International Journal of Innovation and Applied Studies
ISSN 2028-9324 Vol. 21 No. 1 Aug. 2017, pp. 103-123

http://www.ijias.issr-journals.org/

L'arithmétisation et la métathéorie

***A.-Roger LULA BABOLE*[1-2]**

[1]Département de Mathématiques et Informatique, Université de Kinshasa, RD Congo

[2]Faculté de Philosophie, Université Catholique du Congo, RD Congo

ABSTRACT: The resort in arithmetization of arithmetic reaches to construct inside of arithmetic a proposition which confirm its self-indemonstrability. In substance, the proof of coherence presupposes a form of induction in transfinite order for proving the coherence of arithmetic which is the finite order. It is dealing recursive functions which have the properties for all attribute values system to determine them by the means of finite type procedure. It comes to recursive arithmetic, to translate the elements of metatheory formal system.

KEYWORDS: arithmetization, metatheory, recursivity, recursive arithmetic, iteration, metamathematic, incompletenes.

RÉSUMÉ: Le recours à l'arithmétisation de l'arithmétique amène à construire à l'intérieur de l'arithmétique une proposition qui affirme sa propre indémontrabilité. Au fond, la preuve de la cohérence présuppose une forme d'induction de l'ordre transfini aux fins de prouver la cohérence de l'arithmétique qui est de l'ordre fini. Il s'agit donc des fonctions récursives qui ont des propriétés pour tout système de valeurs attribué à ses arguments de les déterminer aux moyens d'une procédure de type fini. Il revient à l'arithmétique récursive de traduire les éléments de la métathéorie du système formel.

MOTS-CLEFS: Arithmétisation, métatheorie, récursivité, arithmétique récursive, itération, métamathématique, incomplétude.

INTRODUCTION

GÖDEL a essayé de formaliser la métamathématique dans une théorie si puissante afin de permettre à la métathéorie de s'exprimer et de se construire en un nombre fini de procédures. Les prédicats qui ont servi des preuves arithmétiques aux formules sont traduits en propriétés arithmétiques en passant d'abord par les propriétés métamathématiques.

La méthode de l'arithmétisation est utilisée dans l'opération de traduction aux fins d'éliminer des paradoxes sémantiques. Cette méthode consiste, à associer de façon biunivoque, des entiers aux objets du système formel étudié.

Le concept de fonctions récursives est au cœur de toute la recherche conduisant à la preuve du théorème de GÖDEL. Ce sont des fonctions arithmétiques dont la propriété s'énonce comme suit : « pour tout système des valeurs qu'on donne aux arguments, on peut préciser la valeur au moyen de procédures de type fini ».

1 L'ARITHMÉTISATION ET LES FONCTIONS RÉCURSIVES

Il s'agit à ce niveau d'établir les rapports entre le principe d'arithmétisation et la notion de fonction récursive. Pour opérer une traduction du genre à éliminer certaines antinomies sémantiques, GÖDEL a utilisé la méthode de l'arithmétisation.

En effet, elle consiste à associer, de façon biunivoque, des entiers aux objets du système formel examiné. Grâce à elle, il est possible de formuler, au moyen des procédés de l'arithmétique, une classe très étendue d'énoncés et de raisonnements métathéoriques.

L'idée de l'arithmétisation se trouve déjà chez LEIBNIZ. Mais c'est bien GÖDEL qui l'a mise en œuvre pour la première fois de façon systématique. L'utilisation pratique de cette méthode repose sur la théorie des fonctions récursives.

1.1 FONCTIONS RÉCURSIVES

Une fonction récursive est une fonction arithmétique qui a la même propriété selon laquelle pour tout système de valeurs attribué à ses arguments, il y a lieu de déterminer sa valeur au moyen d'une procédure de type fini. La notion de fonction récursive a servi à analyser le contenu constructif de certains théorèmes classiques et à étudier le finitisme des formalismes logiques.

Rappelons qu'une sous-classe des fonctions récursives, les fonctions récursives primitives, a été définie et utilisée par HERBRAND et par GÖDEL en 1931. Mais il est à remarquer que l'exponentielle généralisée d'ACKERMANN est récursive sans être récursive primitive.

Les récursions primitives ou réductibles à des récursions primitives s'inscrivent dans le cadre de l'arithmétique finitiste, que celle-ci soit formalisée, c'est-à-dire l'arithmétique récursive ou qu'elle soit contentuelle. On notera que la fonction d'ACKERMANN est récursive simultanément en plusieurs variables simples. Mais l'une de ces fonctions récursives simples est la « Funktionenfunktion » itération de f qui n'est pas récursive ordinaire, puisqu'elle est dépendante d'une variable qui n'est pas arithmétique [1].

1.1.1 LA RÉCURSIVITÉ

On définit la classe des fonctions récursives de la même manière que celle de la gödelienne. D'abord, il y a les fonctions de base qui sont toutes des fonctions totales :

(I) La fonction constante Cnt, dont la valeur est toujours 0 (Const α=0) ;

(II) La fonction successeur Scc, $Scc(x) = x + 1$;

(III) La fonction de sélection :Sel$^{(in)}$, isolant la n^{ième} variable (Sel$^{(in)}$(x_1x_2,…, x_i…x_n=x_i) pour tous les $x_1, \dots, x_n$.

Toutes les fonctions élémentaires de l'arithmétique s'inscrivent dans cette classe (addition, différence, multiplication, exponentiation, etc.). Ensuite, nous avons les règles suivantes :

(IV) La substitution f(x_1, …, x_n) = g(h_1(x_1, …, x_n), …, h_m(x_1, …, x_n)).

(V) La récursionf(x_1, …, x_n, 0) = g(x_1, …, x_n), …, y, f(x_1, …, x_n, y).[1]

(VI) L'opérateur μ. Supposons que g(x_1, …, x_n, y) est une fonction telle que pour tous x_1, …, x_n, il y a au moins un y tel que g(x_1, …, x_n, y) = 0.

Nous signifions par μy(g(x_1, …, x_n, y) = 0) le plus petit nombre y satisfaisant :

g(x_1, …, x_n, y) = 0 si f(x_1, …, x_n) = μy(g(x_1, …, x_n, y) = 0.

Nous disons alors que f(x_1, …, x_n) est obtenue de g(x_1, …, x_n, y) par l'opérateur μ.

