbb{B}(H \square n) = \Lambda$ implique que $!\mathfrak{A}(H)$ et donc $!\mathbb{C}(H)$ a lieu également.

Soit alors un algorithme $\mathbb{E}$ tel que :

$$\mathbb{E}(H) = \varepsilon \tilde{\mathbb{D}}_{H \square^3} \Diamond H_\tau$$

où H_τ est le code de l'algorithme de schème $\{ \to . \, .$ (On vérifie aisément que l'algorithme $\mathbb{E}$ peut être construit).

Alors pour chaque H , $\mathbb{E}(H)$ est un réel.

En effet : si $\mathbb{D}(H \square n) = 0$, alors pour $m \geqslant n$, si $\mathbb{B}(H \square m) = \Lambda$ on aura $!\mathbb{C}(H)$ et donc $\mathbb{C}(H) > n$ et si $\mathbb{B}(H \square m) \neq \Lambda$ on aura $D(H \square m) = 0$; dans les deux cas pour $m \geqslant n$ $|\mathbb{D}(H \square m)| < 2^{-n}$; si $\mathbb{D}(H \square n) \neq 0$, alors pour $m \geqslant n$ $\mathbb{D}(H \square m) = \mathbb{D}(H \square n)$ et donc si k et $\ell \geqslant n$ on a $|\mathbb{D}(H \square k) - \mathbb{D}(H \square \ell)| < 2^{-n}$.

Remarquons alors que si $\mathfrak{A}(H) = \Lambda$, alors $\mathbb{E}(H) = 2^{-\mathbb{C}(H)} > 0$ et si $\mathfrak{A}(H) = -$, alors $\mathbb{E}(H) = -2^{-\mathbb{C}(H)} < 0$; de plus si $\mathfrak{A}$ n'est pas applicable à H , alors $\mathbb{E}(H) = 0$.

Considérons alors l'algorithme :

$$\mathbb{F}(H) \simeq \mathbb{K}(\mathbb{E}(H)).$$

$\mathbb{E}$ est applicable à H pour tout H dans N_0 et il en est donc de même pour $\mathbb{F}$ (car $\mathbb{K}$ est applicable à tous les réels). De plus si $\mathcal{Q}(H) = \Lambda$, alors $\mathbb{E}(H) > 0$ et donc $\mathbb{K}(\mathbb{E}(H)) = \mathbb{F}(H) = \Lambda$ et si $\mathcal{Q}(H) = -$, alors $\mathbb{E}(H) < 0$ et donc $\mathbb{K}(\mathbb{E}(H)) = \mathbb{F}(H) \neq \Lambda$. Or $\mathbb{F}$ ne peut exister d'après la construction de $\mathcal{Q}$. Donc $\mathbb{K}$ lui-même ne peut être construit. D'où l'assertion (7). □.

3. Opérations sur les nombres réels constructifs.

3.1. Module d'un nombre réel.

Considérons x un réel. On construit à partir de x un mot y dans N_3 de la forme $P \diamond Q$ de telle sorte que :

$$\forall n \ \langle P \rangle(n) = |\underline{x}(n)|$$

$$\forall n \ \langle Q \rangle(n) = \overline{x}(n)$$

En fait, on peut construire deux algorithmes $\mathbb{M}_1$ et $\mathbb{M}_2$ sur N_3 tels que pour tous les mots de la forme $P \diamond Q$ dans N_3 on ait $!\mathbb{M}_1(P \diamond Q)$ et $!\mathbb{M}_2(P \diamond Q)$ avec de plus : $\mathbb{M}_1(P \diamond Q)$ et $\mathbb{M}_2(P \diamond Q)$ sont des mots dans $\{0,1\}$, codes d'algorithmes sur les entiers naturels vérifiant les propriétés :

$$\forall n \ (\langle \mathbb{M}_1(P \diamond Q) \rangle(n) \simeq \mathbb{M}(\langle P \rangle(n)))$$

$$\forall n \ (\langle \mathbb{M}_2(P \diamond Q) \rangle(n) \simeq \langle Q \rangle(n))$$

$\mathbb{M}$ étant un algorithme de construction de la valeur absolue d'un nombre rationnel (on peut par exemple convenir que si $!\langle P \rangle(n)$, $\mathbb{M}$ n'est applicable à $\langle P \rangle(n)$ que si ce mot est un nombre rationnel).

Il est clair qu'à l'aide des théorèmes établis sur les algorithmes, en particulier le théorème de l'algorithme universel, on peut construire $\mathbb{M}_1$ et $\mathbb{M}_2$.

Lorsque x est un réel, comme pour tous rationnels a et b on a $||a|-|b|| \leqslant |a-b|$, il est clair que $\mathbb{M}_1(x) \Diamond \mathbb{M}_2(x)$ est un réel qu'on désignera par $|x|$.

On écrira donc :

$$\forall n \ (\underline{|x|}(n) = |\underline{x}(n)|)$$

$$\forall n \ (\overline{|x|}(n) = \overline{x}(n))$$

Pour les autres opérations, c'est sous cette forme qu'on donnera leurs définitions, étant entendu qu'on peut en fait définir des algorithmes plus généraux correspondant aux opérations indiquées.

On remarquera que pour tout rationnel a

$$\mathbb{M}_1(a) \Diamond \mathbb{M}_2(a) = |a|$$ (au sens de l'égalité entre nombres réels), ce qui justifie la notation introduite.

3.2. Somme et différence de nombres réels.

Soient x et y deux réels. On définit leur somme (resp. leur différence par les relations :

$$\underline{x+y}(n) = \underline{x}(n) + \underline{y}(n) \qquad [\text{resp. } \underline{x-y}(n) = \underline{x}(n) - \underline{y}(n)]$$

$$\overline{x+y}(n) = \max(\overline{x}(n+1), \overline{y}(n+1)) \quad [\text{resp. } \underline{x-y}(n) = \max(\overline{x}(n+1), \overline{y}(n+1))]$$

qui permettent aisément de vérifier que les mots dans N_3 :

$\varepsilon \ \underline{x+y} \ \mathfrak{z} \Diamond \varepsilon \ \overline{x+y} \ \mathfrak{z}$ et $\varepsilon \ \underline{x-y} \ \mathfrak{z} \Diamond \varepsilon \ \overline{x-y} \ \mathfrak{z}$ sont des réels et que les opérations ainsi définies correspondent aux opérations définies sur les rationnels. En particulier il est clair que pour tous rationnels a et b :

$$\tau(a) \Diamond \mathfrak{b} + \tau(b) \Diamond \mathfrak{b} = \tau(a+b) \Diamond \mathfrak{b}$$

$$\tau(a) \Diamond \mathfrak{b} - \tau(b) \Diamond \mathfrak{b} = \tau(a-b) \Diamond \mathfrak{b}$$

A l'aide de ces deux opérations on peut définir le maximum et le minimum de deux réels x et y :

On pose :

$$\max(x \square y) = \tfrac{1}{2}(x+y) + \tfrac{1}{2}|x-y|$$

$$\min(x \square y) = \tfrac{1}{2}(x+y) - \tfrac{1}{2}|x-y|$$

Alors pour tous rationnels a et b :

$$\max(a \square b) = \max(a,b)$$

$$\min(a \square b) = \min(a,b)$$

On peut d'ailleurs définir ces opérations directement par :

$$\underline{\max(x \square y)}(n) = \max(\underline{x}(n),\underline{y}(n))$$

$$\overline{\max(x \square y)}(n) = \max(\overline{x}(n+1),\overline{y}(n+1))$$

$$\underline{\min(x \square y)}(n) = \min(\underline{x}(n),\underline{y}(n))$$

$$\overline{\min(x \square y)}(n) = \max(\overline{x}(n+1),\overline{y}(n+1)) \ .$$

Enfin on peut aisément définir la somme d'un nombre quelconque fini de réels.

3.3. Multiplication de nombres réels.

Soient x et y deux nombres réels. On définit leur produit à l'aide des relations :

$$\underline{x.y}(n) \doteq \underline{x}(n)\underline{y}(n)$$

$$\overline{x.y}(n) \doteq \max(\overline{x}(0),\overline{y}(0),\overline{x}(n+1+D),\overline{y}(n+1+D))$$

où $\quad D \doteq \mu i(\max(|\underline{x}(\overline{x}(0))|+1, |\underline{y}(\overline{y}(0))|+1) \leqslant 2^i)$.

L'existence de D résulte aussitôt du fait que toute suite de Cauchy est bornée :

si $k \geqslant \bar{x}(0)$ $\quad |\underline{x}(k)| < |\underline{x}(\bar{x}(0))| + 1$

$k \geqslant \bar{y}(0)$ $\quad |\underline{y}(k)| < |\underline{y}(\bar{y}(0))| + 1$

On voit alors aisément que cette opération coïncide sur les rationnels avec la multiplication habituelle.

3.4. Division d'un réel par un réel non nul.

On peut soit procéder directement comme dans [1], soit définir l'inverse d'un réel non nul, et se ramener ainsi à la multiplication. C'est ce que nous allons faire ici.

La construction repose sur le

Lemme (cf. G.S. Tseïtine [15]). On peut construire des algorithmes $\mathbb{D}^-$, $\mathbb{D}^+$ et $\mathbb{G}$ vérifiant les propriétés suivantes :

(i) $\mathbb{D}^-$ et $\mathbb{D}^+$ sont du type $(\mathbb{A}H \to p)$ et satisfont aux relations :

$$\mathbb{D}^-(x \sqsubset n) < \underline{x} < \mathbb{D}^+(x \sqsubset n)$$

$$\mathbb{D}^+(x \sqsubset n) - \mathbb{D}^-(x \sqsubset n) < 2^{-n}$$

(ii) $\mathbb{G}$ est du type $(\mathbb{A} \overset{\circ}{\to} H)$. Il est applicable au réel x si et seulement si $x > 0$ auquel cas $x > 2^{-\mathbb{G}(x)}$.

Preuve du lemme : Pour (i), on construit des algorithmes $\mathbb{D}^-$ et $\mathbb{D}^+$ tels que l'on ait : $\quad \mathbb{D}^-(x\ \ n) = \underline{x}(\bar{x}(n+3)) - 2^{-n-2}$

$$\mathbb{D}^+(x\ \ n) = \underline{x}(\bar{x}(n+3)) + 2^{-n-2}$$

(on utilise les théorèmes de l'algorithme universel et de composition des algorithmes normaux).

Les relations indiquées résultent de la construction et du lemme technique 2.

Pour (ii) : il est clair qu'on peut construire un algorithme A du type $(p \to H)$ tel que : $(A(p) = \Lambda) \equiv (p > 0)$.

On définit alors : $B(x \square n) \doteq A(\bar{D}(x \square n) - 2^{-n})$ et on définit G par :

$$G(x) \simeq \mu n\ (B(x \square n) = \Lambda).$$

Si $x > 0$, alors du lemme technique 2 on a qu'on peut trouver un entier naturel k tel que $\underline{x}(\bar{x}(k+1)) > 2^{-k}$ c.à.d. $x_k > 2^{-k}$.

Or, par définition du régulateur de convergence, $x_n - x_k \leqslant 2^{-\min(n+1,k+1)}$.

En prenant $n \geqslant k$ on a donc $2^{-k} < x_n + 2^{-k-1}$ d'où $x_n > 2^{-k-1}$.

Or : $\bar{D}(x\ n) = x_{n+2} - 2^{-n-2}$ donc de $x_{n+2} > 2^{-k-1}$

on déduit : $\bar{D}(x\ n) > 2^{-k-1} - 2^{-n-2}$.

Or : $2^{-k-1} - 2^{-n-2} > 2^{-n}$ si $n \geqslant k+1$.

Donc pour $n \geqslant k+1$ $\bar{D}(x \square n) > 2^{-n}$. Donc $!G(x)$ et même $G(x) \leqslant k+1$. Donc $x > \bar{D}(x \square n) > 2^{-G(x)}$ dans ce cas. Si G est applicable à x, c'est qu'il existe un n tel que $B(x \square n) \doteq \Lambda$ et donc $x > \bar{D}(x \square n) > 2^{-n} > 0$.□.

Du lemme technique 2 on déduit aisément :

(1) $\qquad x \neq 0 \equiv |x| > 0$

et (2) $\qquad x = 0 \equiv |x| = 0$

Donc, du lemme précédent on déduit que :

$$x \neq 0 \supset |x| > 2^{-G(|x|)}.$$

On peut alors définir <u>l'inverse</u> d'un réel $x \neq 0$ par :

$$\underline{x^{-1}}(n) = \begin{cases} (\underline{x}(n))^{-1} & \text{si } \underline{x}(n) \neq 0 \\ 0 & \text{xi } \underline{x}(n) = 0 \end{cases}$$

$$\overline{x^{-1}}(n) = \max(\bar{x}(n+2G(|x|)), \bar{x}(G(|x|)))$$

car si $n_o = \mathbb{G}(|x|)$ $\quad x_{n_o+2} > 2^{-n_o-2} + 2^{-\mathbb{G}(|x|)}$

et donc si $k \geqslant \overline{x}(n_o)$ $\quad \underline{x}(k) = x_{n_o+2} + (\underline{x}(k) - x_{n_o+2}) > 2^{-\mathbb{G}(|x|)}$.

De là on tire $\left|\frac{1}{\underline{x}(k)} - \frac{1}{\underline{x}(\ell)}\right| < 2^{-n}\, 2^{-2\,\mathbb{G}(|x|)} |\underline{x}(k)||\underline{x}(\ell)| < 2^{-n}$ pour $k, \ell \geqslant \overline{x^{-1}}(n)$,

et donc $|\underline{x^{-1}}(k) - \underline{x^{-1}}(\ell)| < 2^{-n}$.

Donc le mot $\varepsilon\, \underline{x^{-1}}\, \mathfrak{z} \Diamond \varepsilon\, \overline{x^{-1}}\, \mathfrak{z}$ est un réel, et pour les rationnels on a coïncidence avec la notion habituelle de l'inverse d'un nombre non nul.

On peut alors définir la <u>division</u> du réel x par le réel différent de zéro y par la relation :

$$(3) \qquad x : y = x.y^{-1} \ .$$

3.5. <u>Puissance entière d'un nombre réel</u>.

En utilisant le théorème 1 de §1.5.6 on construit aisément un algorithme $\mathbb{p}$ vérifiant :

$$\mathbb{p}(x\, \square\, 0|) \doteqdot x$$

$$\mathbb{p}(x\, \square\, n+1) \doteqdot \mathbb{p}(x\ \ n).x \qquad\qquad (n \geqslant 1)$$

(rappelons que x est une variable de nombre réel et n une variable d'entier naturel).

Avec l'algorithme $\mathbb{G}$ introduit au point 3.4, on peut exprimer la condition $x \neq 0$ par $!\mathbb{G}(|x|)$. Ceci permet d'exprimer la puissance nulle d'un réel non nul :

$$\mathbb{p}(x\, \square\, 0) \simeq \mathbb{H}_0\, \mathbb{G}(|x|)$$

où $\mathbb{H}$ est un algorithme sur N_3 transformant en $0|$ tous les mots dans l'extension standard de N_3 .

Enfin, pour $x \neq 0$ on peut définir un algorithme de telle façon que

$$\mathbb{p}(x \square -n) \simeq p(x^{-1} \square n) \qquad (n \geqslant 1) .$$

On pose alors $x^n = \mathbb{p}(x \square n)$ (avec ici n entier relatif).

De ce qui a été fait précédemment, il résulte que x^n est toujours un réel (pour x quelconque et n entier naturel positif ou $x \neq 0$ et n entier relatif quelconque).

3.6. Puissance fractionnaire du module d'un nombre réel.

A l'aide des théorèmes sur les algorithmes, on peut construire des algorithmes ζ, θ et χ vérifiant :

$$\zeta(a \square n) \simeq \mu j(a < (j+1)^n)$$

$$\theta(a \square b \square n) \simeq \mu j((b+2^{-j})^n \leqslant a)$$

$$\chi(a \square n \square 0) \simeq \zeta(a \square n)$$

$$\chi(a \square n \square i+1) \simeq \begin{cases} \chi(a \square n \square i) & \text{si } \chi(a \square n \square i)^n = a \\ \chi(a \square n \square i) + 2^{-\theta(a \quad \chi(a \square n \square i) \square n)} & \text{si } \chi(a \square n \square i)^n < a \end{cases}$$

De plus on peut aisément définir des algorithmes dén et num sur N_2 applicables à tout rationnel tels que :

$$\forall a \ (\underline{num}(a)/\underline{dén}(a) = a)$$

$$\forall a \ (\underline{dén}(a) \geqslant 0|)$$

$$\forall a \ (\text{p.g.c.d.}(\underline{dén}(a) , |\underline{num}(a)|) = 0|) .$$

On pose alors Ψ, algorithme sur N_3 tel que :

$$\Psi(a \square b \square n) \simeq \chi(|a|^{\underline{num}(b)} \square \underline{dén}(b) \square n) .$$

On définit alors le réel $|x|^b$ par :

$$\underline{|x|^b}(n) \simeq \Psi(\underline{x}(n) \square b \square n)$$

$$\overline{|x|}^{b}(n) \simeq \max(n+2\ ,\ \bar{x}(0)\ ,\ \bar{x}(R + \underline{dén}(b).(n+2)))$$

avec $R \leftrightarrows \mu j((b|\underline{x}(\bar{x}(0))| + 2)^{\underline{num}\ b-1} \leqslant 2^{j})$.

Dans [1] on trouve la justification de toutes ces définitions et la démonstration du fait que le mot $\varepsilon\ \underline{|x|}^{b}\ni\diamond\varepsilon\overline{|x|}^{b}\ni$ est un réel.