Il est pertinent de faire remarquer [2] qu'il y a des fonctions calculables qui ne sont pas récursives primitives (FRP). Un raisonnement « diagonal » en apporte la preuve. Soit l'ensemble des fonctions récursives primitives d'une variable. On peut numéroter d'une manière effective chaque élément de cet ensemble. Si on note φn la FRP portant le numéro n, alors il est possible de numéroter l'ensemble des FRP à une variable comme suit :$\varphi_0, \varphi_1, \varphi_2, \dots . \varphi_n$. En considérant la fonction ψ définie ainsi : $\exists_n, \psi(n) = \varphi_n + 1$

En clair, la FRP est donc calculable. Cela signifie que la numérotation des FRP est effective. D'où, il y a lieu à partir de n, de déterminer φ_n. Il en résulte que $\varphi_n$étant récursive primitive, $\varphi_n\ (n)$ est calculable pour toute valeur de n, et cela vaut aussi

[1] La récursion intervient dans la démonstration du premier théorème d'incomplétude, au point qu'on se permet d'affirmer que l'incomplétude et l'indécidabilité de l'arithmétique sont intimement liées.

pour $\varphi_n(n) + 1$. Mais il faut noter que ψ n'est pas une FRP, sinon, ψ figurerait dans la numération, à un certain rang m et, peut être, on dit que ψ est identique à la fonction φ_m. Pour tout n, on aurait,

$\varphi(n) = \varphi_m(n) = \varphi_m(n) + 1$; et pour $n = m$

$\varphi_m(m) = \varphi_m(m) + 1$; ce qui amène à l'absurdité.

D'où, finalement, ψ est non récursive primitive quand bien même elle est calculable. Cela est de même de φ.

1.1.2 LES SCHÉMAS DE RÉCURSION

Soient des fonctions partielles $G: \mathbb{N}^n \to \mathbb{N}$ et $H : \mathbb{N}^{n+2} \to \mathbb{N}$. On appelle la fonction définie par récursion à partir de G et H la fonction

$F: \mathbb{N}^{n+1} \to \mathbb{N}$ définie par

$F(x_1, x_{2,} \dots, x_n, 0) = G(x_1, x_{2;} \dots, x_n)$

$F(x_1, x_2 \dots, x_n(Scc\alpha)) = H(x_1, x_2 \dots, x_n\alpha)$

pour tout $(x_1, x_{2,} \dots, x_{n,}\alpha) \in \mathbb{N}^{n+1}$ pour lequel la définition a une signification.

1.1.2.1 LE SCHÉMA DE RÉCURSION PRIMITIVE

On présente en effet le schéma de récursion primitive comme suit :

$F(x_1, x_{2,} \dots, x_n, 0) = G(x_1, x_{2,} \dots, x_n)$

$H(x_1, x_2 \dots, x_n(Scc\alpha) = H(x_1x_2 \dots, x_n\alpha), x_1, \dots x_n\alpha)$

Ce schéma nous montre que F représente une fonction ayant (n+1) arguments, G une fonction de n arguments et H est une fonction ayant (n+2) arguments telle que $x_1, x_{2,} \dots, x_n$ sont des variables et α est un symbole pouvant être remplacé par n'importe lequel des entiers. On suppose que les fonctions g et h sont déjà définies.

1.1.2.2 AUTRES SCHÉMAS DE RÉCURSION

En ce qui concerne les autres schémas de récursion, on peut énumérer trois types, à savoir :

- ***Le schéma de récursion indirecte***, dans lequel il y a dépendance de la fonction vis-à-vis de la variable par l'intermédiaire des termes déjà définis.

  Illustrons cela :

  $FO = a$

  $F1(Sccn) = Gn[F(H_1n)][F(H_2n)] \dots [F(H_kn)]$

  Ce schéma nous indique que G représente une fonction déjà définie à (k+1) arguments, H_1, H_2…H_k sont des fonctions déjà définies à un argument où on a $H_1n \leq H_2n \leq \cdots H_kn \leq n$

- ***Le schéma de récursion simultanée***, dans lequel sont définies à la fois deux ou plusieurs fonctions dépendant les unes des autres.

  Illustrons cela donc :

  $F_1O = a$

  $F_1(Sccn) = G_1n(F_1n)(F_2n)$

  $F_2O = b$

  $F_2(Sccn) = G_2n(F_1n)(F_2n)$

  On voit que dans ce schéma$F_1\ et\ F_2$ sont donc des fonctions devant être définies, $G_1\ et\ G_2$ représentent les fonctions déjà définies ayant trois arguments.

- ***Le schéma de récursion croisée***, dans lequel on représente la fonction d'ACKERMANN.

  En effet, la fonction d'ACKERMANN[2]illustre un cas d'une fonction définie par un schéma de récursion croisée. Son schéma de définition utilise trois fonctions auxiliaires, qui sont alors définies par des schémas de récursion primitive :

  $Akn_a\,0 \quad =0$

  $Ak_{na}\,(Scc\ n) \quad =1,$

  $Ak_{nb}\,0=1$

  $Akn_b(Scc\ n) \quad =0,$

  $Akn_c\,0=0$

  $Akn_c\,(Scc\ n) \quad =n.$

  La fonction Ak_n se définit comme suit :

  $Ak_{n\alpha\beta}\ 0=\alpha+\beta$

  $Ak_\alpha\,0\ (Scc\ n)= Akn_b(Scc\ n)+[\alpha x Ak_{na}\ (Ak_{ncn}]$

  $Ak_{n\alpha}(Scc\ n)=Ak_{n\alpha}[Ak_n\beta(Scc\ n)]n$

  Ce qui donne en effet :

  $Ak_n\beta 1=\alpha x\beta$

  $Ak_n\beta 2=\alpha^\beta$

Il nous faut préciser que la deuxième ligne du schéma nous fait passer de la valeur de la fonction, pour la valeur n du 3eme argument, à la valeur de la fonction pour la valeur (n+1) de cet argument. Quant à la troisième ligne du schéma, elle nous fait passer de la valeur de la fonction pour la valeur n, du deuxième argument, à la valeur de la fonction pour la valeur de (n+1) de cet argument.

La fonction d'ACKERMANN est définie par récurrence double avec imbrication des appels récursifs. Mentionnons notamment que :

Ack (0, x) = x + 2

Ack (1,0) = 0

Ack (n+2,0) = 1

Ack (n+1, x+1) = Ack (n, Ack (n+1, x)).

Notons en effet que chaque Ack_n: x→ Ack (n,x) est récursive primitive. Cela signifie que la fonction Ack_{n+1} s'obtient lorsqu'on itère la fonction Ack_n

Ack_{n+1} (x+1) = Ack_n (Ack_{n+1} (x)).

Il s'ensuit que toutes les fonctions Ack_n sont croissantes, et cette croissance se réalise de plus en plus rapide de telle sorte que

$Ack_1(x) = 2x, Ack_2(x) = 2^x$

$Ack_3(x) = 2^{2^{\cdot^{\cdot^{2}}}\}x}$

[2] Le théorème de GÖDEL datant de 1931a contraint ACKERMANN à revisiter ses points de vue de 1924 (HILBERT et ACKERMANN) et cela en complétant en 1940 une preuve de non-contradiction avec le calcul ϵ qui s'inspire de la preuve par induction transformée de GENTZEN (1936) En clair, sa preuve dépasse largement le cadre de l'arithmétique contentuelle.

Alors on dit qu'une fonction $f: \mathbb{N} \to \mathbb{N}$*domine* une fonction $g: \mathbb{N}^P \to \mathbb{N}$ si f est supérieure à g à partir d'un certain rang. Autrement dit, on a :

$\exists k \in \mathbb{N} \forall \overline{x} \in N^P$ g ($\overline{x}$) ≤ f(sup ($\overline{x}$, K)).

Donc, il est possible de montrer que pour toute fonction primitive récursive, il existe n tel que Ack_n domine f, telle est la déduction.

En d'autres termes, on peut définir la fonction $d'ACKERMANN$ comme suit :

1. $Si\ m = 0\ alors\ Ack(m,n) = n + 1$
2. $Si\ m \neq 0\ mais\ n = 0, alors\ Ack(m,n) = Ack(m-1,1)$
3. $Si\ m \neq 0\ et\ n \neq 0, alors\ Ack(m,n) = Ack[\grave{u} - 1, Ack(m,n-1)]$

En effet, la fonction d'ACKERMANN est une fonction à deux arguments $m\ et\ n$ tels que chacun d'entre eux peut prendre des valeurs 0,1,2,3, … Donc, cette définition est récursive puisqu'elle se réfère à elle – même dans les parties $(2)\ et\ (3)$. Evidemment $Ack(m,n)$ n'est donné de façon explicite que lorsque $m = 0$. Ainsi, les critères de base sont les paires $(0,0), (0,1), (0,2), \dots (0,n), \dots$

Exemple : calculer $Ack(1,3)$ en appliquant la définition de la fonction d'ACKERMANN.

$[il\ y\ a\ 15\ étapes\ pour\ arriver\ à\ la\ solution\ finale]$

(1) $Ack(1,3) = Ack[0, Ack(1,2)]$

(2) $Ack(1,2) = Ack[0, Ack(1,1)]$

(3) $Ack(1,1) = Ack[0, Ack(1,0)]$

(4) $Ack(1,0) = Ack(0,1)$

(5) $Ack(0,1) = 1 + 1 = 2$

(6) $Ack(1,0) = 2$

(7) $Ack(1,1) = Ack(0,2)$

(8) $Ack(0,2) = 2 + 1 = 3$

(9) $Ack(1,1) = 3$

(10) $Ack(1,2) = Ack(0,3)$

(11) $Ack(0,3) = 3 + 1 = 4$

(12) $Ack(1,2) = 4$

(13) $Ack(1,3) = Ack(0,4)$

(14) $Ack(0,4) = 4 + 1 = 5$

(15) $Ack(1,3) = 5$

Pour ACKERMANN, on ne peut donc pas définir cette fonction au moyen d'un schéma de récursion primitive. Partant de ces fonctions, l'idée de généraliser la notion de récursivité a hanté HERBRAND à utiliser deux ou trois équations soumises à certaines conditions bien déterminées [6].

La fonction d'ACKERMANN ainsi définie illustre un cas de fonction récursive générale. Cela va s'en dire que le concept de récursivité se généralise à d'autres catégories d'objets tels que les prédicats, les classes et les ensembles.

1.2 FONCTIONS ET PRÉDICATS RÉCURSIFS

Une fonction f est dite récursive, si et seulement si, elle peut être obtenue des fonctions de base par un nombre fini d'applications des règles (IV), (V) et (VI). Cela signifie que s'il existe une suite de fonctions $f_1, \dots, fn$ telles que, pour tout i, fi est, soit une fonction initiale, soit une fonction obtenue de fonctions $f_{j1}, \dots, f_{jr}$ avec $j_1, \dots, j_r < i$ par l'application des règles (IV), (V) ou (VI).

Une fonction est dite *primitivement récursive,* si elle est récursive et s'il est possible de l'obtenir sans l'application de la règle (VI).

1.2.1 LA REPRÉSENTATION

Une fonction est dite représentative d'un prédicat, si on a une fonction des mêmes arguments prenant la valeur 0, si le prédicat est vérifié ; la valeur 1, s'il ne l'est pas, et ne pouvant prendre d'autres valeurs. Donc la fonction F est dite fonction représentative du prédicat R si on a l'équivalence.

$$Rx_1x_2 \ldots x_n \leftrightarrow Fx_1x_2 \ldots x_n = 0$$

Prenons par exemple le prédicat divisible par 2 ayant pour fonction représentative le reste de la division par 2. Si un nombre entier peut être divisé par 2, le reste de sa division par 2 donne 0, s'il ne le peut être, le reste de sa division par 2 vaut 1.

On appelle donc prédicat récursif (primitif ou général) un prédicat ayant une fonction représentative récursive (primitive ou générale).

Un prédicat arithmétique $\Re(X_1, \ldots, X_n)$ est dit *exprimable* dans SFC si et seulement s'il existe une formule $\Re(x_1, \ldots, x_n)$ telle que pour tous nombres $k_1, \ldots, k_n$:

(1). $Si\ \ R(k_1,\ldots,k_n)\ \text{alors} \mapsto_k \Re(\bar{k}_1,\ldots,\bar{k}_n)$

(2). $Si\ \ \neg R(k_1,\ldots,k_n)\ \text{alors} \mapsto_k \neg\Re(\bar{k}_1,\ldots,\bar{k}_n)$

Une fonction arithmétique $f(x_1, \ldots, x_n)$ est dite représentable dans SFG si et seulement s'il existe une formule $\Re(x_1, \ldots, x_n, x_{n+1})$ de SFG telle que pour tous nombres $k_1, \ldots, k_n$ et m :

(1). $\text{Si}\ \ f(k_1,\ldots,k_n) = m\ \text{alors} \mapsto_k R(\bar{k}_1,\ldots,\bar{k}_n\ \bar{m});$

(2). $\left|-_k \exists_1 x_{n+1} \Re(\bar{k}_1,\ldots,\bar{k}_n, x_{n+1})\right.$

Les fonctions de base sont représentables.