4. Propriétés des opérations sur les nombres réels.

Les opérations introduites jusqu'ici sont des algorithmes des types $(Д \to Д)$ pour le module, $(ДД \to Д)$ pour l'addition et la multiplication, $(ДД \dot{\to} Д)$ pour la division, $(Дц \dot{\to} Д)$ pour la puissance entière, $(Д\ p \dot{\to} Д)$ pour la puissance du module.

Ce sont en fait des algorithmes d'un type particulier, que nous appellerons des fonctions.

Soient $\mu_1,\dots,\mu_k,\mu_{k+1}$ des lettres génériques de variables (on pourra considérer ici que chaque μ_i est une des lettres H , ц , p , Д) (voir §2.6). On dit qu'un algorithme $\mathbb{A}$ de type $(\mu_1\dots\mu_k \dot{\to} \mu_{k+1})$ (resp. de type $(\mu_1\dots\mu_k \to \mu_{k+1})$)] est une fonction du type considéré si :

$$\forall\alpha_1\dots\alpha_k\ \beta_1\dots\beta_k(!\ \mathbb{A}(\alpha_1\ \dots\ \alpha_k)\ \&\ \alpha_1 \underset{\mu_1}{=} \beta_1\ \&\dots\&\ \alpha_k \underset{\mu_k}{=} \beta_k$$
$$\supset !\ \mathbb{A}(\beta_1\ \dots\ \beta_k)\ \&\ \mathbb{A}(\beta_1\ \dots\ \beta_k) \underset{\mu_{k+1}}{=} \mathbb{A}(\alpha_1\ \dots\ \alpha_k))$$

où $\alpha_i \underset{\mu_i}{=} \beta_i$ désigne une relation d'équivalence parmi les mots du genre μ_i , fixée une fois pour toute et qui est alors appelée égalité des mots du genre μ_i (on abandonne la lettre μ_i sous le signe $=$ lorsque aucune ambiguité n'est à craindre).

On peut alors énoncer :

Théorème 1. Les opérations sur les nombres réels construites au point 3 sont des fonctions dont la démonstration ne présente pas de difficulté.

De même, la démonstration du

Théorème 2. L'addition et la multiplication sont associatives et commutatives. La multiplication est distributive par rapport à l'addition. De plus :

$$\forall x\ (x+0 = x) \qquad \forall x\ (x+(-x) = 0)$$

$$\forall xy\ (x+(-y) = x-y) \qquad \forall x\ (x.1 = x)$$

$$\forall x\ (x \neq 0 \supset x.x^{-1} = 1)$$

n'offre aucune difficulté.

Les différences avec les résultats classiques apparaissent dans les propriétés en relation avec l'ordre.

Notons les assertions :

Théorème 3. Pour tous réels x et y on a :

$x \alpha y \equiv (x-y)\alpha 0$ où α est un des symboles $\leqslant$, $\geqslant$, $<$, $>$, $=$, $\neq$.

Preuve : On revient à la définition de la soustraction $(\underline{x-y}(n) = \underline{x}(n) - \underline{y}(n))$ et on applique le lemme technique 2.□

Théorème 4. Pour tous réels x et y on a :

$$(1) \quad x \leqslant \max(x \square y) \quad y \leqslant \max(x \square y) \quad x \geqslant \min(x \square y) \quad y \geqslant \min(x \square y)$$

$$(2) \quad \begin{cases} x \geqslant y \equiv \max(x \square y) = x & x \leqslant y \equiv \max(x \square y) = y \\ x \geqslant y \equiv \min(x \square y) = y & x \leqslant y \equiv \min(x \square y) = x \end{cases}$$

$$(3) \quad \begin{cases} \neg\neg(\max(x \square y) = x \vee \max(x \square y) = y) \\ \neg\neg(\min(x \square y) = x \vee \min(x \square y) = y) \end{cases}$$

(4) $\begin{cases} \neg\forall xy(\max(x \square y) = x \vee \max(x \square y) = y) \\ \neg\forall xy(\min(x \square y) = x \vee \min(x \square y) = y) \end{cases}$

Preuve : Au cours de la démonstration du lemme du point 3.4 on a vu que

(5) $\dot{x} \neq 0 \equiv |x| > 0$

(6) $x = 0 \equiv |x| = 0$

montrons que : (7) $x \geqslant 0 \equiv |x| = x$ (8) $x \leqslant 0 \equiv |x| = -x$

(9) $|x| \geqslant x$ et $|x| \geqslant -x$

En effet, le lemme technique 2 donne :

$$x \geqslant 0 \supset \forall n\ (x_n \geqslant -2^{-n}).$$

Or il est clair que $|x|_n = |x_n|$. Si $x_n \geqslant 0$ alors $|x_n| = x_n$ (x_n est rationnel) et si $x_n \leqslant 0$ on a $|x_n| = -x_n$. Donc $x_n \geqslant -2^{-n} \supset ||x_n|-x_n| \leqslant 2^{-n-1}$. Donc, par le lemme technique 2 (on considère en fait y' défini à partir de y par $\underline{y}'(n) = \underline{y}(n+1)$ et il est alors clair que $y' = y$) on a donc $|x| = x$. Réciproquement, si $|x| = x$, le lemme technique 2 donne $\forall n(||x_n|-x_n| \leqslant 2^{-n})$ si $x_n \geqslant 0$ a fortiori $x_n \geqslant -2^{-n}$. Si $x_n \leqslant 0$ alors $|x_n| = -x_n$ et donc $||x_n|-x_n| = -2x_n$ d'où $x_n \geqslant -2^{-n-1}$. Donc, le même raisonnement donne $x \geqslant 0$. D'où (7). On établit (8) et (9) de la même façon. Pour (1) et (2) on remarque que, en utilisant le théorème 3 :

$\max(x \square y) - x = \frac{1}{2}(x+y) + \frac{1}{2}|x-y| - x = \frac{1}{2}(y-x) + \frac{1}{2}|x-y| \geqslant 0$ en vertu de (9).

On voit de même que (2) découle de (7) et (8) et du théorème 3. (3) résulte de (7), (8), du théorème 3 et de $\forall xy\ \neg\neg(x \geqslant y \vee x \leqslant y)$ (théorème 1 du point 2.3(6)). Il est clair que d'après (7) et (8) (4) est une reformulation de

(10) $\neg\forall xy\ (x > y \vee x \leqslant y)$.

Or $\forall xy\ (x \geqslant y \vee x \leqslant y) \supset \forall x\ (x \geqslant 0 \vee x \leqslant 0)$

en appliquant l'axiome §2.3.14. Le lemme fondamental donne donc (10) par contraposition. D'où les formules (4). □.

En ce qui concerne l'addition et la multiplication on a :

Théorème 5. $x \geqslant 0 \,\&\, y \geqslant 0 \supset x+y \geqslant 0$

pour tous réels : $x > 0 \,\&\, y \geqslant 0 \supset x+y > 0$

x et y on a

$$x+y \neq 0 \supset x \neq 0 \vee y \neq 0$$

$$x \neq 0 \,\&\, y \neq 0 \equiv x.y \neq 0$$

$$x = 0 \vee y = 0 \supset x.y = 0$$

$$x.y = 0 \supset \neg\neg(x = 0 \vee y = 0)$$

$$\neg\forall xy(x.y = 0 \supset x = 0 \vee y = 0)$$

Preuve. Démontrons simplement :

(a) $x.y \neq 0 \equiv x \neq 0 \,\&\, y \neq 0$

(b) $x.y = 0 \supset \neg\neg(x = 0 \vee y = 0)$

(c) $\neg\forall xy(x.y = 0 \supset x = 0 \vee y = 0)$.

Remarquons que (b) est un corollaire de (a) : la formule §2.3.(5) montre que

$$(b) \equiv \neg(x = 0 \vee y = 0) \supset \neg(x.y = 0).$$

Donc par la formule §2.3.(6)

$$(b) \equiv (\neg(x = 0) \,\&\, \neg(y = 0)) \supset \neg(x.y = 0) \equiv (x \neq 0 \,\&\, y \neq 0) \supset x.y \neq 0\ .$$

Dans (a) l'implication $x \neq 0 \,\&\, y \neq 0 \supset x.y \neq 0$ est immédiate. Supposons $x.y \neq 0$.

Par définition on a : $\exists k\ell\ \forall n\ (m \geqslant \ell \supset |\underline{x.y}(m)| \geqslant 2^{-k})$

C.à.d. $|\underline{x}(m)\underline{y}(m)| \geqslant 2^{-k}$ et donc $|\underline{x}(m)||\underline{y}(m)| \geqslant 2^{-k}$.

Or si on a aussi $\ell \geqslant \bar{y}(0)$ on a : $|\underline{y}(m)| \leqslant |\underline{y}(\bar{y}(0))| + 1$.

C.à.d. $2^{-k} \leqslant |\underline{x}(m)|2^{D}$ où $D \leftrightarrows \mu j(|\underline{y}(\bar{y}(0))| + 1 \leqslant 2^{j})$.

D'où $m \geqslant \max(\ell, \bar{y}(0)) \supset |\underline{x}(m)| \geqslant 2^{-k-D}$ d'où $x \neq 0$.

On établit de même que $y \neq 0$.

Démontrons (c).

Supposons : (c') $\forall xy\ (x.y = 0 \supset x = 0 \vee y = 0)$.

On a donc $\forall x\ ((|x|-x).(|x|+x) = 0 \supset |x|-x = 0 \vee |x|+x = 0)$.

Or : $(|x|-x).(|x|+x) = |x|^2 - x^2$ (en utilisant les définitions et théorèmes précédents).

De plus $|x|^2 = |x|.|x|$, donc $\underline{|x|}^2(n) = \underline{|x|}(n).\underline{|x|}(n) = |\underline{x}(n)|^2 = (\underline{x}(n))^2$.

Donc $(|x|-x).(|x|+x) = 0$.

Donc : $(c') \supset \forall x\ (|x|-x = 0 \vee |x|+x = 0)$

On a déjà vu que $|x|-x = 0 \equiv x \geqslant 0$

$|x|+x = 0 \equiv x \leqslant 0$

et donc $(c') \supset \forall x\ (x \geqslant 0 \vee x \leqslant 0)$.

Du lemme fondamental on déduit donc (c). □ .

Nous terminerons ce point par les propriétés relatives aux modules :

Théorème 6. $\forall x\ (|x| \geqslant 0)$

$\forall xy\ |x+y| \leqslant |x| + |y|$

$\forall xy\ |x.y| = |x|.|y|$

$\forall xy\ \big||x| - |y|\big| \leqslant |x-y|$

$\forall x\ (|x| > 0 \supset |x| = x \vee |x| = -x)$

$$\forall x \neg\neg(|x| = x \vee |x| = -x)$$

$$\neg\forall x\ (|x| = x \vee |x| = -x).$$

La démonstration n'offre aucune difficulté après ce qui a été fait dans les précédentes démonstrations.

§4. Quelques propriétés de la droite réelle constructive.

1. Systèmes d'intervalles de la droite réelle.

1.1. On considère l'alphabet $N_4 \leftrightarrows N_3 \cup \{\Delta, \nabla\}$. Dans ce qui suit les lettres grecques α et β seront des variables de nombres réels tout comme x , y , z , etc...

On appelle intervalle tout mot dans N_4 de la forme $P\nabla Q$ où P et Q sont des réels (c.à.d. des mots dans N_3 satisfaisant aux définitions données au point 1 du paragraphe 3) vérifiant la relation $P < Q$. P et Q sont appelés extrémités de l'intervalle $P\nabla Q$, P est dit le début ou extrémité de gauche de l'intervalle, Q est appelé fin ou extrémité de droite de l'intervalle.

On appelle segment tout mot dans N_4 de la forme $P\Delta Q$ où P et Q sont des réels vérifiant la relation $P \leqslant Q$. P et Q sont les extrémités du segment $P\ Q$, P l'extrémité de gauche ou début et Q l'extrémité de droite ou fin.

On appelle continu tout mot dans N_4 qui est soit un intervalle, soit un segment.

Lorsque les extrémités d'un continu sont toutes deux des nombres rationnels, on dira que le continu lui-même est rationnel.

Ainsi : $\alpha \Delta \beta$ et $\alpha \nabla \beta$ représentent respectivement un segment (si $\alpha \leqslant \beta$) et un intervalle (si $\alpha < \beta$) ; $a \Delta b$ avec $a \leqslant b$ est un segment rationnel ; $a \nabla b$ avec $a < b$ est un intervalle rationnel.

Dans le cas d'un segment $\alpha \Delta \beta$ avec $\alpha \leqslant \beta$, on dira qu'il est dégénéré (resp. non dégénéré) si $\alpha = \beta$ (resp. $\alpha < \beta$).

Enfin, on peut associer à un continu un ensemble de réels ou de rationnels :

Soit $\alpha \Delta \beta$ $(\alpha \leqslant \beta)$ un segment. On lui associe l'ensemble $\mathfrak{M}(\alpha \Delta \beta)$ défini par la formule : $(\alpha \leqslant x \,\&\, x \leqslant \beta)$. Dans le cas où on a affaire à un segment rationnel $a \Delta b$ $(a \leqslant b)$, on peut en outre associer l'ensemble $\mathfrak{M}_p(a \Delta b)$ défini par la formule : $(a \leqslant c \,\&\, c \leqslant b)$ (c est la variable de la formule).

Dans le cas d'un intervalle $\alpha \nabla \beta$ $(\alpha < \beta)$ on associe l'ensemble $\mathfrak{M}_{\mathfrak{n}}(\alpha \nabla \beta)$ défini par $(\alpha < x \,\&\, x < \beta)$. Dans le cas d'un intervalle rationnel on peut définir en outre $\mathfrak{M}_p(a \nabla b)$ défini par $(a < c \,\&\, c < b)$.

Dans chacun de ces cas on notera respectivement $x \in \alpha \Delta \beta$, $x \in \alpha \nabla \beta$, $c \in a \Delta b$, $c \in a \nabla b$.

Désignons par K la lettre générique pour désigner les continus. On peut construire des algorithmes que nous appellerons Ext_1 et Ext_2 tels que : pour tout continu $\alpha \varphi \beta$ (où φ désigne la lettre Δ ou la lettre ∇), Ext_i $(i = 1,2)$ soit applicable à $\alpha \varphi \beta$ et :

$$\mathrm{Ext}_1(\alpha \varphi \beta) = \alpha \quad \text{et} \quad \mathrm{Ext}_2(\alpha \varphi \beta) = \beta .$$

Il suffit de prendre les algorithmes de schèmes respectifs :

$$\mathrm{Ext}_1 \begin{cases} \Delta\xi \to \Delta \\ \nabla\xi \to \nabla \\ \Delta \to . \\ \nabla \to . \end{cases} \quad (\xi \in N_4) \qquad \mathrm{Ext}_2 \begin{cases} \xi\Delta \to \Delta \\ \xi\nabla \to \nabla \\ \Delta \to . \\ \nabla \to . \end{cases}$$

1.2. Nous allons définir des relations entre ces objets :

On dit alors que le continu K_1 est <u>inclus</u> ou <u>contenu</u> dans le continu K_2 de <u>même espèce</u> (c.à.d. K_1 et K_2 sont simultanément intervalles ou segments) si on a : $\mathrm{Ext}_1(K_1) \geqslant \mathrm{Ext}_1(K_2)$ et $\mathrm{Ext}_2(K_1) \leqslant \mathrm{Ext}_2(K_2)$ et on note $K_1 \subset K_2$.

On a alors :

<u>Proposition</u> 1. Si K_1 et K_2 sont des continus de même espèce, on a la relation $K_1 \subset K_2$ si et seulement si

$$\mathcal{M}_{\mathcal{A}}(K_1) \subset \mathcal{M}_{\mathcal{A}}(K_2)$$

(voir point 5.1 du paragraphe 2 pour la définition de l'inclusion de deux ensembles).

<u>Preuve</u>. La démonstration est triviale si K_1 et K_2 sont des segments. Si K_1 et K_2 sont des intervalles, on montre trivialement que

$$(K_1 \subset K_2) \supset (\mathcal{M}_{\mathcal{A}}(K_1) \subset \mathcal{M}_{\mathcal{A}}(K_2)) \ .$$

Pour la réciproque, on remarquera que si $\alpha_1 \leftrightarrows \mathrm{Ext}_1(K_1)$. Alors $\alpha_1 + 2^{-n} \in \mathcal{M}_{\mathcal{A}}(K_1)$ pour tout n à partir d'un certain rang, donc $\alpha_1 - \alpha_2 > -2^{-n}$ pour tout n à partir d'un certain rang. Comme $x > 0 \supset x \geqslant 0$, en appliquant le lemme technique 2 du paragraphe précédent on trouve $\alpha_1 - \alpha_2 \geqslant 0$ où $\alpha_2 \leftrightarrows \mathrm{Ext}_1(K_2)$. □.

On peut définir également la réunion et l'intersection de deux continus de même espèce. Désignons par $\mathcal{F}(K)$ la formule de définition de l'ensemble $\mathcal{M}_{\mathcal{A}}(K)$.

Soient K_1 et K_2 deux continus de même espèce. Par définition

$\mathfrak{M}_{\Delta}(K_1) \cap \mathfrak{M}_{\Delta}(K_2)$ est l'ensemble de réels défini par :

$$(\mathfrak{F}(K_1) \,\&\, \mathfrak{F}(K_2))$$

$\mathfrak{M}_{\Delta}(K_1) \cup \mathfrak{M}_{\Delta}(K_2)$ est l'ensemble de réels défini par :

$$(\mathfrak{F}(K_1) \vee \mathfrak{F}(K_2))$$

Ces ensembles sont-ils associés à des continus ?

Proposition 2. Soient K_1 et K_2 désignant des continus de même espèce.

Si on peut trouver un réel x_o appartenant à $\mathfrak{M}_{\Delta}(K_1) \cap \mathfrak{M}_{\Delta}(K_2)$, alors si $K_3 \leftrightarrows \max(\mathbb{E}xt_1) \square \mathbb{E}xt_1(K_2)) \varphi \min(\mathbb{E}xt_2(K_1) \square \mathbb{E}xt_2(K_2))$ (où φ désigne la lettre Δ ou ∇ suivant le genre de continu que représentent K_1 et K_2) on a :

$\mathfrak{M}_{\Delta}(K_1) \cap \mathfrak{M}_{\Delta}(K_2) = \mathfrak{M}_{\Delta}(K_3)$.

Proposition 3. Il n'est pas vrai que l'on a la relation suivante :

$$\mathfrak{M}_{\Delta}(0 \,\Delta\, 0|) \cup \mathfrak{M}_{\Delta}(0| \,\Delta\, 0||) = \mathfrak{M}_{\Delta}(0 \,\Delta\, 0||) .$$

Preuve de la proposition 2. On revient au lemme technique 2 du paragraphe précédent pour montrer que :

$$(x_o \leqslant \beta_1 \,\&\, x_o \leqslant \beta_2) \supset x_o \leqslant \min(\beta_1 \square \beta_2)$$

en prenant par exemple la définition de min(x y) par :

$$\underline{\min(x \square y)}(n) = \min(\underline{x}(n), \underline{y}(n)).$$

Pour le cas d'intervalles on utilise le même type de raisonnement que celui qui a été indiqué dans la démonstration de la proposition 1.

Preuve de la proposition 3. L'assertion indiquée s'écrit :

$$\forall x \; (0 \leqslant x \leqslant 2 \equiv (0 \leqslant x \leqslant 1 \vee 1 \leqslant x \leqslant 2)) .$$

Donc :

$$\forall x\ (\min(x \square 2) \geqslant 0 \supset (0 \leqslant \min(x \square 2) \leqslant 1 \vee \min(x \square 2) \geqslant 1))$$

C'est-à-dire :

$$\forall x\ (x \geqslant 0 \supset 0 \leqslant x \leqslant 1 \vee x \geqslant 1)\ .$$

Si $x_+ \doteq \max(x \square 0)$ on a donc :

$$\forall x\ (x_+ \leqslant 1 \vee x_+ \geqslant 1)$$

qui résulte de $\forall x\ (x \leqslant 1 \vee x \geqslant 1)$ par restriction. Mais on sait que cette assertion est fausse donc les précédentes le sont aussi. $\square$.

De la même manière on établit (voir [ ]) :

Proposition 4. Il n'est pas vrai que :

$$\mathfrak{M}_{\mathfrak{Q}}(0\,\Delta\,0|) = \mathfrak{M}_{\mathfrak{Q}}(0\,\Delta\,0) \cup \mathfrak{M}_{\mathfrak{Q}}(0\ \ 0|) \cup \mathfrak{M}_{\mathfrak{Q}}(0|\,\Delta\,0|)\ .$$

Dans la démonstration de la proposition 3 on peut cependant voir que comme

$$\forall xy\ (x < y \vee x = y \supset x \leqslant y)$$

(cf. Théorème 1 du point 2.3 de §3).

On peut dire que $\mathfrak{M}_{\mathfrak{Q}}(0\,\Delta\,0||)$ contient la réunion des ensembles $\mathfrak{M}_{\mathfrak{Q}}(0\,\Delta\,0|)$ et $\mathfrak{M}_{\mathfrak{Q}}(0|\,\Delta\,0||)$. On a également (même théorème) :

$$\forall xy\ (x \leqslant y \equiv \neg\neg(x > y \vee x = y)).$$

Ce qui fait que le complémentaire de $\mathfrak{M}_{\mathfrak{Q}}(0\,\Delta\,0||)$ est égal à l'intersection des complémentaires de $\mathfrak{M}_{\mathfrak{Q}}(0\,\Delta\,0|)$ et $\mathfrak{M}_{\mathfrak{Q}}(0|\,\Delta\,0||)$.

On remarquera cependant, que dans le cas de continus rationnels si on considère les ensembles de rationnels qui leur sont associés, on peut alors écrire

$\mathfrak{M}_{\mathfrak{p}}(0\,\Delta\,0||) = \mathfrak{M}_{\mathfrak{p}}(0\,\Delta\,0|) \cup \mathfrak{M}_{\mathfrak{p}}(0|\,\Delta\,0||)$ et que

$\mathfrak{M}_{\mathfrak{p}}(0\,\Delta\,0|) = \mathfrak{M}_{\mathfrak{p}}(0\,\Delta\,0) \cup \mathfrak{M}_{\mathfrak{p}}(0\ \ 0|) \cup \mathfrak{M}_{\mathfrak{p}}(0|\,\Delta\,0|)$.

On est donc conduit à définir les opérations de réunion et d'intersection pour des continus rationnels. Cependant la réunion, au sens ordinaire de deux continus n'est pas nécessairement un continu. D'où l'introduction de la notion de système de continus.

1.3. Considérons $N_5 \leftrightarrows N_4 \cup \{*\}$ et φ désignant une des deux lettres ∇ ou Δ, toujours la même dans un même mot, ou dans un même contexte.

On dit qu'un segment $\alpha_1 \Delta \beta_1$ précède le segment $\alpha_2 \Delta \beta_2$ si $\beta_1 < \alpha_2$ et qu'un intervalle $\alpha_1 \nabla \beta_1$ précède l'intervalle $\alpha_2 \nabla \beta_2$ si $\beta_1 \leqslant \alpha_2$.

On appelle système de continus du même genre les mots dans N_5 définis par les règles génératives suivantes :

(i) tout continu est un système de continus du même genre.

(ii) si C est un système de continus du même genre, K un continu du même genre que les continus de C, et si K est précédé par chacun des continus de C, alors le mot $C*K$ est un système de continus du même genre.

La lettre générique des systèmes de continus du même genre est la lettre C.

On dit qu'un système de continus du même genre C_1 est contenu dans un système C_2 du même genre que C_1, et on écrit $C_1 \subset C_2$ si et seulement si pour chaque continu figurant dans C_1, on peut trouver un continu figurant dans C_2 qui le contienne.

On définira alors l'intersection et la réunion de deux systèmes rationnels du même genre comme étant le système dont l'ensemble rationnel associé est l'intersection (resp. la réunion) des ensembles rationnels associés aux systèmes initiaux.

A la notion de système de continus, on peut associer la notion de partition d'un continu, qui revient à écrire ce continu sous la forme d'un système de continus contenant des segments dégénérés et en utilisant une définition de système de continus ne faisant pas intervenir le genre de ces continus. En fait, cette notion est adéquate dans le cas d'un segment.

On appellera cortège de réels (ou de rationnels) les mots dans N_5 définis par les règles génératives suivantes :

(i) tout réel (resp. rationnel) est un cortège de réels (resp. de rationnels)

(ii) si le mot P est un crotège de réels (resp. de rationnels) et α un réel (resp. a un rationnel) majorant chacun des nombres figurant dans P, alors le mot $P*\alpha$ (resp. $P*a$) est un cortège de réels (resp. de rationnels).

Soit maintenant un segment $\alpha \Delta \beta$ (resp. $a \Delta b$ un segment rationnel). On appelle partition de ce segment tout cortège de réels (resp. de rationnels) $\alpha_1 * \ldots * \alpha_n$ (resp. $a_1 * \ldots * a_n$) vérifiant la propriété : $\alpha_1 = \alpha$ (resp. $a_1 = a$) et $\alpha_n = \beta$ (resp. $a_n = b$).

Ainsi on ne considèrera comme partition d'un segment rationnel, qu'une partition rationnelle.

Un problème qu'on peut se poser au sujet d'une partition, est de savoir si on peut localiser les points du segment par rapport à la partition. Il n'y a pas de problèmes pour les points rationnels. Quant aux points réels on sait qu'il faut tenir compte de l'assertion $\neg\forall x\ (x \geqslant 0 \vee x \leqslant 0)$.

Cependant on a (voir [17]) :

Lemme technique 1. Soient α et β deux réels tels que l'on ait $\alpha < \beta$. Alors :

$$\forall x \quad (x > \alpha \vee x < \beta)$$

est vraie.

Preuve. Comme $\alpha < \beta$, on a $\beta - \alpha > 2^{-\mathcal{E}(\beta-\alpha)}$. Comparons $\mathbb{D}^+(x \square n)$ et $\mathbb{D}^-(\beta \square m)$ qui sont des nombres rationnels.

On a : $\mathbb{D}^+(x \square n) = x_{n+2} + 2^{-n-2}$ $\qquad$ $\mathbb{D}^-(\beta \square m) = \beta_{m+2} - 2^{-m-2}$

$$\mathbb{D}^+(x \square n) - \mathbb{D}^-(\beta \square m) = x_{n+2} - \beta_{m+2} + 2^{-n-2} - 2^{-m-2}.$$

Donc $$\mathbb{D}^+(x \square n) - \mathbb{D}^-(\beta \square n+1) = x_{n+2} - \beta_{n+3} + 2^{-n-3}.$$

Si $$\mathbb{D}^+(x \square n) - \mathbb{D}^-(\beta \square n+1) < 0$$

on a $$x_{n+2} < \beta_{n+3} - 2^{-n-3}$$

c'est-à-dire en vertu du lemme technique 2 de § 3 $x < \beta$

si $$\mathbb{D}^+(x \square n) - \mathbb{D}^-(\beta \square n+1) \geqslant 0$$

on a : $$x_{n+2} \geqslant \beta_{n+3} - 2^{-n-3}.$$

Mais $$\mathbb{D}^-(x \square n) < x < \mathbb{D}^+(x \square n)$$

et $\mathbb{D}^+(x \square n) - \mathbb{D}^-(x \square n) < 2^{-n}$ on tire :

$$x < \mathbb{D}^-(x \square n) + 2^{-n} \qquad \text{et} \qquad x > \mathbb{D}^+(x \square n) - 2^{-n}.$$

Donc : $$\mathbb{D}^-(\beta \square n+1) - \mathbb{D}^+(\alpha \square n+1) > \beta - 2^{-n-1} - \alpha - 2^{-n-1} = \beta - \alpha - 2^{-n}.$$

Donc si $$\mathbb{D}^-(\beta \square n+1) - \mathbb{D}^+(\alpha \square n+1) > 2^{-k}$$

on aura $$\mathbb{D}^+(x \square n) - \mathbb{D}^+(\alpha \square n+1) > 0 \quad \text{et donc} \quad x > \alpha.$$

Il suffit donc de prendre $n = \mathcal{E}(\beta - \alpha) + 1$.

La condition $\mathbb{D}^+(x \square n) - \mathbb{D}^-(\beta \square n+1) > 0$ se vérifie aisément par un algorithme. On peut donc savoir si on a $x > \alpha$ ou si on a $x > \beta$. Donc, d'après le principe (i) du point 4 de § 2. l'assertion annoncée est donc vraie. □.

Corollaire. Il existe un algorithme $\mathbb{C}$ applicable à tout triple de réels x, α, β avec la condition $\alpha < \beta$ et tel que l'on ait :

$$!\mathbb{C}(x \square \alpha \square \beta) \supset \langle w_2 \rangle\ (\mathbb{C}(x \square \alpha \square \beta)) \doteq \Lambda$$
$$\mathbb{C}(x \square \alpha \square \beta) \doteq 0| \supset x > \alpha$$
$$\mathbb{C}(x \square \alpha \square \beta) \doteq 0|| \supset x < \beta .$$

La preuve est immédiate.

Ce corollaire va nous permettre d'étudier la position d'un réel appartenant à un segment non dégénéré, par rapport aux composantes d'une partition $\alpha_1 * \ldots * \alpha_n$ vérifiant $\alpha_i < \alpha_{i+1}$ pour $i = 1, \ldots, n-1$ $(n \geqslant 2)$.

Théorème 1 (voir [17]). Soit $\alpha \Delta \beta$ un segment non dégénéré et $\alpha_1 * \ldots * \alpha_n$ une partition de ce segment vérifiant :

$$\alpha_i < \alpha_{i+1} \qquad i = 1, \ldots, n-1 \qquad (n \geqslant 3).$$

Alors, pour tout x appartenant à $\alpha \Delta \beta$, on peut trouver un entier m avec $1 \leqslant m \leqslant n-2$ tel que

$$\alpha_m \leqslant x \leqslant \alpha_{m+2} .$$

Preuve. On sait que pour tout k avec $1 \leqslant k \leqslant n-1$ on a :

$$\forall x \quad (x > \alpha_k \vee x < \alpha_{k+1})$$

et, plus précisément :

$$\forall x \quad (!\mathbb{C}(x \square \alpha_k \square \alpha_{k+1}) \,\&\, \langle w_2 \rangle\ (\mathbb{C}(x \square \alpha_k \square \alpha_{k+1})) \doteq \Lambda$$
$$\&\ (\mathbb{C}(x \square \alpha_k \square \alpha_{k+1}) \doteq 0| \supset x > \alpha_k)$$
$$\&\ (\mathbb{C}(x \square \alpha_k \square \alpha_{k+1}) \doteq 0|| \supset x < \alpha_{k+1}))$$

si $\mathbb{C}(x \square \alpha_1 \square \alpha_2) \doteq 0||$ on prend $m = 1$

si pour tout $k \leqslant n-2$ $\mathbb{C}(x \square \alpha_k \square \alpha_{k+1}) \doteq 0|$ on prend $m = n-2$

si on n'est dans aucun de ces cas on prend

$$m = \mu k(\mathbb{C}(x \square \alpha_{k+1} \square \alpha_{k+2}) \doteq 0||).$$

On peut remarquer que dans ce dernier cas, on aura en fait $\alpha_m < x < \alpha_{m+2}$ ce qui fait qu'on ne peut pas atteindre le résultat $\alpha_m \leqslant x \leqslant \alpha_{m+1}$. □.

2. Suites de nombres réels.

2.1. On appelle suite de nombres réels (en abrégé suite de réels) tout algorithme du type $(H \to \Pi)$.

Si Φ est une suite de réels, on appelle régulateur de convergence de la suite Φ tout algorithme Ψ du type $(H \to H)$ tel que :

$$\forall k\, \ell\, m \quad (\ell, m \geqslant \Psi(k) \supset |\Phi(\ell) - \Phi(m)| < 2^{-k}).$$

Pour simplifier les notations, on notera Φ_ℓ au lieu de $\Phi(\ell)$, $\underline{\Phi}_\ell(n)$ au lieu de $\underline{\Phi(\ell)}(n)$, $(\Phi_\ell)_n$ pour $(\underline{\Phi(\ell)})_n \leftrightarrows \underline{\Phi(\ell)}(\overline{\Phi(\ell)}(n+1))$, etc...

On appelle suite de Cauchy dans la droite réelle, tout couple (Φ, Ψ) d'algorithmes dont le premier élément est une suite de réels et le second, un régulateur de convergence de la suite ainsi définie.

Soit Φ une suite de réels. On dit que le réel x est limite de la suite Φ, et on note $x = \lim \Phi$ si on peut construire un algorithme Ψ du type $(H \to H)$ tel que :

$$\forall k\, \ell \qquad (\ell \geqslant \Psi(k) \supset |\Phi_\ell - x| < 2^{-k}).$$

On peut alors énoncer :

Théorème 1. Pour toute suite de Cauchy, on peut construire un réel x qui est limite de cette suite ; toutes les limites d'une même suite de Cauchy sont égales.

En fait nous allons démontrer mieux :

"On peut construire un algorithme noté $\mathbb{lim}$ et appelé algorithme de passage à la limite, applicable à toute suite de Cauchy et tel que l'on ait alors $\mathbb{Lim}\,\Phi = \lim \Phi$."

Preuve. On utilise simplement la méthode diagonale de Cantor. A l'aide du théorème de l'algorithme universel, du théorème de composition des algorithmes on peut construire $\mathbb{lim}$ ainsi :

$$\underline{\mathbb{lim}\,\Phi}(n) \simeq \underline{\Phi}_n(\overline{\Phi}_n(n))$$

et

$$\overline{\mathbb{lim}\,\Phi}(n) \simeq \max(\Psi(n+2), n+2).$$

On vérifie sans difficultés que cet algorithme satisfait aux conditions du théorème. □.

Une forme un peu différente du même procédé permet d'établir que la droite réelle constructive n'est pas énumérable :

Théorème 2. Pour tous réels α et β vérifiant $\alpha < \beta$, et pour toute suite de réels Φ, on peut construire un réel x tel que

$$\alpha \leqslant x \leqslant \beta$$

et

$$\forall n (\Phi_n \neq x).$$

Preuve. On pose $\gamma = \alpha + \frac{\beta - \alpha}{3}$ $\quad \delta = \alpha + 2.\frac{\beta - \alpha}{3}$ et soit y un nombre réel. Nous ne savons pas a priori si $y \in \alpha \Delta \beta$ ou non, ce qui fait que nous ne pouvons pas appliquer le théorème précédent. Etudions cependant $\mathbb{C}(y \sqcap \alpha \sqcap \delta)$ et $\mathbb{C}(y \sqcap \gamma \sqcap \beta)$. Suivant les valeurs obtenues on obtient les conclusions suivantes :

(a)
$$\left.\begin{array}{l} \mathbb{C}(y \sqcap \alpha \sqcap \delta) \neq 0 \\ \mathbb{C}(y \sqcap \gamma \sqcap \beta) \neq 0 \end{array}\right| \supset \begin{array}{l} y > \alpha \\ y < \gamma \end{array} \quad \text{c.à d.} \quad y \notin \alpha \Delta \gamma$$

(b) $\left.\begin{array}{l} C(y \sqcap \alpha \sqcap \delta) \mathrel{\dot{=}} 0 \\ C(y \sqcap \gamma \sqcap \beta) \mathrel{\dot{=}} 0 \end{array}\right| \supset \begin{array}{l} y > \alpha \\ y < \beta \end{array}$ c.à d. $y \in \alpha \Delta \beta$

(c) $C(y \sqcap \alpha \sqcap \delta) \mathrel{\dot{=}} 0 \,||\, \supset y < \delta$ c.à d. $y \notin \delta \Delta \beta$.