La fonction $\beta(x_1, x_2, x_3) = \text{le reste de la division euclidienne de } 1+(x_3+1).x_2 \text{ par } x_3.$

1.2.2 PROPRIÉTÉS ARITHMÉTIQUES RÉCURSIVES

Pour conférer à la théorie des preuves la rigueur requise, GÖDEL essaye de la formaliser dans une théorie si puissante que la métathéorie y soit exprimable et qu'elle soit constructible en un nombre fini d'étapes.

Pour ce faire, GÖDEL a créé des prédicats pour appliquer les preuves arithmétiques aux formules. Il a traduit Ces propriétés métamathématique par des propriétés arithmétiques.

Ainsi, la fonction Γproprement dite doit avoir les propriétés métathéoriques.

1. x divise y, x : $x / y \cong \exists z(z \leq y \wedge y = z.x)$
2. x est un nombre premier, Pr (x) : $\Pr(x) \equiv \forall y((y \leq x \wedge y/x) \rightarrow (y = 1 \vee y = x))$
3. Pr (n, x) est le n^{ième} premier dans la suite des nombres premiers qui divise x :

   $\Pr(0,x) \equiv 0 \text{ et } \Pr(n+1,x) \equiv \varepsilon y(y \leq x \wedge \Pr(y) \wedge y/x \wedge y < \Pr(n,x))$
4. La fonction factorielle : 0 ! et (n+1)! = (n+1). n!
5. Pr(n) est le n^{ième} premier dans la suite des nombres premiers :

   $\Pr(n) \cong 0 \text{ et } \Pr(n+1) \cong \varepsilon y(y \leq x \wedge \Pr(n)! + 1 \wedge \Pr(y) \wedge \Pr(n))$
6. Terme (n,x) est le n^{ième} terme dans la suite numérique x :

$\text{Terme}(n,x) \cong \varepsilon y(y \le x \wedge \text{Pr}(n,x)^{y} / x \wedge \neg \text{Pr}(n,x)^{y+1} / x)$

7. l(x) est la longueur de la suite x :

$\text{Terme}(n,x) \cong \varepsilon y(y \le x \wedge \text{Pr}(n,x)^{y} / x \wedge \neg \text{Pr}(n,x)^{y+1} / x)$

8. x*y est la concaténation des deux suites finies x et y :

$x * y \cong \varepsilon z(z \le (\text{Pr}(l(x)+l(y)))^{x+y} \wedge \forall n(n \le l(x) \rightarrow \text{Terme}(n,x)$

$= \text{Terme}(n, z)) \wedge \forall n(0 < n \le l(y) \rightarrow \text{Terme}(n, y) = \text{Terme}(n + l(y), z)))$

9. R(x) est la suite du nombre x tout seul : $R(x) \cong 2^{x}$

10. E(x) est l'opération de mise du nombre x entre parenthèses : $E(x) \cong R(11) * x * R(13)$

11. x est une variable de type n, Var(n,x) :

$\text{Var}(n,x) \cong \exists z(13 < z \le x \wedge \text{Pr}(z) \wedge x = z^{n})$

12. x est une variable, Var(x) : $\text{Var}(x) \cong \exists n(n \le x \wedge \text{Var}(n))$

13. Neg(x) est la négation de x : $\text{Neg}(x) \cong R(5) * (x)^{*}) = R(5) * x$

14. Dis(x,y) est la disjonction de x et y :

$\text{Dis}(x, y) \cong E(x) * R(7) * E(y) \cong E(x) * R(7) * y$

15. Gen(x,y) est la généralisation de y par rapport à x :

$\text{Gen}(x, y) \cong R(x) * R(7) * E(y) \cong E(9) * R(x) * y$

16. N(n,x) est l'action qui place n symboles « ' » devant x :

$N(o, x) \cong x$ et $N(n+1, x) \cong R(3) * N(n, x)$

17. N(x) est le nombre de GÖDEL x : $N(x) \cong N(x,1)$

18. x est un signe de type 1, type 1(x) :

$\text{type}\,1(x) \cong \exists m \exists n(m, n \le x \wedge (m = 1 \vee Var(1, m)) \wedge x = N(n, R(m)))$

19. x est un signe de type n, $\text{type}_n(x)$:

$\text{type}_n(x) \cong ((n = 1 \wedge type1(x)) \vee (n > 1 \wedge \exists y(y \le x \wedge Var(n, x) \wedge x = R(y)))$

20. x est une formule élémentaire, Elf(x) :

$\text{Elf}(x) \cong \exists y \exists z \exists n(y, z, n \le x \wedge type_n(y) \wedge type_{n+1}(z) \wedge x = z * E(y)$

21. x s'obtient de la formule y par négation ou généralisation ou des formules y et z par conjonction Op(x, y, z) :

$\text{Op}(x, y, z) \cong (x = Neg(y) \vee (x = Dis(y, z) \vee \exists v(v \le x \wedge Var(v) \wedge x = Gen(v, y))$

22. x est une suite de formules, chacune est une formule élémentaire ou une des formules obtenue précédemment par l'opération Op(x,y,z), FR(x) :

$\text{FR}(x) \cong l(x) > 0 \wedge \forall n(0 < n \le l(x) \rightarrow \text{Elf}(\text{Terme}(n,x)) \vee \exists p \exists y(0 < p, qn \wedge$
$\text{Op}(\text{Terme}(n,x), \text{Terme}(p,y), \text{Terme}(q,z)))$

23. x est une formule, Form(x) :

$\text{Form}(x) \cong \exists n(n \le (\text{Pr}(l(x)^{2})) \wedge \text{FR}(n) \wedge x = \text{Terme}(l(n), n))$

24. La variable *v* est liée à la n^{ième} place dans x, liée (*v,n,x*) :

Liée(v,n,x) $\cong$ (Var(v) $\wedge$ Form(x) $\wedge \exists a \exists b \exists c$(a,b,c $\leq$ x $\wedge$ l(a)+1 $\leq$ n $\leq$ l(a)+l(Gén(v,b)) $\wedge x$
$= a * \overline{Gen}(v,b) * c \wedge$ Form(x))))

25. La variable v est libre à la n[ième] place dans x, Libre(*v,n,x*) :

Libre(v,n,x) $\cong$ (n $\leq l($x$) \wedge$ Var(v) $\wedge$ Form(x) $\wedge$ v = Teme(n,x) $\wedge \neg$Liée(v,n,x)

26. La variable v est libre dans x, Libre(v,x) :

Libre(v,x) $\cong \exists n$(n $\leq l($x$) \wedge$ Libre(v,n,x))

27. Su(x,n,y) est le résultat de la substitution du nième terme de x par y :

Su(x,n,y) $\cong \varepsilon$z(z $\leq$ (Pr(l(x)+l(y))$^{x+y}$ $\wedge \exists$u$\exists$v(u,v $\leq$ x $\wedge$ x = u * R(Terme(n,x)) * v $\wedge$ z
$=$ u * y * v $\wedge$ n = l(u)+1))

28. St(k,v,x) est la (K+1)[ième] place où v est libre dans x.

St(o,v,x) $\equiv \varepsilon$n(n $\leq$ l(x) $\wedge$ Libre(v,n,x) $\wedge \forall$p(n < p $\leq$ l(x) $\rightarrow \neg$Libre(v,p,x)))

St(K+1,v,x) $\equiv \varepsilon$n(n $\leq$ St(k,v,x) $\wedge$ Libre(v,n,x) $\wedge \forall$p(n < p $\leq$ St(x) $\rightarrow \neg$Libre(v,p,x)))

29. A(v,x) est le nombre de places où v est libre dans x :

A(v,x) $\equiv \varepsilon$n(n $\leq$ l(x) $\wedge$ *St*(n,v,x) = 0)

30. Sub$_k$(x,v,y) est le résultat de la substitution de v par y à la 1[ère], 2[ème], K[ième]places dans x :

Sub$_0$(x,v,y) $\equiv x$

Sub$_{K+1}$(x,v,y) $\equiv$ *Su*(*Sub*$_K$(*x*,*v*,*y*), *St*(*k*,*v*,*x*), *y*)

31. Sub(x,v,y) est le résultat de la substitution de v par y dans x :

Sub(x,v,y) $\equiv$ *Sub*$_{A(v,x)}$(*x*,*v*,*y*)

32. Imp(x,y) est l'implication de y par x :

Imp(x,y) $\cong$ Dis(Neg(x),y)

Con(x,y) est la conjonction de x et y :

Con(x,y) $\cong$ Neg(Dis(Neg(x),Neg(y)))

Eq(x,y) est l'équivalence entre x et y :

Con(Imp(x,y),Imp(y,x))

Exist(v,x) est la quantification existentielle de x par v :

Neg(Gen(v,Neg(x)))

33. Th(n,x) est la n[ième] élévation de type de x :

Th(n,x) $\equiv \varepsilon$y(y $\leq$ x$^{(x^n)}$ $\wedge \forall$k(k $\leq$ l(x) $\rightarrow$ (Terme(k,x) $\leq$ 13 $\wedge$ Terme(k,x)

$=$ Terme(k,y)) $\vee$ (Terme(k,x) > 13 $\wedge$ *Terme*(*k*, *y*)

$=$ Terme(k,x.Pr(1,Terme(k,x))^n))))

34. x est un des axiomes de PEANO ou axiomes du groupe 1,Ax| :

Ax|(x) $\equiv$ (x = n$_1 \vee$ x = n$_2 \vee$ x = n$_3$ où des axiomes n$_1$, n$_2$ et n$_3$ sont les ndg I.1, I.2. et I.3respectivement.

35. x est une formule résultant du schéma d'axiome II.1 par la substitution, Ax// 1(x) :

$Ax//1(x) \equiv \exists y(y \leq x \wedge Form(y) = \mathrm{Im}\, p(Dis(y, y), y))$

De façon similaire, on obtient pour les axiomes II.2, II.3 et II.4 les prédicats suivantsAx//2(x), Ax//3(x) et Ax//4(x), respectivement.

36. x est un des axiomes de la logique propositionnelle ou axiomes du groupe 2, Ax//(x) :

$Ax//(x) \equiv (Ax//1(x) \vee Ax//2(x) \vee Ax//3(x) \vee Ax//4(x)$

37. z ne contient pas de variables liées en y à l'endroit où v est libre, Q(z,y,v) :

$Q(z, y, v) \equiv \neg\exists n\exists m\exists w(n \leq l(y) \wedge m \leq l(z) \wedge w \leq z \wedge w$

$= Terme(m,z) \wedge Liée(w,n,y) \wedge Libre(v,,,y))$

38. x est une formule résultant de l'axiome III.1 par substitution, AxIII1(x) :

$AxIII1(x) \equiv \neg\exists v\exists y\exists z\exists n(v, y, z, n \leq x \wedge Var(n; v) \wedge \mathrm{Type}_n(z) \wedge \mathrm{Form}(y) \wedge Q(z, y, v) \wedge x = \mathrm{Imp}(\mathrm{Gen}(v,y),\mathrm{Sub}(y,v,z)))$

39. x est une formule résultant du schéma d'axiomes III.2. par substitution, AxIII2(x) :

$AxIII2(x) \equiv \neg\exists v\exists q\exists p(v,q,p \leq x \wedge \mathrm{Var}(v) \wedge \mathrm{Form}(p) \wedge \neg\mathrm{Libre}(q) \wedge x$

$= \mathrm{Imp}(\mathrm{Gén}(v,\mathrm{Dis}(p,q)),\mathrm{Dis}(p,\mathrm{Gén}(v,q))))$

40. x est une formule résultant de l'axiome de compréhension par substitution, AxIV1(x) :

$AxIV1(x) \equiv \exists u\exists v\exists y\exists n(u, v, y, n \leq x \wedge Var(n; v) \wedge Var(n+1, u) \wedge \neg Libre(u, y) \wedge \mathrm{Form}(y) \wedge x = \mathrm{Existe}(u,\mathrm{Gén}(v,\mathrm{éq}(R(u) * E(R(v)),y))))$

41. x est une formule résultant de l'axiome d'extensionnabilité par substitution, AxIV2(x) :

$AxIV2(x) \equiv \exists n(n \leq x \wedge x = Th(n, n_4)),$

Où n_4 est le ndg de l'axiome d'extensionnabilité.