Dans le cas b) étudions $\mathbb{C}(y \sqcap \gamma \sqcap \delta)$. Cela donne :

$\mathbb{C}(y \sqcap \gamma \sqcap \delta) \mathrel{\dot{=}} 0 \,|\, \supset y > \gamma$ c.à d. $y \notin \alpha \Delta \gamma$

$\mathbb{C}(y \sqcap \gamma \sqcap \delta) \mathrel{\dot{=}} 0 \,||\, \supset y < \delta$ c.à d. $y \notin \delta \Delta \beta$.

Donc en résumé on peut écrire :

$$\forall y \quad (y \notin \alpha \Delta \gamma \vee y \notin \delta \Delta \beta)$$

(au sens de l'interprétation constructive de cette assertion).

On peut donc construire un algorithme $\mathbb{I}$, applicable à tout triple de réels y, α, β avec $\alpha < \beta$ et tel que : $\mathbb{I}(y \sqcap \alpha \sqcap \beta)$ désigne celui des segments $\alpha \Delta \gamma$ et $\delta \Delta \beta$ auquel y n'appartient pas.

On pose alors : $\alpha_0 \Delta \beta_0 = \alpha \Delta \beta$

$\alpha_1 \Delta \beta_1 \mathrel{\dot{=}} \mathbb{I}(\Phi_0 \sqcap \alpha_0 \sqcap \beta_0)$

$\alpha_{n+1} \Delta \beta_{n+1} \mathrel{\dot{=}} \mathbb{I}(\Phi_n \sqcap \alpha_n \sqcap \beta_n)$.

On peut alors aisément construire un algorithme $\mathbb{R}$ tel que :

$$\forall n \quad (\mathbb{R}(n) \mathrel{\dot{=}} \alpha_{n+1})$$

(on prend l'algorithme $\mathbb{E} \times t_1 \circ \mathbb{I} \circ \mathbb{J}$ où $\mathbb{J}$ est un algorithme tel que :

$\forall n \quad (\mathbb{J}(n) \mathrel{\dot{=}} \Phi_n \sqcap \alpha_n \sqcap \beta_n))$.

Il est évident qu'on peut construire un algorithme Ψ tel que $(\mathbb{R}, \Psi)$ soit une suite de Cauchy. Soit x sa limite.

En utilisant le lemme technique 2 de § 3, on établit aisément que :

$$\forall n \quad (x \in \alpha_n \Delta \beta_n) \text{ et en particulier } x \in \alpha \Delta \beta.$$

Or pour chaque n, $\Phi_n \not\in \alpha_n \Delta \beta_n$ et donc $x = \Phi_n$ donne une contradiction. Donc : $\forall n \quad (x \neq \Phi_n)$. □.

Corollaire (cf. [1]). Il n'existe pas d'algorithme sur N_3 reconnaissant les réels parmi les mots dans N_3.

Preuve. L'idée de la démonstration est que si un tel algorithme existait, à l'aide d'une bijection constructive entre les entiers naturels et les mots dans N_3, on pourrait construire une surjection constructive des entiers sur les réels, et en prenant le segment $0 \Delta 0|$ on obtient une contradiction au théorème précédent.

Désignons par $\mathcal{A}$ un algorithme sur N_3 reconnaissant les réels parmi les mots dans N_3. On a vu dans la démonstration de la proposition 2 du point 5.1 de § 2 comment construire des algorithmes réalisant une bijection constructive des entiers naturels sur les mots dans un alphabet. Nous désignerons Γ un algorithme du type $(H \to N_3)$ (N_3 lettre générique de N_3) et Γ^{-1} son inverse, des algorithmes réalisant cette bijection, et nous supposerons (on peut toujours s'y ramener) que $\Gamma(0) \doteq 0$.

On construit un algorithme Ξ tel que :

$$\Xi(n) \doteq \begin{cases} 1 & \text{si } \Gamma(n) \text{ est un réel (c.à d. } \mathcal{A}(\Gamma(n)) \doteq \Lambda) \\ 0 & \text{si } \Gamma(n) \text{ n'est pas un réel (c.à d. } \mathcal{A}(\Gamma(n)) \neq \Lambda). \end{cases}$$

On pose alors :

$$\theta(0) \doteq 0 \quad \text{et} \quad \theta(n+1) \doteq (n+1)\,\Xi(n+1) + \theta(n)(1 - \Xi(n+1)).$$

Si $\Gamma(\theta(n))$ est un réel, si $\Xi(n+1) \doteq 0$, alors $\theta(n+1) \doteq \theta(n)$ et donc $\Gamma(\theta(n+1))$ est un réel. Si $\Xi(n+1) \doteq 1$, alors $\theta(n+1) \doteq n+1$ et par définition de Ξ, $\Gamma(\theta(n+1))$ est un réel. Donc, comme $\theta(0) \doteq 0$, par induction, on voit que $\Gamma(\theta(n))$ est un réel pour chaque n.

Désignons par $\mathbb{E}$ l'algorithme $\Gamma_o\theta$. On a donc $\mathcal{A}(\mathbb{E}(n)) \doteq \Lambda$ pour tout n. Soit x un réel et $n_o \leftrightarrows \Gamma^{-1}(x)$. Or : $\theta(n_o) \doteq n_o$ si $n_o = 0$. Si $n_o \neq 0$ alors

$$\theta(n_o) \doteq \theta(n_o-1)+1) \doteq \begin{cases} n_o & \text{si } \Gamma(n_o) \text{ est un réel} \\ 0(n_o-1) & \text{sinon .} \end{cases}$$

On a donc $\theta(n_o) \doteq n_o$ et donc on peut trouver un n tel que $\mathbb{E}(n) \doteq x$.

Comme les réels sont définis par une formule normale, on vient de démontrer, d'après l'interprétation constructive des assertions :

$$(*) \qquad \forall x\, \exists n \quad (\mathbb{E}(n) \doteq x).$$

Le théorème précédent donne :

$$\exists x \quad (0 \leqslant x \leqslant 1 \ \& \ \forall n \ (x \neq \mathbb{E}(n))$$

car $\mathbb{E}$ est une suite de réels. On a donc a fortiori

$$\exists x\, \forall n \quad (x \neq \mathbb{E}(n))$$

(car $A \& B \supset$ voir § 2.3.(7) et par le principe (ii) de § 2.4.).

Désignons par P ce réel. D'après le principe (ii) de § 2.4. l'assertion

$$\forall n \quad (P \neq \mathbb{E}(n))$$

est vraie. Et d'après (*) et § 2.3.14 on a :

$$\exists n \quad (P = \mathbb{E}(n).$$

Or $$\forall n \ \neg (P = \mathbb{E}(n)) \supset \neg \exists\, n \ (P = \mathbb{E}(n))$$

car on a $$(\forall x \neg \mathfrak{F}(x) \equiv \neg \exists\, x\, \mathfrak{F}(x)) \qquad \text{(cf. § 2.3.(3))}.$$

Donc $\mathcal{A}$ ne peut être construit. □.

2.2. Au sujet des suites de nombres réels, on sait qu'en analyse classique, un des critères de convergence le plus utilisé est le théorème de la borne supérieure pour des

suites croissantes bornées.

Ce théorème n'est pas vrai constructivement :

Théorème 3 (E. Specker). On peut construire un algorithme β du type $(H \to p)$ tel que :

(i) $\forall n \quad (\beta(n) \leqslant \beta(n+1)$

(ii) $\forall n \quad (0 \leqslant \beta(n) < 1)$

(iii) β n'admet aucun régulateur de convergence.

Preuve. Supposons qu'on dispose d'un algorithme α de type $(HH \to H)$ tel que :

$$(*) \qquad \begin{cases} \forall ij \quad (0 \leqslant \alpha(i \square j) \leqslant 1) \\ \forall ij \quad (\alpha(i \square j) \leqslant \alpha(i \square j + 1) \end{cases} \qquad (i \text{ et } j \text{ variables d'entiers naturels})$$

On définit

$$\beta(n) \simeq \sum_{i=0}^{n} \alpha(i \square n).2^{-i-1}$$

il est alors clair que β vérifie (i) et (ii).

Pour algorithme α, nous allons partir de l'algorithme $\mathbb{H}$ construit au théorème 2 du point 6 de 1. Soit $\mathbb{B}$ l'algorithme de développement canonique de $\mathbb{H}$. On sait que $\mathbb{B}$ vérifie :

Pour tout mot H dans N_o et tout entier naturel n on a : $!\,\mathbb{B}(H \square n)$ et si $\mathbb{B}(H \square n) \underset{\cdot}{=} \Lambda$ pour tout $m \geqslant n$ $\mathbb{B}(H \square n) \underset{\cdot}{=} \Lambda$, et $\mathbb{H}$ est applicable au mot H si et seulement si $\exists n \, (\mathbb{B}(H \square n) \underset{\cdot}{=} \Lambda)$. Soit $\mathbb{N}$ un algorithme énumérant sans répétition tous les mots dans $\{0,1\}$ $\mathbb{N}^{-1}$ l'algorithme inverse et soit $\mathbb{M}$ l'algorithme de schème :

$$\begin{cases} \xi \longrightarrow \cdot 0 \quad (\xi \in \text{alph. } \mathbb{B}) \text{ où alph. } \mathbb{B} \text{ désigne l'alphabet de } \mathbb{B}. \\ \dot{\longrightarrow} \cdot 1 \end{cases}$$

On prendra alors (théorèmes de réunion et de composition des algorithmes) :

$$\alpha(i \square j) \_ M(\mathbb{B}(\mathbb{N}(i) \square j)).$$

Supposons maintenant que β admette un régulateur de convergence. Soit $\mathbb{f}$ ce régulateur. On peut supposer que

$$\forall n(\mathbb{f}(n) \geqslant n).$$

Soit $\ell \leftrightarrows \mathbb{f}(n+1)$, n étant fixé et considérons $m \geqslant 1$. On a donc :

$$0 \leqslant \beta(\ell+m) - \beta(\ell) < 2^{-n-1}$$

ou :

$$0 \leqslant \sum_{i=0}^{\ell} (\alpha(i \square \ell+m) - \alpha(i \square \ell)).2^{-i-1}$$

$$+ \sum_{i=\ell+1}^{\ell+m} \alpha(i \square \ell+m).2^{-i-1} < 2^{-n-1} .$$

Comme $\ell \geqslant n+1$ on a :

$$\forall i\ m\ \ (0 \leqslant i \leqslant n \supset \alpha(i \square \ell + m) \underset{\circ}{=} \alpha(i \square \ell))$$

et en particulier, en faisant $i = n$

$$\forall m\ \ (\alpha(n \square \mathbb{f}(n+1) + m) = \alpha(n \square f(n+1))).$$

On peut construire un algorithme $\mathbb{g}$ tel que :

$$\mathbb{g}(n) \simeq \alpha(n \square \mathbb{f}(n+1))$$

et il est clair que $\mathbb{g}$ est du type $(H \to H)$.

En outre, il est clair d'après (*) que l'on a :

$$(**) \qquad \mathbb{g}(n) \underset{\circ}{=} 1 \equiv \exists\, m(\alpha(n \square m) \underset{\circ}{=} 1).$$

Considérons alors $\mathbb{K}$ algorithme sur $\{0,1\}$ défini par :

$$\mathbb{K}(H) \simeq \mathbb{P}(\mathbb{g}(\mathbb{N}^{-1}(H))) \qquad (H \text{ mot dans } \{0,1\}).$$

où $\mathbb{P}$ est l'algorithme de schème :

$$\begin{cases} 0| \to \cdot \\ 0 \to \cdot\, 0 . \end{cases}$$

Alors il est facile de voir que : $!\,\mathbb{K}(H)$ pour tout mot H et

$$\begin{aligned}\mathbb{K}(H) \doteq \Lambda &\equiv \mathcal{S}(\mathbb{N}^{-1}(H)) \doteq 1\\ &\equiv \exists m(\mathcal{X}(\mathbb{N}^{-1}(H)\sqsupset m) \doteq 1)\\ &\equiv \exists m(\mathbb{B}(H\sqsupset m) \doteq \Lambda)\\ &\equiv !\,\mathbb{H}(H).\end{aligned}$$

Or $\mathbb{H}$ a été construit de telle sorte que la réalisation de $\mathbb{K}$ soit impossible. Il en résulte que β ne peut admettre de régulateur de convergence. □.

Cependant, on a le résultat suivant :

Théorème 4. Soit Φ une suite de rationnels telle que l'on ait :

$$\forall n \quad (\Phi_n \leqslant \Phi_{n+1})$$
$$\exists a \forall n \quad (\Phi_n \leqslant a).$$

Alors on a :

$$\forall k \neg\neg \exists \ell \; \forall n\, m \quad (n,m \geqslant \ell \supset |\Phi_n - \Phi_m| < 2^{-k}).$$

Preuve. Sans restreindre la généralité de la situation, on peut supposer $\Phi_o \geqslant 0$.

Soit k un entier naturel.

Supposons que l'on ait :

$$(*) \qquad \neg \exists \ell \; \forall n\, m \quad (n,m \geqslant \ell \supset |\Phi_n - \Phi_m| < 2^{-k}).$$

En vertu de § 2.3.(3) on a :

$$\forall \ell \neg \forall n\, m \quad (n,m \geqslant \ell \supset |\Phi_n - \Phi_m| < 2^{-k})$$

or : $(n,m \geqslant \ell \supset |\Phi_n - \Phi_m| < 2^{-k})$ est de la forme $(A \supset \neg B)$ où $A \leftrightarrows (n,m \geqslant \ell)$ et $B \leftrightarrows (|\Phi_n - \Phi_m| \geqslant 2^{-k})$.

Donc en vertu de § 2.3.(7) la formule (*) est équivalente à

$$\forall \ell \neg \forall n\, m \quad (n,m \geqslant \ell \supset |\Phi_n - \Phi_m| < 2^{-k})$$

qui, de nouveau en vertu de § 2.3(3) est équivalente à

(**) $\forall \ell \neg\neg \exists\, n\ m\ \ (n,m \geqslant \ell\ \&\ |\Phi_n - \Phi_m| \geqslant 2^{-k})$.

Or, n, m et ℓ sont des entiers, Φ_n et Φ_m des rationnels.

Donc si $A \leftrightarrows (n,m \geqslant \ell\ \&\ |\Phi_n - \Phi_m| \geqslant 2^{-k})$ peut se mettre sous la forme $(\langle H\rangle(n\square m) = \Lambda)$ avec $\forall nm!\ \langle H\rangle(n\square m)$. Donc on a $\forall n\ m\ (A \vee \neg A)$. Donc en vertu du principe de Markov, (**) est équivalente à :

(***) $\forall \ell\ \exists\, n\ m\ \ (n,m \geqslant \ell\ \&\ |\Phi_n - \Phi_m| \geqslant 2^{-k})$.

Posons $D \leftrightarrows \mu_j\ (a < 2^{-k}.j)$.

Soit $\ell_o = 0$. On peut donc construire n_o et m_o tels que $n_o, m_o \geqslant 0$ et $|\Phi_{n_o} - \Phi_{m_o}| \geqslant 2^{-k}$.

Soit $\ell_1 = \max(n_o+1\ ,\ m_o+1)$. On peut construire n_1 et m_1 tels que $n_1, m_1 \geqslant \ell_1$ et $|\Phi_{n_1} - \Phi_{m_1}| \geqslant 2^{-k}$, etc...

Considérons enfin $\ell_D = \max(n_{D-1}+1\ ,\ m_{D-1}+1)$ (on remarquera que $D \geqslant 1$). On peut construire n_D et $m_D \geqslant \ell_D$ et tels que

$$|\Phi_{n_D} - \Phi_{m_D}| \geqslant 2^{-k}\ .$$

Dans la suite $n_o, m_o, n_1, m_1, \ldots, n_D, m_D$ on peut toujours supposer que $n_i \leqslant m_i$ $(i = 0, \ldots, D)$.

On a alors :

$$\Phi_{m_D} \geqslant \Phi_{m_D} - \Phi_{n_o} \geqslant \Phi_{m_D} - \Phi_{n_D} + \Phi_{n_D} - \Phi_{m_{D-1}} + \Phi_{m_{D-1}} - \Phi_{n_{D-1}} + \ldots + \Phi_{m_1} - \Phi_{n_1} + \Phi_{n_1} - \Phi_{m_o} + \Phi_{m_o} - \Phi_{n_o}$$

Or, par les conditions sur les ℓ_i, $n_{i+1} \geqslant m_i$, $i = 0, \ldots, D-1$ et par la croissance $\Phi_{n_{i+1}} - \Phi_{m_i} \geqslant 0$.

Donc $$\Phi_{m_D} \geqslant \sum_{i=0}^{D} (\Phi_{m_i} - \Phi_{n_i}) \geqslant (D+1).2^{-k} > a.$$

Ceci contredit l'hypothèse $\exists\, a\, \forall n\ (\Phi_n \leqslant a)$.

Donc la négation de (*) est vraie. □.

Ainsi l'algorithme construit au théorème 3 vérifie deux propriétés :

$$\neg\forall k\, \exists \ell\ \forall m\, n\ (m,n \geqslant \ell \supset |\Phi_n - \Phi_m| < 2^{-k})$$

et

$$\forall k\, \neg\neg\exists \ell\ \forall mn\ \ (m,n \geqslant \ell \supset |\Phi_n - \Phi_m| < 2^{-k})$$

qui sont contradictoires du point de vue de la logique classique.