42. x est un axiome, Ax(x) :

$Ax(x) \equiv (Ax/(x) \vee Ax//(x) \vee Ax///(x) \vee Ax/V1(x) \vee Ax/V2(x))$

43. x est une conséquence immédiate de y et z, Cons(x,y,z) :

$\mathrm{Cons}(x,y,z) \equiv (y = \mathrm{Imp}(z,x) \vee \exists v(v \leq x \wedge \mathrm{Var}(v) \wedge x = \mathrm{Gen}(v,y)))$

44. x est une figure de preuve, Dm(x) :

$Dem(x) \equiv (l(x) > 0 \wedge \forall n(0 < n \leq l(x) \rightarrow (Ax(\mathrm{Terme}(n,x)) \vee \exists p\exists q(0 < p.q < n \wedge$

$\mathrm{Cons}(\mathrm{terme}(n,x),\mathrm{Terme}(p,x),\mathrm{Terme}(q,x))))))$

45. x est une preuve de y,Dem(x,y) :

$Dem(x,y) \equiv \mathrm{Dm}(x) \wedge y = \mathrm{Terme}(l(x),x))$

46. x est une formule prouvable, Dem(x) :

$Dem(x) \equiv \exists y \mathrm{Dem}(y,x)$

De ces formules, il nous faut déduire que (1) à l'exception de la formule 46, qui est non récursive, toutes les définitions, soit qu'elles sont les prédicats (1, 2, 11, 12, 18-26 et 34-46), soit qu'elles sont les fonctions(3-10, 13-17 et 27-33) sont toutes récursives, (2) ne sont pas construites dans le système SFG, étant donné qu'une extension dans le système SFG, n'inclut pas les notions de récursivité, (3) les fonctions ou les prédicats (6-45) sont toutes les traductions d'opérations ou prédicats métathéoriques, c'est-à-dire elles sont des fonctions ou des prédicats entre les nombres de GÖDEL.

1.2.3 ARITHMÉTIQUE RÉCURSIVE

C'est une partie importante de l'arithmétique dans laquelle sont utilisés des procédés constructifs tels que la récursion et l'induction. En clair, l'arithmétique récursive est la partie de la théorie des nombres entiers qui est obtenue au moyen des opérations logiques élémentaires, exceptées les opérations de généralisation, de la relation d'égalité (avec les axiomes qui la caractérisent), du principe d'induction et de définitions récursives (définitions de fonctions au moyen du schéma de récursion primitive) ; il s'agit en particulier de plusieurs théories arithmétiques qui relèvent de l'arithmétique récursive, la théorie de la récursion, la théorie du plus grand commun diviseur à savoir , la théorie de la décomposition en facteurs premiers (avec la théorie sur l'unicité de la décomposition).

Si dans un système approprié, on formalise l'arithmétique récursive, on peut représenter le calcul des valeurs d'une fonction récursive primitive quelconque par des déductions réalisées dans ce système formel.La méthode de GÖDEL, ou l'arithmétisation, utilise l'arithmétique récursive. C'est ce qui lui donne un caractère entièrement constructif.

L'arithmétique récursive est, selon HILBERT, une arithmétique sans variables libres (sa logique sous-jacente est le calcul élémentaire avec des axiomes libres de l'égalité) qui comporte un schéma d'induction ayant des variables libres seulement et un schéma de définition par récursion. Les moyens mis en œuvre dans ce système formel embrassent la classe des fonctions récursives primitives. L'arithmétique récursive a un sens finitiste ; car toutes ses formules peuvent être vérifiées, cela signifie qu'elles peuvent être interprétées par des moyens finistes[3].

1.2.4 PROCÉDURE RÉCURSIVE

1.2.4.1 NOTION DE LA PROCÉDURE

On sait, en effet, que de nombreux algorithmes sont décrits de manière plus descriptive en termes de récursivité. Prenons en compte une procédure P comprenant une instruction Call (appel) sur elle – même, soit une instruction Call sur une autre procédure, qui d'ailleurs, peut entraîner une instruction Call de retour sur la procédure initiale P ; suivant ces conditions, la procédure P constitue de ce fait une procédure récursive. Ainsi, pour empêcher le programme de tourner indéfiniment en rond, une procédure récursive est sensée avoir les deux propriétés ci –après :

1. Il faut qu'il y ait des critères appelés critères fondamentaux pour lesquels la procédure ne s'appelle pas elle – même.
2. Chaque fois que la procédure s'appelle elle – même, elle doit être plus proche de ses critères fondamentaux.

Une procédure récursive est dite bien définie, si elle possède ces deux critères de base. En effet, une fonction se définit de manière récursive, si sa définition fait référence à elle – même. Pour que cette fonction ne soit pas circulaire, elle doit posséder les deux propriétés ci – après :

1. Il faut qu'il ait les valeurs de base ou arguments pour lesquels la fonction ne se réfère pas à elle – même.
2. Chaque fois que la fonction se réfère à elle – même, son argument doit être plus proche d'une valeur de base.

1.2.4.2 ILLUSTRATION DE LA PROCÉDURE

Calcul de n !

Pour tout entier positif $n, on\ a\ n! = n(n-1)!$

En effet, cette fonction factorielle se définit d'une manière récursive.

1. Si $n = 0, alors\ n! = 1$
2. Si $n > 0, alors\ n! = n(n-1)!$

La définition ci – dessus est récursive puis qu'elle se réfère à elle – même lorsqu'elle applique $(n-1)!$

[3] L'arithmétique primitivement récursive est non seulement un fragment de l'arithmétique de PEANO, mais un fragment de l'arithmétique formelle intuitionniste, obtenue par ajout des axiomes de PEANO à la logique intuitionniste de premier ordre, une logique plus restreinte que la logique classique.

1. On voit que la valeur de $n!$ est de façon explicite donnée par $n = 0$, d'où 0 est une valeur de base.
2. On voit également que la valeur de $n!$ Pour n arbitraire se définit en fonction d'un plus petit n plus voisin de la valeur de base 0. Par conséquent, la définition n'est pas circulaire. Cela veut dire donc que la procédure récursive est bien définie.

Calcul de 3 ! Partant de la définition récursive, on a :

(1) $3! = 3.2!$

(2) $2! = 2.1!$

(3) $1! = 1.0!$

(4) $0! = 1$(valeur de base de la définition récursive)

(5) $1! = 1.1 = 1$

(6) $2! = 2.1 = 2$

(7) $3! = 3.2! = 6$

En clair, cela montre que :

Etape (1) : définit 3 ! en fonction de 2 !

Etape (4) : évalue 0 ! explicité car 0 étant la valeur de base de la définition récursive

Etape (5) et (7) : repartent en arrière, en utilisant 0 ! pour évaluer 1 ! servant à évaluer 2 ! et ainsi de suite jusqu'à 3 !

1.2.4.3 PROCÉDURES DE CALCUL

1° Première procédure de calcul de la fonction factorielle n !

FACTO (FAC, N) calcul n ! et affecte le résultat à FAC

1. $Si\ N = 0, alors\ poser\ FAC = 1\ et\ return$
2. $Poser\ FAC = 1\ [initialiser\ FAC\ pour\ la\ boucle]$
3. $Repeat\ pour\ K = 1$ à N

$[fin\ de\ la\ boucle]$

4. $Exit$

2° Deuxième procédure du calcul de la fonction factorielle n !

FACTO (FAC, N) calcul n ! et affecte le résultat à FAC

1. $Si\ N = 0, alors\ poser\ FAC = 1\ et\ Return$
2. $Call\ FACTO\ (FAC, N - 1)$
3. $Poser\ FAC = N^{*}FAC$
4. $Exit$

Il nous faut souligner que la procédure 1 évalue n ! à l'aide d'un processus de boucle itérative. Par contre, la procédure 2 est plutôt récursive, puisqu'elle contient un appel à elle – même. Il y a des langages de programmation, en particulier, qui n'autorisent pas de sous – programmes récursifs de ce genre.

1.2.4.4 SUITE DE FIBONACEE

Par la suite de FIBONACEE, on entend une suite ayant la forme :

$0,1,1,2,3,5,8,13,21,34,55, \ldots notée\ en\ général\ F_0, F_1, F_2, \ldots soit\ F_0 = 0\ et\ F_1 = 1$ les deux premiers termes.

Et on voit que chaque terme suivant est la somme des deux termes précédant :

$8 + 13 = 21$

$13 + 21 = 34$

$21 + 34 = 55$

$34 + 55 = 89$

$55 + 89 = 144$

La suite de FIBONACEE se définit comme suit :

$1° \; Si \; n = 0 \; ou \; n = 1, alors \; F_n = n$

$2° \; Si \; n > 0, alors \; F_n = F_{n-2} + F_{n-1}$

En clair, c'est une définition récursive, puisqu'elle se réfère à elle – même lorsqu'elle utilise$F_{n-2} + F_{n-1}$.

1° On voit que les valeurs de base sont 0 et 1

2° On voit également que la valeur de F_n se définit en fonction des valeurs plus petites de n étant plus proches des valeurs de base.

D'où, il y a lieu d'affirmer que la fonction est bien définie.

Il est à remarquer que la procédure suivante calcule F_n et affecte sa valeur au premier FI.

$FIBO \; (FI, N)$

1. $Si \; N = 0 \; ou \; N = 1 \; alors \; poser \; FI = N, et \; return$
2. $Call \; FIBO \; (FIA, N - 2)$
3. $Call \; FIBO \; (FIB, N - 1)$
4. $Poser \; FI = FIA + FIB$
5. $Exit$

Ici, c'est une procédure récursive ; car elle contient un appel à elle –même. Elle comporte en effet deux appels à elle – même. Cela montre que la procédure suivante calcule les N premiers nombres de FIBONACEE et les affecte à un tableau F.

1. $Poser \; F[1] = 1 \; et \; F[2] = 1$
2. $Repeat \; pour \; K = 3$ à N

$Poser \; F[K] = F[K - 1] + F[K - 2]$

$[fin \; de \; la \; boucle]$

3. $Exit$

Il s'agit ici d'une procédure itérative qui est beaucoup plus efficace que la procédure récursive reprise ci – haut.

1.2.4.5 SCHÉMA ITÉRATIF – SCHÉMA RÉCURSIF

Comparativement aux algorithmes itératifs, les algorithmes récursifs se présentent sous une forme simple. Le programmeur peut éprouver des difficultés pour les concevoir parce qu'ils nécessitent un grand effort d'abstraction. On dit qu'un problème se prête bien à une démarche ou procédure récursive, quand il peut être décomposé en une suite d'actions dont l'une constitue un sous-problème semblable à celui d'origine mais qui est appliqué à un environnement réduit.

Aussi faut-il préciser que la récursivité est coûteuse, mais, en la comparant avec l'itération, elle donne des solutions s'exprimant de manière plus concise que les solutions itératives correspondantes. Néanmoins, si la solution itérative est simple à formuler, il est donc déconseillé d'utiliser une solution récursive en raison du coût que cela implique eu égard à l'exécution de la gestion des environnements successifs.

Par ailleurs, il y a des problèmes auxquels il est difficile de donner une solution, qui ne soit pas récursive. Il s'agit particulièrement des problèmes dont la recherche de la solution se fait par essais successifs. Cela se passe comme dans le parcours d'un labyrinthe, c'est-à-dire on essaie un chemin et, si cela n'amène pas à la solution, il faut revenir en arrière et essayer un nouveau chemin. Pour s'en sortir, il existe des méthodes de traduction d'une procédure récursive en une procédure non récursive.

1.2.4.6 TRADUCTION D'UNE PROCÉDURE RÉCURSIVE EN UNE PROCÉDURE NON RÉCURSIVE

Considérons P une procédure récursive telle qu'un appel à P ne peut provenir que de P. Ainsi, on peut traduire P en une procédure non récursive comme suit :

1. On considère une pile STY pour chaque paramètre Y ;
2. On considère également une pile STX pour chaque variable locale ;
3. On considère finalement une variable locale AD et une pile STAD pour ranger les adresses de retour.

Le constat est que chaque fois qu'apparaîtra un appel récursif à P, les valeurs actuelles des paramètres et des variables locales vont être empilées sur les piles correspondantes et cela pour traitements ultérieurs et à chaque retour récursif sur P. De même, les valeurs des paramètres et des variables locales liées à l'exécution actuelle de P vont être restaurées après défilements.