On peut se demander sous quelles hypothèses supplémentaires on peut conclure à l'existence d'une limite dans ce cas, ou dans le cas d'un ensemble borné.

Il s'agit de la notion d'ensemble totalement borné, que l'on peut définir ainsi : si on appelle intervalle centré en x tout intervalle de la forme $x - a \vee x + a$ où a est un rationnel positif $(a > 0)$, un ensemble $\mathfrak{M}$ de réels est dit totalement borné si, pour tout $a > 0$, on peut construire un cortège de réels appartenant à $\mathfrak{M}$ (cette dernière hypothèse n'est pas nécessaire) tel que pour tout réel x appartenant à $\mathfrak{M}$, on peut indiquer un élément du cortège tel que x appartiennent à l'intervalle $y - a \vee y + a$. On peut alors démontrer (cf. [1]) qu'un tel ensemble admet une borne supérieure et une borne inférieure, de même que toute fonction uniformément continue sur un tel ensemble. Nous n'aborderons pas davantage cette question dans le cadre de cet exposé.

3. Compacité constructive et non-compacité, au sens de Heine-Borel, des segments de la droite réelle constructive.

3.1. On dira qu'un ensemble de réels $\mathfrak{M}$ est compact si :

(i) toute suite de Cauchy dans $\mathfrak{M}$ admet une limite dans $\mathfrak{M}$

(ii) $\mathfrak{M}$ est totalement borné.

On a alors :

Théorème 1. Tout segment non dégénéré $\alpha \Delta \beta$ définit un ensemble de réels $\mathcal{M}_{\Delta}(\alpha \Delta \beta)$ qui est compact.

Preuve. La propriété de complétude découle du fait suivant : si Φ est une suite de réels convergeant vers x et si :

$$\forall n \quad (\alpha \leqslant \Phi_n \leqslant \beta)$$

alors $$\alpha \leqslant x \leqslant \beta.$$

Cela résulte des lemmes techniques 2 et 3 du point 2 de § 3.

La propriété (ii) de la définition résulte du théorème 1 du point 1.3 de ce paragraphe, en choisissant, $a>0$ étant donné des partitions du genre $\alpha * \alpha + \frac{\beta-\alpha}{N} *...$ $..* \alpha + \frac{N-1}{N}(\beta-\alpha) * \beta$ où $N \leftrightarrows \mu n(\frac{\beta-\alpha}{2a} < n)$.

3.2. Cependant, l'ensemble de réels associé à un segment n'est pas compact au sens de la définition classique à l'aide de la propriété de recouvrement fini par des ouverts.

Nous allons suivre ici la démonstration donnée par I. D. Zaslavsky dans [17].

Nous allons en premier lieu établir la

Proposition 1 (I. D. Zaslavsky). On peut construire des algorithmes Φ, $\mathbb{N}$, $\mathbb{G}^-$ et $\mathbb{G}^+$ tels que :

(i) Φ est applicable à tout entier naturel n qu'il transforme en un système de segments rationnels

(ii) $\mathbb{N}$ est applicable à tout réel qu'il transforme en un entier naturel

(iii) $\mathbb{G}^-$ et $\mathbb{G}^+$ sont applicables à tout réel et ils transforment chaque réel en un rationnel

(iv) $\Phi_0 = 0 \Delta 0|$

(v) $\forall n \quad (\Phi_{n+1} \subset \Phi_n)$

(vi) $\forall x \quad (\mathbb{G}^-(x) < x < \mathbb{G}^+(x))$

(vii) pour tout x, l'intervalle $G^-(x) \nabla G^+(x)$ est disjoint du système $\Phi_N(x)$, c.à d. est disjoint de chaque segment de ce système.

On dit que l'intervalle $\alpha_1 \nabla \beta_1$ est disjoint du segment $\alpha_2 \Delta \beta_2$ si on a $\alpha_2 \gg \beta_1$ ou $\alpha_1 \gg \beta_2$.

Preuve de la proposition 1.

1ère étape. Définition des $\mathcal{J}$-cortèges.

A tout cortège P de zéros et de uns (on notera $1 \leftarrows 0|$) on associe un segment, un segment $\alpha_o \Delta \beta_o$ initial étant donné. On peut construire un algorithme ϖ applicable à tout mot de la forme $\alpha \square \beta \square P$ où α et β sont des réels tels que $\alpha < \beta$ et P un cortège de zéros et de uns (on dira dorénavant un cortège), tel que l'on ait :

$$\varpi(\alpha \square \beta \square 0) \underset{\cdot}{=} \alpha \Delta \alpha + \frac{\beta-\alpha}{3}$$

$$\varpi(\alpha \square \beta \square 1) \underset{\cdot}{=} \alpha + 2.\frac{\beta-\alpha}{3} \Delta \beta$$

si $$\alpha_p \Delta \beta_p \leftarrows \varpi(\alpha \square \beta \square P)$$

alors $$\varpi(\alpha \square \beta \square P * 0) \underset{\cdot}{=} \alpha_p \Delta \alpha_p + \frac{\beta_p - \alpha_p}{3}$$

$$\varpi(\alpha \square \beta \square P * 1) \underset{\cdot}{=} \alpha_p \quad 2.\frac{\beta_p - \alpha_p}{3} \Delta \beta_p .$$

En outre, on désigne par $\mathcal{Q}$ et $\mathcal{C}omp$ les algorithmes de schèmes respectifs :

$$\mathcal{Q} \quad \begin{cases} \delta * \longrightarrow |\delta \\ \delta \longrightarrow \cdot \\ \xi \longrightarrow \xi \in N_4 \\ \quad \longrightarrow 0|\delta \end{cases}$$

$$\mathcal{C}omp \quad \begin{cases} \square\square\xi \longrightarrow \xi\square\square\square & (\in N_4) \\ |\xi\square \longrightarrow | \\ |\square\xi \longrightarrow \xi\square\square \\ |\square\square* \longrightarrow |\square \\ 0\xi\square \longrightarrow \xi 0 \\ 0\square\square*\eta \longrightarrow 0\square\square* & (\eta \in N_4 \cup \{*\}) \\ 0\square\square* \longrightarrow \cdot \end{cases}$$

Il est alors clair que pour tout cortège P, $\mathcal{Q}(P)$ donne le nombre de zéros et de uns de P, qu'on appellera longueur de P, et que $\mathcal{C}omp$ est applicable à tout mot $n \square P$ où n est un entier naturel et P un cortège avec : $\mathcal{C}omp(n \square P)$ est la $n^{\text{ième}}$ composante

du cortège si $1 \leqslant n \leqslant \mathbb{Q}(P)$. On peut remarquer que $\mathbb{C}\mathrm{omp}$ a cette propriété pour des cortèges de mots dans N_4 définis par la lettre $*$.

On considère alors $\mathbb{J}$ un algorithme du type $(H \longrightarrow H)$ tel que :

(i) pour chaque n, si $! \mathbb{J}(n)$, alors $\mathbb{J}(n) \doteq 0$ ou $\mathbb{J}(n) \doteq 1$

(ii) il est impossible de construire un algorithme $\mathbb{K}$ du type $(H \longrightarrow H)$ vérifiant la propriété :

$$\forall n \quad (n > 0 \;\&\; ! \mathbb{J}(n) \supset \mathbb{K}(n) \doteq \mathbb{J}(n)).$$

Au cours de la démonstration du lemme fondamental de § 3 on a indiqué un moyen de construire un tel algorithme.

Nous désignerons alors par $\mathbb{B}$ l'algorithme de développement canonique de $\mathbb{J}$.

On appellera alors $\mathbb{J}$-cortège, tout cortège P vérifiant la propriété :

$$(*) \quad \forall n \quad (1 \leqslant n \leqslant \mathbb{Q}(P) \;\&\; \mathbb{B}(n \square \mathbb{Q}(P)) \doteq \Lambda \supset \mathbb{C}\mathrm{omp}(n \square P) \doteq \mathbb{J}(n)).$$

Remarquons que la condition $\mathbb{B}(\Lambda \square \mathbb{Q}(P)) \doteq \Lambda$ implique $! \mathbb{J}(n)$.

2ème étape. Propriétés des J-cortèges.

Les propriétés des $\mathbb{J}$-cortèges (voir [17]) sont les suivants :

(a) on peut construire un algorithme $\mathbb{R}$ applicable à tous les cortèges P et tel que pour chaque P, $\mathbb{R}(P) \doteq \Lambda$ si et seulement si P est un $\mathbb{J}$-cortège ;

(b) pour tout entier n positif, on peut construire un $\mathbb{J}$-cortège de longueur n ;

(c) soient P et Q deux cortèges tels que Q soit un début de P (comme mots dans N_5) et P soit un $\mathbb{J}$-cortège, alors Q est un $\mathbb{J}$-cortège ;

(d) il est impossible de construire un algorithme $\mathbb{A}$ du type $(H \longrightarrow H)$ tel que pour chaque n le mot $\mathbb{A}(1) * \ldots * \mathbb{A}(n)$ soit un $\mathbb{J}$-cortège de longueur n.

(e) pour chaque n positif, un $\mathcal{J}$-cortège de longueur n P est quasi-réalisable (c. à d. on ne peut pas construire P) tel qu'on puisse construire un nombre aussi grand qu'on veut de $\mathcal{J}$-cortèges dont P est un début.

La propriété (a) résulte de ce que (*) est algorithmiquement vérifiable (on utilise le théorème de ramification des algorithmes pour tester $1 \leqslant n \leqslant \mathfrak{L}(P)$, $\mathcal{B}(n \square \mathfrak{L}(P)) \doteq \Lambda$ ou non et ce qu'on doit faire dans chaque cas).

La propriété (c) résulte des définitions et des propriétés de $\mathcal{B}$: on a par hypothèse $\mathfrak{L}(Q) \leqslant \mathfrak{L}(P)$. Alors :

(**) $\quad \forall n \quad (1 \leqslant n \leqslant \mathfrak{L}(Q) \supset \mathcal{C}\mathrm{omp}(n \square Q) \doteq \mathrm{Comp}(n \square P))$

De plus, si $\mathcal{B}(n \square \mathfrak{L}(Q)) \doteq \Lambda$ alors $\mathfrak{L}(P) \geqslant \mathfrak{L}(Q)$ fait que $\mathcal{B}(n \square \mathfrak{L}(P)) \doteq \Lambda$ et comme $\mathcal{C}\mathrm{omp}(n \square P) \doteq \mathcal{J}(n)$ par (**) on a $\mathcal{C}\mathrm{omp}(n \square Q) \doteq \mathcal{J}(n)$. D'où la propriété.

Considérons l'algorithme $\mathcal{A}$ ainsi défini :

$$\mathcal{A}(n \square m) \doteq \mathcal{J}(n) \qquad \text{si} \qquad \mathcal{B}(n \square m) \doteq \Lambda$$

$$\mathcal{A}(n \square m) \doteq 0 \qquad \text{si} \qquad \mathcal{B}(n \square m) \neq \Lambda .$$

Soit alors $\qquad P \leftrightarrows \mathcal{A}(1 \square n) * \mathcal{A}(2 \square n) * \ldots * \mathcal{A}(n \square n).$

Alors si $1 \leqslant k \leqslant n$ et $\mathcal{B}(k \square n) \doteq \Lambda$ on a donc $\mathcal{C}\mathrm{omp}(k \square P) \doteq \mathcal{A}(k \square n)$ et alors $\mathcal{A}(k \square n) \doteq \mathcal{J}(k)$. Donc P est un $\mathcal{J}$-cortège de longueur n et comme $\mathcal{A}$ peut être construit (ramification des algorithmes), la propriété (b) a lieu.

Supposons que l'on puisse construire un algorithme $\mathcal{A}$ du type $(H \longrightarrow H)$ tel que pour chaque n le mot $\mathcal{A}(1) * \ldots * \mathcal{A}(n)$ soit un $\mathcal{J}$-cortège de longueur n. Soit k tel que $!\mathcal{J}(k)$. Alors il existe un entier naturel m tel que $\mathcal{B}(k \square m) \doteq \Lambda$. Considérons alors $\mathcal{A}(1) * \ldots * \mathcal{A}(m_o)$ où $m_o = \max(k,m)$ c'est un $\mathcal{J}$-cortège et donc $1 \leqslant k \leqslant m_o$

entraine $\mathbb{C}\mathrm{omp}(k \sqsubset P) = \mathbb{J}(k)$ c. à d. $\mathbb{A}(k) \doteq \mathbb{J}(k)$. Or un tel algorithme est impossible par construction de $\mathbb{J}$. D'où la propriété (d).

Montrons (e) : considérons dans ce point particulier la lettre b comme lettre générique des cortèges. Comme pour chaque entier naturel m il y a au plus 2^{m+1} $\mathbb{J}$-cortèges de longueur m+1, en vertu de la propriété (c), l'assertion "on peut construire un aussi grand nombre qu'on veut de $\mathbb{J}$-cortèges dont P est un début" équivaut à "pour chaque entier naturel m, on peut construire un $\mathbb{J}$-cortège Q de longueur m+1, dont P est un début". Ainsi (e) peut s'écrire :

$$\forall n \ \neg\neg \exists\, b_1 \ ((\mathbb{R}(b_1) \doteq \Lambda) \& (\mathbb{P}(b_1) \doteq n+1) \& \forall m \exists\, b_2 ((\mathbb{R}(b_1 * b_2) \doteq \Lambda) \& (\mathbb{L}(b_2) \doteq m+1))).$$

Notons $I(b_1) \leftrightarrows \forall m \exists\, b_2 ((\mathbb{R}(b_1 * b_2) \doteq \Lambda) \& (\mathbb{P}(b_2) \doteq m+1)$. Alors on veut établir

$$\forall n \neg\neg \exists\, b_1 ((\mathbb{R}(b_1) \doteq \Lambda) \& (\mathbb{L}(b_1) \doteq n+1) \& I(b_1)).$$

Supposons que l'on ait :

$$(\overset{**}{*}) \qquad \exists\, n \neg \exists\, b_1 ((\mathbb{R}(b_1) \doteq \Lambda) \& (\mathbb{L}(b_1) \doteq n+1) \& I(b_1)$$

on a donc par (2.3.3) et (2.3.7)

$$(\overset{**}{**}) \qquad \forall b_1 (((\mathbb{R}(b_1) \doteq \Lambda) \& (\mathbb{P}(b_1) \doteq n+1)) \supset \neg I(b_1)$$

b_1 étant fixé, supposons que :

$$(\square) \qquad \neg \exists\, m \forall b_2 ((\mathbb{L}(b_2) \doteq m+1) \supset \neg (\mathbb{R}(b_1 * b_2) \doteq \Lambda)).$$

Alors par (2.3.3) puis (2.3.7) on a successivement

$$\forall m \neg \forall b_2 \ (\mathbb{L}(b_2) \doteq m+1 \supset \neg \mathbb{R}(b_1 * b_2) \doteq \Lambda))$$

$$\forall m \neg \forall b_2 \ \neg ((\mathbb{L}(b_2) \doteq m+1) \& (\mathbb{R}(b_1 * b_2) \doteq \Lambda)).$$

Donc : $\forall m \ \neg\neg \exists\, b_2 \ (\mathbb{L}(b_2) \doteq m+1) \& \mathbb{R}(b_1 * b_2) \doteq \Lambda)$.

On pose $A(b_1,b_2,m) \leftrightarrows ((\mathfrak{L}(b_2) \doteq m+1) \& \mathfrak{R}(b_1 * b_2) \doteq \Lambda)$

il est clair, puisque $A(b_1,b_2,m)$ est défini par l'intersection de deux algorithmes, que $A(b_1,b_2,m)$ est algorithmiquement vérifiable et qu'on a :

$$\forall b_1 \ (A(b_1,b_2,m) \vee \neg A(b_1,b_2,m)).$$

Donc, par le principe de Markov :

$\forall m \exists b_2 \ A(b_1,b_2,m)$ c. à d. $\forall m \exists b_2 \ ((\mathfrak{L}(b_2) \doteq M+1) \& (\mathfrak{R}(b_1 * b_2) \doteq \Lambda))$.

Donc : $I(b_1)$ découle de $(\square)$.

Donc $(\square)$ contredit $(\overset{**}{**})$ et on a donc :

$(\square\square)$ $\forall b_1(((\mathfrak{R}(b_1) \doteq \Lambda) \& (\mathfrak{L}(b_1) \doteq n+1)) \supset \neg\neg\exists m \ \forall b_2((\mathfrak{L}(b_2) \doteq m+1) \supset \neg(\mathfrak{R}(b_1 * b_2) \doteq \Lambda)))$.

Posons $C(b_1,b_2,m) \leftrightarrows ((\mathfrak{L}(b_2) \doteq m+1) \supset \neg(\mathfrak{R}(b_1 * b_2) \doteq \Lambda))$

la formule $(\square\square)$ est de la forme :

$$\forall b_1 \ ((\mathfrak{R}(b_1) \doteq \Lambda) \ \& \ (\mathfrak{L}(b_1) \doteq n+1)) \supset \neg\neg\exists \ m \ \forall b_2 \ C \ (b_1,b_2,m).$$

Or on a :

(0) $$\forall m \ (\forall b_2 \ C \ (b_1,b_2,m) \vee \neg \forall b_2 \ C \ (b_1,b_2,m))$$

en effet :

$$\neg\forall b_2 \ C \ (b_1,b_2,m) \equiv \neg\forall b_2 \neg((\mathfrak{L}(b_2) \doteq m+1) \& (\mathfrak{R}(b_1 * b_2) \doteq \Lambda))$$

par (2.3.7) donc par (2.3.3) :

$$\equiv \neg\neg\exists \ b_1 \ ((\mathfrak{L}(b_2) \doteq m+1) \& (\mathfrak{R}(b_1 * b_2) \doteq \Lambda))$$

et donc, par ce qu'on a vu plus haut $(\forall b_1(A(b_1,b_2,m) \vee \neg A(b_1,b_2,m)))$ on a donc

$$\neg\forall b_2 \ C \ (b_1,b_2,m) \equiv \exists \ b_1((\mathfrak{L}(b_2) \doteq m+1) \& (\mathfrak{R}(b_1 * b_2) \doteq \Lambda)).$$

Donc (0) équivaut à :

(00) $\forall m \ (\forall b_2(\mathfrak{L}(b_2) \doteq m+1 \supset \neg(\mathfrak{R}(b_1 * b_2) \doteq \Lambda)) \vee \exists \ b_2((\mathfrak{L}(b_2) \doteq m+1) \& (\mathfrak{R}(b_1 * b_2) \doteq \Lambda)))$.

Or ceci est algorithmiquement vérifiable en étudiant un à un tous les b_2 pour lesquels $\ell(b_2) \doteq m+1$, pour chaque m.

Donc, par une nouvelle application du principe de Markov, on déduit de ($\square\square$) la formule :

($\square\!\!\square$) $\forall b_1(((\mathbb{R}(b_1) \doteq \Lambda) \& (\ell(b_1) \doteq n+1)) \supset \exists m \forall b_2((\ell(b_2) \doteq m+1) \supset \neg(\mathbb{R}(b_1 * b_2) \doteq \Lambda)))$.

En revenant à l'interprétation constructive des assertions, et en considérant tous les b_1 pour lesquels $\ell(b_1) \doteq n+1$ et le maximum des m associés par la formule ($\square\!\!\square$), on trouve qu'à partir d'un certain rang, il n'y a plus de $\mathbb{J}$-cortège (en utilisant à nouveau (c)) ce qui contredit (b).

Donc on a :

$$\neg\exists n \neg\exists b_1 \quad (\mathbb{R}(b_1) \doteq \Lambda) \& (\ell(b_1) \doteq n+1) \& I(b_1))$$

d'où la propriété (e).

3ème étape. Construction de Φ, de $\mathbb{N}$, de $\mathbb{G}^-$ et de $\mathbb{G}^+$.

On construit Φ de la façon suivante :

$\Phi_0 = 0 \Delta 1$ et pour tout entier naturel $n > 0$, Φ_n désigne la réunion des segments de la forme $\sigma(0\square 1\square P)$ où P est un $\mathbb{J}$-cortège de longueur n. Par définition de la réunion de segments rationnels, Φ_n est un système de segments rationnels. La construction de Φ est possible puisqu'on dispose d'un algorithme $\mathbb{R}$ reconnaissant les $\mathbb{J}$-cortèges et puisqu'il y a un nombre fini de cortèges de longueur n. En utilisant l'ordre lexicographique pour les mots dans $\{0,1\}$ (on a une bijection constructive évidente entre cortèges et mots non vides de $\{0,1\}$), on construit (voir 2.5.1 démonstration de la proposition 2) un algorithme réalisant une bijection $\mathbb{B}$ entre cortèges et entiers naturels. Les cortèges de longueur n seront donc numérotés entre $m(n)$ et $m(n+1) - 1$. A partir de $\mathbb{R}$ on construit un algorithme $\mathbb{Q}$ tel que :

$$\mathbb{Q}(k) \underset{\circ}{=} \mathbb{C}(0 \square 1 \square \mathbb{b}(k)) \qquad \text{si} \qquad \mathbb{R}(\mathbb{b}(k)) \underset{\circ}{=} \Lambda$$

$$\mathbb{Q}(k) \underset{\circ}{=} \Lambda \qquad \text{si} \qquad \mathbb{R}(\mathbb{b}(k)) \underset{\circ}{\neq} \Lambda \ .$$

Avec le théorème de réunion des algorithmes, si $\mathbb{T}$ est un algorithme "régularisant" les occurences de la lettre $*$ (c. à d. on ne rencontrera pas $** A ** \ldots * B * \ldots$ où A et B sont des mots dans N_4) on peut représenter Φ par :

$$\Phi(n) \simeq \pi\, (\mathbb{Q}(m(n)) * \mathbb{Q}(m(n)+1) * \ldots * \mathbb{Q}(m(n+1)-1).$$

On construit alors $\mathbb{p}$ un algorithme <u>applicable</u> aux mots de la forme $x \square m \square P$ où x est réel, m entier naturel et P cortège et tel que :

$$\mathbb{p}(x \square m \square P) \underset{\circ}{\neq} \Lambda \qquad \text{si et seulement si}$$

$$\mathbb{R}(P) \underset{\circ}{=} \Lambda \ \&\ (\mathbb{D}^-(x \square m) \in \mathbb{C}(0 \square 1 \square P) \vee \mathbb{D}^+(x \square m) \in \mathbb{C}(0 \square 1 \square P)).$$

Cela est possible puisque $\mathbb{C}(0 \square 1 \square P)$ étant un segment <u>rationnel</u> la condition que l'on vient d'écrire est algorithmiquement vérifiable (on sait que $\mathbb{D}^-(x \square m)$ et $\mathbb{D}^+(x \square m)$ sont des rationnels).

On pose alors :

$$\mathbb{N}(x) \simeq 1 + \mu n \Big(\bigvee_{\mathbb{Q}(P)=n} \mathbb{p}(x \square 2n \square P) \underset{\circ}{=} \Lambda \Big)$$

où $\bigvee_{\mathbb{Q}(P)=n} \mathbb{p}(x \square 2n \square P) \leftrightarrows \mathbb{p}(x \square 2n \square P_1)\mathbb{p}(x \square 2n \square P_2) \ldots \mathbb{p}(x \square 2n \square P_{2^n})$

où $P_1, \ldots, P_{2^n}$ désignent, dans l'ordre lexicographique les 2^n cortèges de longueur n.

On construit $\mathbb{G}^-$ et $\mathbb{G}^+$ à l'aide de $\mathbb{N}$:

$$\mathbb{G}^-(x) \simeq \mathbb{D}^-(x \square 2\mathbb{N}(x))$$

$$\mathbb{G}^+(x) \simeq \mathbb{D}^+(x \square 2\mathbb{N}(x)).$$

<u>4ème étape. Propriétés de $\mathbb{N}$ et fin de la démonstration.</u>

Le point essentiel est de montrer que :

$$\forall x \ ! \ \mathbb{N}(x).$$

Supposons qu'il existe un x tel que $\mathbb{N}$ ne soit pas applicable à x. C'est-à-dire $\exists\, x \neg !\, \mathbb{N}(x)$.

Cela signifie que pour chaque n on peut trouver un cortège P_n de longueur n tel que $\mathbb{p}(x \square 2n \square P) \neq_{\circ} \Lambda$ (en effet si $!\,\mathbb{N}(x)$ on ne peut avoir $\exists\, n\ \pi\ \ldots(\mathbb{p}(x \square 2n \square \pi) =_{\circ} \Lambda)$ où π désigne les cortèges de longueur n, donc on a $\forall n \neg \forall \pi\ (\mathbb{p}(x \square 2n \square \pi) =_{\circ} \Lambda)$ et comme π parcourt un ensemble fini on a $\forall n \rceil \pi\ (\mathbb{p}(x \square 2n \square \pi) \neq_{\circ} \Lambda).(*))$. Donc P_n est un $\mathbb{J}$-cortège de longueur n et on a

$$\mathbb{D}^{-}(x \square 2n) \in \varpi(0 \square 1 \square P_n) \vee \mathbb{D}^{+}(x \square 2n) \in \varpi(0 \square 1 \square P_n).$$

Or $\mathbb{D}^{+}(x \square 2n) - \mathbb{D}^{-}(x \square 2n) < 4^{-n}$.

Or : $\mathbb{D}^{-}(x \square 2(n+1)) \in \varpi(0 \square 1 \square P_{n+1}) \vee \mathbb{D}^{+}(x \square 2(n+1)) \in \varpi(0 \square 1 \square P_{n+1})$.

La distance entre $\mathbb{D}^{+}(x \square 2(n+1))$ et $\mathbb{D}^{-}(x \square 2n)$ est inférieure à $2.4^{-n-1} < 4^{-n}$. Or si $\varpi(0 \square 1 \square P_{n+1}) \not\subset \varpi(0 \square 1 \square P_n)$, par construction des ϖ, $\varpi(0 \square 1 \square P_{n+1}) \subset \varpi(0 \square 1 \square (P_{n+1})_n)$ où $(P_{n+1})_n$ est le début de longueur n de P_{n+1} et donc $(P_{n+1})_n \neq P_n$

$\varpi(0 \square 1 \square P_{n+1}) \not\subset \varpi(0 \square 1 \square P_n)$ est le début de longueur n de P_{n+1} et donc $(P_{n+1})_n \neq P_n$

et donc la distance entre deux rationnels quelconques de ces segments rationnels est au moins égale à $3^{-n} > 4^{-n}$.

On déduit donc de ceci que $\varpi(0 \square 1 \square P_{n+1}) \subset \varpi(0 \square 1 \square P_n)$ et donc :

$$\forall n\ \ ((P_{n+1})_n \neq P_n).$$

Soit θ l'algorithme défini par $\theta(0) =_{\circ} 0$ et :

$$\forall n\ \ \theta(n) = \mathbb{C}\mathrm{omp}(n \square P_n).$$

P_n est obtenu algorithmiquement donc θ est du type $(H \longrightarrow H)$ et de plus il est clair, par construction que

$$\forall n \qquad \theta(1) * \ldots * \theta(n)$$

est un $\mathbb{J}$-cortège. Or ceci est impossible.

(*) $\mathbb{p}$ est applicable à tous les mots de la forme $x \square 2n \square P$ et donc

$$\neg(\mathbb{p}(x \square 2n \square \pi) = \Lambda) \equiv \mathbb{p}(x \square 2n \square \pi) \neq \Lambda.$$

Donc : $\neg\exists x \neg! \mathbb{N}(x)$ d'où (§ 2.3.(3) et principe de Markov)

$$\forall x\, !\, \mathbb{N}(x).$$

Supposons $\mathbb{G}^-(x) \vee \mathbb{G}^+(x) \cap \Phi_{N(x)}$ non vide. Cela signifie donc (on a des continus rationnels, on peut donc raisonner sur les ensembles de rationnels associés) que pour un $\mathbb{J}$-cortège P de longueur $\mathbb{N}(x)$ on a $\mathbb{G}^-(x) \vee \mathbb{G}^+(x) \cap \varpi(0 \square 1 \square P)$ vide et donc (définition de $\mathbb{G}^-$ et $\mathbb{G}^+$) si $a \Delta b \leftarrow \varpi(0 \square 1 \square P)$ on a

$$\mathbb{G}^-(x) < b \qquad \text{et} \qquad \mathbb{G}^+(x) > a.$$

Or $$\mathbb{G}^+(x) - \mathbb{G}^-(x) < 4^{-\mathbb{N}(x)} < 3^{-\mathbb{N}(x)}.$$

Donc $\mathbb{G}^+(x) > a$ fait $\mathbb{G}^-(x) > \mathbb{G}^+(x) - 3^{-\mathbb{N}(x)}$.

Si $\mathbb{G}^+(x) > b$ on a donc $\mathbb{G}^-(x) > a$ d'où $\mathbb{G}^-(x) \in a \Delta b$ si $\mathbb{G}^-(x) < a$ on voit de même que $\mathbb{G}^+(x) \in a \Delta b$.

Donc en fait on a :

$$\mathbb{G}^-(x) \in \varpi(0 \square 1 \square P) \vee \mathbb{G}^-(x) \in \varpi(0 \square 1 \square P)$$

c'est-à-dire $\mathrm{p}(x \square 2\mathbb{N}(x) \square P) \neq \Lambda$. D'où, par définition de $\mathbb{N}$ on a $\mathbb{N}(x) > \mathbb{N}(x)$ ce qui est absurde.

Donc $\mathbb{G}^-(x) \vee \mathbb{G}^+(x)$ est disjoint de $\Phi_{\mathbb{N}}(x)$.

Il reste à établir $\Phi_{n+1} \subset \Phi_n$ pour chaque n, mais cela découle de la propriété (c) des $\mathbb{J}$-cortèges et de la construction de $\varpi \circ \square$.

Théorème 2. On peut construire un algorithme Φ transformant tout entier naturel n en un système de segments rationnels Φ_n tel que :

(i) $\Phi_0 = 0 \Delta 1$

(ii) $\forall n\, (\Phi_{n+1} \subset \Phi_n)$

(ii) $\cdot \neg\exists x \forall n\ (x \in \Phi_n)$.

Preuve. Avec (vi) et (vii) de la proposition 1 on a évidemment (iii).

Théorème 3. On peut construire un algorithme $\mathbb{H}$, transformant tout entier naturel en un système d'intervalles tels que :

pour tout réel $x \in 0\Delta 1$, on peut construire un n tel que $x \in \mathbb{H}_n$ et tel que pour tout entier naturel m, aucun x de $0\Delta 1$ ne se trouve dans au moins un des systèmes $\mathbb{H}_0, \ldots, \mathbb{H}_n$.

Preuve. On peut construire un algorithme $\mathbb{H}$ tel que : si $\mathbb{Q}$ est un système de segments rationnels dans $0 \Delta 1$, alors $\mathbb{H}(\mathbb{Q})$ est un système d'intervalles rationnels tels que :

$$x \in \mathbb{H}(\mathbb{Q}) \equiv x \in -\frac{1}{2} \nabla \frac{3}{2} \,\&\, x \notin \mathbb{Q} .$$

On prend en quelque sorte le complémentaire dans $\mathfrak{M}_p(-\frac{1}{2} \nabla \frac{3}{2})$ de $\mathfrak{M}_p(\mathbb{Q})$ qui peut se traduire en un système rationnel $\mathbb{H}(\mathbb{Q})$. La propriété $x \mathbb{H}(\mathbb{Q}) \supset x \in -\frac{1}{2} \nabla \frac{3}{2} \,\&\, x \notin \mathbb{Q}$ résulte du théorème 1 de 1.3 de ce paragraphe : $\mathbb{H}(\mathbb{Q})$ définit une partition de $-\frac{1}{2} \nabla \frac{3}{2}$ et le théorème cité permet de dire à quel intervalle de $\mathbb{H}(\mathbb{Q})$ x appartient (sous l'hypothèse $x \in \mathbb{H}(\mathbb{Q})$). $x \in \frac{1}{2} \nabla \frac{3}{2} \,\&\, x \notin \mathbb{Q} \supset x \in \mathbb{H}(\mathbb{Q})$ résulte de la définition.

On pose : $$\mathbb{H}(n) \underset{\circ}{=} \mathbb{H}(\Phi_n).$$

Il est clair que :

(i) $\forall x \ (x \in 0 \Delta 1 \supset x \in \mathbb{H}(\mathbb{N}(x)))$ (c'est le point (ii) de la proposition 1)

(ii) $\mathbb{H}(0) \subset \mathbb{H}(1) \subset \ldots \subset \mathbb{H}(n)$ pour chaque n

(iii) $\forall n \ (\mathrm{Ext}_1(\Phi_{n+1}) \in 0 \Delta 1 \,\&\, \mathrm{Ext}_1(\Phi_{n+1}) \notin \mathbb{H}(n))$ car $\Phi_{n+1} \subset \Phi_n$. □.

3. On va donner deux corollaires intéressants de la proposition 1. Pour ce faire, il nous faut modifier légèrement la construction de Φ. Au lieu de la fonder sur

l'algorithme ϖ, nous la fonderons sur l'algorithme ϖ' ainsi défini : pour tout mot $\alpha \square \beta \square P$ où $\alpha < \beta$ et P est un cortège ϖ' est applicable à un tel mot ;

$$\alpha < \beta \qquad \varpi'(\alpha \square \beta \square 0) \underset{\cdot}{=} \alpha + \frac{\beta-\alpha}{5} \Delta\, \alpha + 2.\frac{\beta-\alpha}{5}$$

$$\varpi'(\alpha \square \beta \square 1) \underset{\cdot}{=} \alpha + 3.\frac{\beta-\alpha}{5} \Delta\, \alpha + 4.\frac{\beta-\alpha}{5}$$

et si $\alpha_p \Delta \beta_p \leftarrows \varpi'(\alpha \square \beta \square P)$, alors :

$$\varpi'(\alpha \square \beta \square P * 0) \underset{\cdot}{=} \alpha_p + \frac{\beta_p-\alpha_p}{5} \Delta\, \alpha_p + 2.\frac{\beta_p-\alpha_p}{5}$$

$$\varpi'(\alpha \square \beta \square P * 1) \underset{\cdot}{=} \alpha_p + 3.\frac{\beta_p-\alpha_p}{5} \Delta\, \alpha_p + 4.\frac{\beta_p-\alpha_p}{5}.$$

Le reste de la démonstration de la proposition 1 se conserve littéralement. On a :

Théorème 1 (cf. [17]). On peut construire un algorithme λ du type $(H \rightarrow p)$ tel que l'on ait :

(i) $\forall n \quad (\lambda(n) < \lambda(n+1))$

(ii) $\forall n \quad (0 \leqslant \lambda(n) \leqslant 1)$

(iii) $\forall x \quad (\forall n(\lambda(n) \leqslant x) \supset \forall n \ \lambda(n) \leqslant \mathfrak{G}^-(x))$.

Preuve. On prend $\lambda(n) \underset{\cdot}{=} \mathbb{E}xt_1(\Phi'(n))$ (c. à d. construit à partir de ϖ'). Alors (i) est évident (on ne l'avait pas nécessairement avec ϖ) ainsi que (ii).

Pour (iii) on remarque que $\mathfrak{G}^-(x) \nabla \mathfrak{G}^+(x)$ ne rencontre pas $\Phi' N(x)$. Comme $\Phi'_{n+1} \subset \Phi'_n$, a fortiori $\mathfrak{G}^-(x) \nabla \mathfrak{G}^+(x)$ ne rencontre pas Φ'_n pour $n \geqslant \mathbb{N}(x)$. Comme $\lambda(n) \leqslant x$, on ne peut avoir $\lambda(n) \geqslant \mathfrak{G}^+(x)$ et donc nécessairement $\lambda(n) \leqslant \mathfrak{G}^-(x)$ pour $n \geqslant \mathbb{N}(x)$, donc, les $\lambda(n)$ étant croissants (par rapport à n) on a donc $\forall n \ (\lambda(n) \leqslant \mathfrak{G}^-(x))$. $\square$.

Ce théorème constitue en quelque sorte une amélioration du théorème de Specker.

Nous concluerons sur le corollaire important suivant :

Théorème 2 (I. D. Zaslavsky [17]). On peut construire une fonction constructive sur $0 \Delta 1$ continue et non bornée.

Preuve. Considérons Φ' défini précédemment.

Soit $\mathbb{F}$ la fonction ainsi définie, $a < b$ étant fixés :

$$\mathbb{F}(a \square b \square x) = \frac{5|x-a| - |5x-4a-b| - |5x-a-4b| + 5|x-b|}{2(b-a)}.$$

On a (voir fig. 1) :

$$\mathbb{F}(a \square b \square a) = \mathbb{F}(a \square b \square b) = 0$$

$$\mathbb{F}(a \square b \square a+\tfrac{1}{5}(b-a)) = \mathbb{F}(a \square b \square a+\tfrac{4}{5}(b-a)) = 1$$

et $\mathbb{F}$ est linéaire sur chacun des segments

$a \Delta a + \frac{1}{5}(b-a)$, $a+\frac{1}{5}(b-a) \Delta \frac{4}{5}(b-a)$,

$a+\frac{4}{5}(b-a) \Delta b$, et nulle hors de $a \Delta b$.

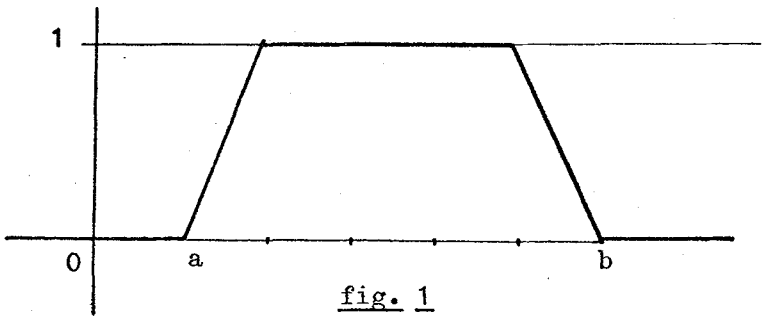

fig. 1

Si $\mathbb{Q}$ est un système de segments rationnels non dégénérés, on pose :

$$\mathbb{F}(\mathbb{Q} \square x) = \sum_{k=1}^{\mathbb{L}(\mathbb{Q})} \mathbb{F}(\mathbb{E}xt_1(\mathbb{C}omp(k \square \mathbb{Q})) \square \mathbb{E}xt_2(\mathbb{C}omp(k \square \mathbb{Q})) \square x).$$

(voir fig. 2)

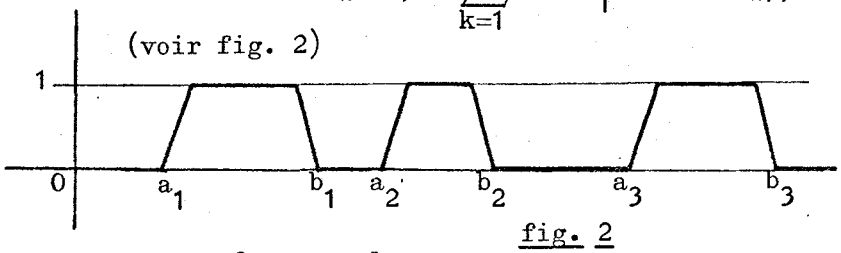

fig. 2

Il est clair que $\mathbb{F}(\mathbb{Q} \square x)$ est une fonction constructive de x.

On pose alors

$$\mathbb{f}(x) = \sum_{k=0}^{\mathbb{N}(x)} \mathbb{F}(\Phi'_k \square x).$$

Comme $\mathbb{N}$ est simplement un algorithme, $\mathbb{f}$ n'est pas nécessairement une fonction.

Il faut montrer

$$(x = y \supset \mathbb{f}(x) = \mathbb{f}(y)).$$

(Il est clair que $\mathbb{f}$ est un algorithme applicable à tout réel).

Soit $x = y$.

Alors si $\mathbb{N}(x) = \mathbb{N}(y)$ $\quad \mathbb{f}(x) = \mathbb{f}(y)$.

Si $\mathbb{N}(x) \neq \mathbb{N}(y)$ on peut toujours supposer $\mathbb{N}(x) < \mathbb{N}(y)$.

Alors :

$$\mathbb{f}(y) = \sum_{k=0}^{\mathbb{N}(x)} \mathbb{F}(\Phi'_k \square y) + \sum_{k=\mathbb{N}(x)+1}^{\mathbb{N}(y)} \mathbb{F}(\Phi'_k \square y)$$

$$= \mathbb{f}(x) + \sum_{k=\mathbb{N}(x)+1}^{\mathbb{N}(y)} \mathbb{F}(\Phi'_k \square y).$$

Or : $x \notin \Phi'_k$ pour $k \geqslant \mathbb{N}(x)$ et comme $x = y$ on a de même $y \notin \Phi'_k$ et donc $\mathbb{F}(\Phi'_k \square y) = 0$ pour $k = \mathbb{N}(x)+1, \ldots, \mathbb{N}(y)$. Donc $\mathbb{f}(y) = \mathbb{f}(x)$.

Pour la même raison on voit que $\mathbb{G}^-(x) < y < \mathbb{G}^+(x) \supset \mathbb{f}(y) = \sum_{k=0}^{\mathbb{N}(x)} \mathbb{F}(\Phi'_k \square y)$ c'est-à-dire est une somme finie (ne dépendant pas de y) de fonctions continues(*). Donc $\mathbb{f}(y)$ est continue dans $\mathbb{G}^-(x) \nabla \mathbb{G}^+(x)$ et donc en chaque point x. Posons $a_n \leftrightarrows \mathbb{E}xt_1(\Phi_n)$

$$\mathbb{f}(a_n) = \sum_{0}^{\mathbb{N}(a_n)} \mathbb{F}(\Phi'_k \square a_n).$$

Il est clair que si $k > n$ $a_n \notin \Phi'_k$ (la construction de ϖ' intervient pour ce point) et donc $\mathbb{f}(a_n) = \sum_{k=0}^{n} \mathbb{F}(\Phi'_k \square a_n)$.

Or : $\mathbb{F}(\Phi'_n \square a_n) = 0$ car $a_n = \mathbb{E}xt_1(\Phi_n)$ (voir la fig. 2).

Si $k \leqslant n-1$ on a cependant $\mathbb{F}(\Phi'_k \square a_n) = 1$ car si P_n est le cortège de longueur n tel que $a_n \in \varpi'(P_n)$ on a $a_n \in \varpi'((P_n)_{n-1})$ et si $c_n = \mathbb{E}xt_1\, \varpi'((P_n)_{n-1})$ on a donc $a_n = c_n + \frac{1}{5^n}$ (voir fig. 1) $F(\Phi'_k \square a_n) = 1$.

Donc on a $\mathbb{f}(a_n) = n$. $\square$.

(*) On dit qu'une fonction f du type $(A \xrightarrow{\cdot} A)$ est continue si :

$$\forall x\, (!\, \mathbb{f}(x) \supset \forall k\, \exists \ell\, (|y-x| < 2^{-\ell} \,\&\, !\, \mathbb{f}(y) \supset |\mathbb{f}(x) - \mathbb{f}(y)| < 2^{-k}\,).$$

Il est facile de voir qu'une somme finie de fonctions continues l'est aussi et que les fonctions $\mathfrak{g}_1(x) = |x|$ et $\mathfrak{g}_2(x) = |x-a|$ où a est un rationnel donné, sont des fonctions continues.

Appendice 1

Sur les notions constructives de nombre réel

Dans cet exposé, nous avons défini les nombres réels par la donnée de deux algorithmes, $\langle P\rangle$ et $\langle Q\rangle$, l'un, par exemple $\langle P\rangle$, de type $(H \text{ — } p)$, l'autre de type $(H \text{ — } H)$ vérifiant la propriété :

$$\forall k\ m\ n\ \ (m,n \geqslant \langle Q\rangle(k) \supset |\langle P\rangle(m) - \langle P\rangle(n)| < 2^{-k}).$$

On peut introduire également d'autres notions(*) :

1. On appelle nombre réel faible (en abrégé réel faible) tout rationnel et tout mot dans N_3 de la forme $P\Diamond$ où P est le code dans $\{0,1\}$ d'un algorithme du type $(H \to P)$ vérifiant la condition :

$$(1) \qquad \forall k\ \exists \ell\ \forall m\ n\ \ (m,n \geqslant \ell \supset |\langle P\rangle(m) - \langle P\rangle(n)| < 2^{-k}).$$

D'après l'interprétation constructive des énoncés, la formule $\forall m\ n\ (m,n \geqslant \ell \supset |\langle P\rangle(m) - \langle P\rangle(n)| < 2^{-k})$ étant normale, (1) se reformule en

$$(1') \qquad \exists f\ \ \forall k\ m\ n\ \ (m,n \geqslant f(k) \supset |\langle P\rangle(m) - \langle P\rangle(n)| < 2^{-k}).$$

2. On appelle nombre quasi-réel (en abrégé quasi réel) tout rationnel et tout mot dans N_3 de la forme $P\Diamond$ où P est le code dans $\{0,1\}$ d'un algorithme du type $(H \to p)$ vérifiant la condition :

$$(2) \qquad \neg\neg\forall k\ \exists \ell\ \forall mn\ (m,n \geqslant \ell \supset |\langle P\rangle(m) - \langle P\rangle(n)| < 2^{-k})$$

ou :

$$(2') \qquad \neg\neg\exists f\ \forall kmn\ (m,n \geqslant f(k) \supset |\langle P\rangle(m) - \langle P\rangle(n)| < 2^{-k}).$$

(*) Dans [1], N.A. Chanine appelle FR-nombre ou duplexe réel, ce que nous appelons nombre réel et F-nombre ce que nous appelons réel faible. Pour les pseudo-réels nous suivons la terminologie de N.A. Chanine. La notion de quasi-réel est définie dans [16]. On remarquera que la notion de nombre quasi-réel est formellement proche de la notion intuitionniste de suite de nombres réels non oscillante (cf. [6]).

3. On appelle pseudo nombre pseudo-réel (en abrégé pseudo-réel) tout rationnel et tout mot dans N_3 de la forme $P\Diamond$ où P est le code dans $\{0,1\}$ d'un algorithme du type $(H \to p)$ vérifiant la condition :

(3) $\quad \forall k \; \neg\neg \exists \ell \; \forall mn \; (m,n \geqslant \ell \supset |\langle P\rangle(m) - \langle P\rangle(n)| < 2^{-k})$.

On désignera respectivement par B , B' et B" les lettres génériques pour représenter les réels faibles, les quasi-réels et les pseudo-réels.

Les relations entre ces nombres peuvent être ainsi formulées :

a. Tout réel faible est un quasi-réel.

b. Tout quasi-réel est un pseudo-réel.

c. Pour tout réel, sa base est un réel faible.

d. On peut construire un pseudo-réel qui n'est identique à la base d'aucun réel.

e. On peut construire un pseudo-réel qui n'est pas un réel faible.

f. On peut construire un pseudo-réel qui n'est pas un quasi-réel.

g. Pour tout réel faible, on peut construire un réel dont la base lui est identique.

h. On ne peut construire un réel faible différent de la base de n'importe quel réel.

i. Il n'existe pas d'algorithme transformant tout réel faible en un régulateur de convergence de ce nombre.

j. Il n'existe pas de quasi-réel qui ne soit pas un réel faible.

k. Il n'est pas vrai que tout quasi réel est un réel faible.

Ces assertions peuvent se formuler ainsi :

(a') $\forall B\ (B \in B')$ où $B \in (B')$ signifie que B est un quasi réel.

(b') $\forall B'(B' \in (B''))$

(c') $\forall \mathfrak{A}\ (\underline{\mathfrak{A}} \in (B))$ où $\mathfrak{A}$ étant de la forme $P \diamond Q$, $\underline{\mathfrak{A}}$ représente $P \diamond$.

(d') $\exists B''\ \forall \mathfrak{A}(\underline{\mathfrak{A}} \neq B'')$

(e') $\exists B''(B'' \notin (B))$ $\quad$ (e'$_1$) $\exists B''\ \forall B\ (B'' \neq B)$

on a aussi

(f') $\exists B''(B'' \notin (B'))$ $\quad$ (f'$_1$) $\exists B''\ \forall B'(B'' \neq B')$

(g') $\forall B\ \exists \mathfrak{A}(\underline{\mathfrak{A}} \doteq B)$

(h') $\neg\exists B\ \forall \mathfrak{A}\ (\underline{\mathfrak{A}} \neq B)$

(i') $\neg\exists \Phi\ \forall B(\Phi(B) \in (\mathfrak{A}) \& \underline{\Phi(B)} \doteq B)$

(j') $\neg\exists B'(B' \notin (B))$

(k') $\neg\forall B'(B' \in (B))$

Les assertions (d'), (b'), (c') sont immédiates.

Pour les autres assertions, nous utiliserons l'expression

f <u>reg. conv.</u> c pour : "l'algorithme représenté par f (ou <f>) est un régulateur de convergence de l'algorithme représenté par <c> .

Notons que : $c \in (B)$ signifie $\exists f(f$ <u>reg-conv</u> $c)$

$c \in (B')$ signifie $\neg\neg\exists f(f$ <u>reg-conv</u> $c)$

Donc $\exists B'(B' \notin (B))$ signifie : $\exists c\ (\neg\neg\exists f(f$ <u>reg-conv</u> $c) \& \neg\exists f(f$ <u>reg-conv</u> $c)$

ce qui est impossible (cf. §2.3 (9). Et donc (j') est démontrée.

Les formules (d'), (e'), (e'$_1$), (f') et (f'$_1$) découlent directement du théorème de Specker.

Remarquons enfin que (h') découle de (g') et (i') de (k') : Il est clair que $(\underline{\mathfrak{A}} \neq B)$ concernant une identité graphique est une formule normale. Donc (h') équivaut à (h_1) $\neg\exists B\, \forall A\, \neg\neg(\underline{\mathfrak{A}} \neq B)$ d'où (h_2) $\neg\exists B\, \forall \mathfrak{A}\, \neg(\mathfrak{A} \doteq B)$.

Donc : (h_3) $\neg\exists B\, \neg\exists \mathfrak{A}\ (\mathfrak{A} \doteq B)$ par §2.3 (3).

Donc : (h_4) $\forall B\, \neg\neg\exists \mathfrak{A}\ (\mathfrak{A} \doteq B)$ par §2.3 (3) qui est une conséquence de $\forall B\ \exists \mathfrak{A}\ (\mathfrak{A} \doteq B)$.

Supposons maintenant que l'on ait $(*)$ $\exists \Phi\, \forall B'\, (\Phi(B) \in (\mathfrak{A}) \,\&\, \underline{\Phi(B)} \doteq B)$.

Cela signifie : $\exists \Phi\, \forall c\, (\exists f(f \underline{\text{ reg-conv }} c) \supset (\overline{\Phi(c)} \underline{\text{ reg-conv }} \underline{\Phi(c)} \,\&\, \underline{\Phi(c)} = c))$

où c est une variable de mots dans N_3, car $c \in (\mathfrak{A})$ signifie $\bar{c}$ $\underline{\text{reg.Conv}}$ $\underline{c}$ (on suppose c de la forme $P \diamond Q$).

Donc $\exists \Phi\, \forall c\, \neg\neg\exists f(f \underline{\text{ reg.conv.}} c) \supset \neg\neg(\overline{\Phi(c)} \underline{\text{ reg.conv }} \underline{\Phi(c)} \,\&\, \underline{\Phi(c)} \doteq c))$.

Or la formule $(\overline{\Phi(c)} \underline{\text{ reg.conv.}} \underline{\Phi(c)} \,\&\, \underline{\Phi(c)} \doteq c)$ est normale. Donc on a :

$$\exists \Phi\, \forall c(\neg\neg\exists f(f \text{ reg conv } c) \supset (\overline{\Phi(c)} \underline{\text{ reg.conv.}} \underline{\Phi(c)} \,\&\, \underline{\Phi(c)} \doteq c))$$

qui est une reformulation de :

$$(**) \quad \exists \Phi\, \forall B'(\Phi(B') \in (\mathfrak{A}) \,\&\, \underline{\Phi(B')} \doteq B').$$

Donc $(*)$ implique $(**)$.

Or de $(**)$ il découle aussitôt que :

$$\forall B'(\exists f(f \underline{\text{ reg-conv }} B'))$$

C.à.d. $$\forall B'(B' \in (B)).$$

Donc : (k') implique (i') par contraposition.

Il nous reste donc à prouver (g') et (k').

Démonstration de (g').

Cette formule s'écrit aussi :

(0) $\forall c\ (\exists f(f\ \underline{\text{reg.conv.}}\ c) \supset \exists T\ ((\overline{T}\ \underline{\text{reg.conv.}}\ \underline{T}) \& \underline{T} \doteq c))$

où c et T est une variable de mots dans N_3, c de la forme P ou P$\diamond$ et T de la forme P ou P$\diamond$Q.

La formule entre parenthèses dans (0) se reformule constructivement (cf. [ ] §4) en : $\exists\sigma\ \forall g\ (g\ \underline{\text{reg.conv.}}\ c \supset (\overline{\sigma(g)}\ \underline{\text{reg.conv.}}\ \underline{\sigma(g)} \& \underline{\sigma(g)} \doteq c))$

car la formule considérée s'interprète : de la construction d'un f tel que f reg.conv. c on déduit la construction d'un T tel que $\overline{T}$ reg.conv. $\underline{T} \& \underline{T} \doteq c$, donc on dispose d'un algorithme passant de la construction d'un tel f à celle de T. Donc (0) devient :

$$\forall c\ \exists\sigma\ \forall g\ (g\ \underline{\text{reg.conv.}}\ c \supset (\sigma(g)\ \underline{\text{reg.conv.}}\ \underline{\sigma(g)} \& \underline{\sigma(g)} \doteq c))$$

qui est de la forme $\forall c\ \exists\sigma\ A$ où A est une formule normale. En vertu de §2.4 (iii) la reformulation constructive de (0) est donc

(00) $\exists\Phi\ \forall cg\ (g\ \underline{\text{reg.conv.}}\ c \supset (\overline{\langle\Phi(c)\rangle(g)}\ \underline{\text{reg.conv.}}\ \underline{\langle\Phi(c)\rangle(g)} \& \underline{\langle\Phi(c)\rangle g} \doteq c))$.

Or on construit aisément un algorithme B tel que :

$\langle B(c)\rangle(g) \doteq c$ si c est rationnel

$\langle B(c)\rangle(g) \doteq c\diamond\varepsilon g 3$ si c n'est pas rationnel.

Alors si on remplace Φ par B dans B ((00) est de la forme $\exists\Phi\ B$), on obtient une formule vraie. □.

Démonstration de (k') (cf. 6.5. Tseïtine [16] pp. 370-373).

Soit $\mathbb{A}$ un algorithme du type $(H \to K)$ décrivant une suite de segments rationnels emboîtés, c.à.d. tels que :

$$\forall n \ (\mathbb{A}_n \subset \mathbb{A}_{n+1}).$$

Soient $\mathfrak{p}$ et σ deux algorithmes, $\mathfrak{p}$ du type $(H \to p)$ énumérant sans répétition tous les rationnels, l'autre σ, du type $(p \to H)$ tel que

$$\forall a \ (\mathfrak{p}(\sigma(a)) \doteq a).$$

On peut donner divers moyens de construire $\mathfrak{p}$ et σ : par exemple en passant par l'intermédiaire de la forme irréductible et en classant les rationnels irréductibles de la forme p/q selon la valeur de $|p| + q$ et procéder comme on l'a fait pour construire des algorithmes énumérant les mots dans un alphabet donné.

On définit des algorithmes α, β et γ de la façon suivante :

$$\gamma(n) \doteq \mu m(\mathfrak{p}(m) \in \mathbb{A}_n)$$

$$\alpha(n) \doteq \mathfrak{p}(\gamma(n)) .$$

Si $\mathbb{A}_0 \doteq a \ \Delta \ b$ et $\ell = \mu j(j > b-a)$ (j entier)

$$\beta(n) \doteq \mu j(j > \max_{0 \leqslant m \leqslant \ell . 2^n} (a + \frac{m}{2^k})).$$

On vérifie alors que ces algorithmes sont respectivement du type $(H \to p)$, $(H \to H)$ et $(H \to H)$ et qu'ils ont les propriétés :

(i) $\forall n \ \gamma(n) \leqslant \gamma(n+1)$

(ii) $\forall n \ \gamma(n+1) = \gamma(n) \supset \alpha(n+1) = \alpha(n)$

(iii) $\forall k \, m \, n \ (\gamma(m) \geqslant \beta(k) \ \& \ \gamma(n) \geqslant \beta(k) \supset |\alpha(m) - \alpha(n)| < 2^{-k})$.

Les propriétés (i) et (ii) sont immédiates.

Supposons $\gamma(n) \geqslant \beta(k)$, alors pour tout $m \leqslant \ell.2^k$ on a $\sigma(a+\frac{m}{2^k}) < \gamma(n)$ et donc, par définition de γ, $\rho(\sigma(a+\frac{m}{2^k})) \notin A_n$ c.à.d. $a+\frac{m}{2^k} \notin A_n$. Si λ_n désigne la longueur de A_n*). On en déduit $\lambda_n < 2^{-k}$. Donc, par définition de $\mathfrak{D}$, et par la propriété de A, on a $\gamma(m) \geqslant \beta(k)$ et $\gamma(n) \geqslant B(k)$ impliquent que $|\mathfrak{D}(m)-\mathfrak{D}(n)| < 2^{-k}$ car $\mathfrak{D}_m \in A_n$ (on peut supposer $m \geqslant n$), ce qui est (iii).

Nous allons montrer que le mot $\alpha = \varepsilon\alpha_{30}$ est un quasi-réel.

Supposons que α n'admet pas de régulateur de convergence :

Alors : $\neg\exists n\, \forall m(m \geqslant n \supset \mathfrak{D}(m) = \alpha(n))$.

Sinon l'algorithme transformant tout mot dans $\{0,1\}$ en n serait un tel régulateur de convergence. Donc, en vertu de (ii) on a :

$$\neg\exists n\, \forall m\ (m \geqslant n \supset \gamma(m) = \gamma(n)).$$

Comme $\gamma(m) = \gamma(n)$ est une formule normale, ceci équivaut à :

$$\neg\exists n\, \neg\exists m\ (m \geqslant n \,\&\, \gamma(m) \neq \gamma(n)).$$

C.à.d. par §2.3 (3)

$$\forall n\, \neg\neg\exists m\ (m \geqslant n \,\&\, \gamma(m) \neq \gamma(n)).$$

Or : $\forall m\ (A \vee \neg A)$ où A désigne $(m \geqslant n \,\&\, \gamma(m) \neq \gamma(n))$ car A est exprimable algorithmiquement. Donc, par le principe de Markov,

$$\forall n\, \exists m\ (m \geqslant n \,\&\, \gamma(m) \neq \gamma(n))$$

et en vertu de (i)

$$\forall n\, \exists m\ (m \geqslant n \,\&\, \gamma(m) > \gamma(n)).$$

*) si $a_n \Delta b_n = A_n$, alors par définition $\lambda_n = b_n - a_n$; on a toujours $\lambda_n \geqslant 0$

Donc l'algorithme $\mathfrak{d}$ ainsi défini :

$$\mathfrak{d}(n) \simeq \mu m(\gamma(m) > n)$$

est applicable à tout entier naturel n .

Alors, par la condition (iii), l'algorithme $\mathfrak{d}\circ\beta$ est un régulateur de convergence de α . Donc, α est un quasi-réel. De plus, si x est un réel commençant par le mot $\varepsilon\alpha\mathfrak{z}\diamond$, alors du fait que :

$$\forall m(m \geqslant n \supset \alpha(m) \in \mathbb{A}_n)$$

on a $x \in \mathbb{A}_n$ et cela pour chaque n . On exprime ce fait en disant que le quasi réel x appartient conditionnellement à $\mathbb{A}_n$. On a donc établi :

(*) Pour chaque suite de segments rationnels emboîtés, on peut construire un quasi réel appartenant conditionnellement à chaque segment de la suite considérée

Considérons maintenant l'algorithme $\mathfrak{A}$ construit dans la démonstration du lemme fondamental (cf. §3.2) et soit $\mathfrak{B}$ son algorithme de développement canonique. On définit pour tous H et n un algorithme $\mathfrak{C}'$ applicable au mot $H\square n$ par :

$$\mathfrak{C}'(H\ n) \simeq \begin{cases} 0\ \Delta\ 3 & \text{si } \mathfrak{B}(H\square n) \neq \Lambda \\ (1+\mathfrak{A}(H)0|)\ \Delta\ (2+\mathfrak{A}(H)0|) & \text{si } \mathfrak{B}(H\square n) = \Lambda \end{cases}$$

Lorsque $B(H\square n) = \Lambda$ on a donc $C'(H\square n) = 2\ \Delta\ 3$ si $\mathfrak{A}(H) = \Lambda$ et $\mathfrak{C}'(H\square n) \simeq 0\ \Delta\ 1$ si $\mathfrak{A}(H) = -$. Pour H fixé, l'algorithme qui à chaque n associe $\mathfrak{C}'(H\square n)$ est une suite de segments rationnels emboîtés. Si on note (cf. [ ]) $\tilde{\mathfrak{C}}_{H\square}$ cet algorithme, on note $\mathbb{D}$ l'algorithme :

$$\mathbb{D}(H) \simeq \varepsilon\tilde{\mathfrak{C}}'_H 3$$

Si on suppose que pour toute suite de segments rationnels emboîtés (une telle suite est définie par une formule normale) on peut construire un réel appartenant à tous les segments de la suite, en vertu de l'interprétation constructive des énoncés, on peut construire un algorithme $\mathbb{E}$, applicable au code de tout algorithme définissant une suite de segments rationnels emboîtés et le transformant en un réel appartenant à tous les segments de la suite.

On a donc : $\forall n \quad \mathbb{E}(\mathbb{D}(H)) \in \mathbb{C}'(H \square n)$.

Si $\mathfrak{A}(H) \doteq \Lambda$ pour un certain n $\mathbb{C}'(H \square n) \doteq 2 \Delta 3$ et donc $\mathbb{E}(\mathbb{D}(H)) \geqslant 2$.

De même, si $\mathfrak{A}(H) \doteq -$ alors $\mathbb{E}(\mathbb{D}(H)) \leqslant 1$. Soit alors

$\mathbb{G}(H) \simeq \mathbb{J}(\mathbb{C}(\mathbb{E}(\mathbb{D}(H)) \square 1 \square 2))$ (cf. §4.1.3 corollaire du lemme technique 1) où $\mathbb{J}$ est défini par : $\mathbb{J}(0|) \doteq \Lambda$ et $\mathbb{J}(0||) \neq \Lambda$. Alors $\mathbb{G}$ est applicable à tout mot H. De plus si $\mathfrak{A}(H) \doteq \Lambda$ alors $\mathbb{E}(\mathbb{D}(H)) \geqslant 2$, donc $\neg\mathbb{E}(\mathbb{D}(H)) < 2$ et donc on ne peut avoir $\mathbb{C}(\mathbb{E}(\mathbb{D}(H) \square 1 \square 2) \doteq 0||$. Donc $\mathbb{G}(H) \doteq \Lambda$. De même, si $\mathfrak{A}(H) \doteq -$, alors $\mathbb{G}(H) \neq \Lambda$.

Or on sait, par construction de $\mathfrak{A}$, qu'un tel algorithme $\mathbb{G}$ ne peut exister. On a donc établi :

(**) Il n'est pas vrai que pour toute suite de segments rationnels emboîtés on puisse construire un réel appartenant à chaque segment de la suite.

Supposons maintenant : $(\overset{*}{**})$ $\forall B'(B' \in (B))$.

Soit $\mathbf{A}$ une suite de segments rationnels emboîtés. Par (**) on peut lui associer un quasi-réel α appartenant conditionnellement à chaque segment de la suite. Par $(\overset{*}{**})$ α est un réel faible et donc, par (g') (on considère un régulateur de convergence de α) on peut trouver un mot Q tel que le mot αQ soit un réel. Et donc par (*) ce réel appartient à chaque segment de la suite $\mathbf{A}$. Ceci contredit (**). Donc on a $\neg\forall B'(B' \in (B))$. $\square$.

Appendice 2

Sur les formes du principe de Markov

Dans le §2 nous avons dit que le principe de Markov :

$$(1)\quad (\forall x\ (A \vee \neg A) \supset (\neg\neg \exists x\ A \supset \exists x\ A))$$

est équivalent à l'énoncé :

"si le processus d'application de l'algorithme $\mathbb{A}$ au mot donné P n'est pas indéfiniment prolongeable, l'algorithme $\mathbb{A}$ est applicable au mot P".

Ce dernier énoncé peut se résumer par la formule :

$$(2)\quad \neg\neg !\ \mathbb{A}(P) \supset !\ \mathbb{A}(P)$$

Nous allons établir cette équivalence.

a. (1) implique (2).

En effet, si on considère $\mathbb{B}$ l'algorithme de développement canonique de $\mathbb{A}$, alors

$$!\ \mathbb{A}(P) \equiv \exists n\ (\mathbb{B}(P \square n) \doteq \Lambda)$$

Or il est clair que :

$$\forall n\ (\mathbb{B}(P \square n) \doteq \Lambda \quad \mathbb{B}(P \square n) \neq \Lambda)$$

puisque pour tout mot $P \square n$, $\mathbb{B}$ est applicable à $P \square n$.

Donc (2) résulte bien de (1).

b. (2) implique (1).

Nous allons montrer que : chaque fois que $\forall x(A \vee \neg A)$ est vraie, on peut déduire de $\neg\neg \exists x\ A$ l'assertion $\exists x\ A$.

En effet : $\forall x(A \vee \neg A)$ signifie, d'après le principe §2.4(i) qu'on peut construire un algorithme $\mathbb{A}$ applicable à tous les mots que représente la variable x (on suppose ici qu'il ne s'agit pas d'une variable restreinte), prenant les valeurs $0|$ ou $0||$ et tel que l'on ait

$$\mathbb{A}(x) \doteq 0| \supset A$$

$$\mathbb{A}(x) \doteq 0|| \supset A$$

Il suffit de voir qu'on peut, à partir de $\mathbb{A}$, construire un algorithme $\mathbb{B}$ tel que $$\exists x\,(\mathbb{A}(x) \doteq 0|) \equiv !\,\mathbb{B}(P)$$ pour un mot P dans l'alphabet de $\mathbb{B}$.

Considérons alors $\mathbb{N}$ un algorithme énumérant sans répétition tous les mots dans l'alphabet de la variable x. Soit A un alphabet quelconque. On pose, pour tout mot P dans A :

$$\mathbb{B}(P) \simeq \mu n(\mathbb{A}(\mathbb{N}(n)) \doteq 0|)$$

Il est alors évident que :

$$\exists x\,(\mathbb{A}(x) \doteq 0|) \equiv \exists n\,(\mathbb{A}(\mathbb{N}(n)) \doteq 0|) \equiv !\,\mathbb{B}(P).$$

Donc de $\neg\neg\exists x(\mathbb{A}(x) \doteq 0|)$ on tire $\exists x\,(\mathbb{A}(x) \doteq 0|)$ puisque de $\neg\neg!\,\mathbb{B}(P)$ on tire $!\,\mathbb{B}(P)$.

Bibliographie

[1] CHANINE, N. A. Konstruktivnye veščestvennye čisla i konstruktivnye funkcional'nye npostranstva. Trudy Mat. inst. Steklova, T. 67 (1962), 15-294.

[2] ———— Ob algorifme konstruktivnoj rassšifrovki matematičeskix suždenij. Zeitschr. f. math. Logik und Grundlagen d. Math. Bd 4 (1958), 293-303.

[3] ———— O Konstruktivnom nonimanii matematičeskix suždenij. Trudy Mat. inst. Steklova, T. 52 (1958), 226-311.

[4] GENTZEN, G. Untersuchungen über das logische Schliessen. II. Math. Zeitschr. 39, n° 3 (1934), 405-431.

[5] HEYTING, A. Die formalen Regeln der intuitionistischen Logik. Sitzungsber. Preuss. Akad. Wiss. phys-math. Kl. 42-56, 57-71 (1930).

[6] ———— Intuitionism an Introduction. Amsterdam 1966.

[7] KLEENE, S. C. and VESLEY, R. E. Foundations of intuitionistic mathematics. Amsterdam 1965.

[8] KOLMOGOROV, A. N. O principe "tertium non datur". Matem. Sbor. 32, n° 4, (1925), 646-667.

[9] MARKOV, A. A. Ob odnom principe konstruktivnoj matematičeskoj logiki. Trudy III Vsesojvznogo mat. sezda. T. II (1956), 146-147.

[10] ———— Teoria algorifmov. Trudy Mat. Inst. Steklova, T. 42 (1954).

[11] NAGORNY, N. M. K usileniu teoremy privedenija teorii algorifmov. Doklady Akademii Nauk SSSR, T. 90, n° 3 (1953), 341-342.

[12] ———— Nekotovye obobščenija ponjatija normal'nogo algorifma. Trudy Mat. Inst. Steklova T. 52 (1958), 7-65.

[13] ORLOVSKY; E. S. Nekoturye voprusy teorii algorifmov. Trudy Mat. inst. Steklova, T. 52 (1958), 140-171.

[14] SPECKER, E. Nicht konstruktiv beweishar Sätze der Analysis. Journ. of Symp. Logic. 14, n° 3 (1949), 145-158.

[15] TSEITINE, G. S. Algorifmičeskie operatory v konstruktivnyx metričeskix prostranstvax. Trudy Mat. inst. Steklova, T. 67 (1962), 295-361.

[16] ———— Teoremy o srednem značenii v konstruktivnom analyze. Trudy Math. inst. Steklova, T. 67 (1962), 362-384.

[17] ZASLAVSKY, I. D. Nekotorye suojstva konstruktivnyx veščestvennyx čisel i konstruktivnyx funkcij. Trudy Mat. inst. Steklova, T. 67 (1962), 385-457.