La gestion de l'adresse de retour suivra la démarche que voici :

A ce stade, on suppose que la procédure P contient une instruction récursive Call P à l'étape K. on a deux adresses de retour associées à cette étape K.

1. L'adresse actuelle de retour de la procédure P qui sera utilisée lorsque l'exécution courante de la procédure P sera achevée.
2. La nouvelle adresse de retour K+1, qui est l'adresse de l'étape suivant Call P et qui sera utilisée aux fins de revenir au stade d'exécution actuelle de la procédure.

De cette traduction de différentes procédures, il est à remarquer que pour certains acteurs, il y a lieu d'empiler la première de ces deux adresses, c'est-à-dire l'adresse de retour actuelle, sur la pile d'adresse de retour STAD, et, pour d'autres acteurs, on peut empiler la seconde adresse, la nouvelle adresse de retour K+1, sur STAD. Donc, la deuxième démarche a l'avantage de simplifier le processus de traduction de P en une procédure non récursive.

Elle veut dire donc qu'une pile STAF vide va indiquer un retour au programme principal appelant initialement la procédure récursive P.

2 L'ARITHMÉTISATION DE LA MÉTAMATHÉMATIQUE

2.1 L'IDÉE D'ARITHMÉTISATION

Soit K une théorie du premier ordre. Le leitmotiv de l'idée d'arithmétisation chez GÖDEL consiste à encoder les énoncés métamathématiques à propos de la théorie K de manière à pouvoir les substituer par des formules arithmétiques. A leur tour, ces formules pourront même être exprimées dans la théorie K. Donc, on peut exprimer les propriétés métamathématiques de la théorie K dans K elle-même. L'arithmétisation se définit de la façon suivante [3]:

2.2 DÉFINITION

On appelle arithmétisation d'une théorie K, une fonction monomophique g de l'ensemble des symboles, expressions et suites fines d'expressions de K vers l'ensemble des entiers positifs vérifiant les conditions suivantes :

(1) g est effectivement calculable ;

(2) il ya une procédure effective qui détermine si un entier positif m est dans l'image de la fonction g et le cas échéant, cette procédure permet de trouver l'objet x' tel que $g(x) \equiv m$.

Au regard du premier théorème d'incomplétude, l'idée d'arithmétisation a pour aboutissement son application à la théorie arithmétique formelle S et au langage L_A. D'une manière plus générale, GÖDEL procède en deux étapes. La première définit une fonction d'arithmétisation en associant à tout symbole du langage de K, un nombre entier positif. La deuxième étape traduit les énoncés du métalangage de K en des formules arithmétiques.

2.3 ARITHMÉTISATION DE LA SYNTAXE [4]

La preuve d'incomplétude de GÖDEL fait appel à l'arithmétisation de la syntaxe. En effet, on considère les expressions d'un langage formel (les symboles, séquences de symboles, etc.) comme étant des objets formant une classe à dénombrer. C'est donc une classe susceptible d'être corrélée effectivement et injectivement à l'ensemble des entiers naturels.

Au niveau métathéorique, on peut se référer aux indices de l'énumération plutôt qu'aux objets énumérés et de représenter dans ce cas les propriétés et relations syntaxiques des systèmes formels par les propriétés des nombres naturels [4].

Ainsi, dans les systèmes axiomatiques, il est possible d'énumérer tous les symboles primitifs, tous les termes et toutes les preuves telles que pour un entier positif n, il y a lieu de trouver en un nombre fini d'étapes, quelle combinaison de signes est le m^{ième} énoncé et quelle combinaison de signes est la n^{ième} preuve. D'où, pour chaque système, une méthode permet, étant donnés deux entiers positifs m et n, de décider en un nombre fini d'étapes si la m^{ième} preuve est une preuve du n^{ième} énoncé. Quelle en soit la preuve enregistrée, c'est possible de préciser de quel énoncé elle est la preuve. Ce qui signifie que la relation binaire « être une preuve de » est récursive ou axiomatiquement énumérable [4].

La preuve classique sur l'incomplétude de l'arithmétique de PEANO amène GÖDEL à supposer que la consistance ω de l'arithmétique est définie sur l'ensemble infini des nombres naturels. Il formule au début de deuxième ordre du postulat chez PEANO sur l'existence d'un tel ensemble infini.

$$\exists_x\{\emptyset \in \neg\forall y(y \in x \longrightarrow Sy \in x)\} \longrightarrow \forall y \in x$$

La théorie ou le système formel S_2 de l'arithmétique de PEANO est cependant finie au sens de la théorie de la démonstration de HILBERT aux fins de représenter le système formel dans l'arithmétique de PEANO. On voit GÖDEL assigner des nombres naturels (les nombres de GÖDEL) aux signes primitifs S_2 et aux suites finies de tels signes (les formules deS_2). Ainsi on a par exemple :

$$\text{"0"} \ldots 1 \text{ " "}\ldots 5 \text{ "}\pi\text{"} \ldots 9$$
$$\text{"}f\text{"} \ldots 3 \text{ " } \vee \text{ "}\ldots 7\text{"(" } \ldots 11$$
$$\text{"(" } \ldots 13$$

Notons qu'on établit la bijection entre une suite de nombres naturels et une suite finie de signes de $L(S_2)$, *le langage du système formel s'accompagne d'une application :* $\emptyset: \mathbb{N}_s \longrightarrow \mathbb{N}$ des suites finies de nombres naturels sur des nombres naturels associant par bijection, cette fois à la suite $n_1, n_2 \ldots n_k$, le nombre $2^{n1}.3^{n2} \ldots p_k^{nk}$ où p_k dénote le K^{ième} nombre premier. GÖDEL veut ainsi plonger le système formel dans l'arithmétique de PEANO.

Il s'agit de l'arithmétique des fonctions récursives primitives construites par les règles de substitution et de composition (ou récursion) à partir des fonctions de base de successeur, des fonctions constantes dont la fonction zéro.

$$\forall x(Zx = 0)$$

et les fonctions d'identité. Si l'on en a plus, comme chez GÖDEL, la fonction μ pour « le plus petit nombre K tel que »

$$\emptyset(K+1, x_2, \ldots, x_n) = \mu(k, \emptyset(k, x_2, \ldots, x_n), x_2, \ldots, x_n)$$

On a les fonctions récursives générales. Mais si les fonctions récursives, comme on le sait, sont des fonctions arithmétiques, leur domaine et codomaine sont l'ensemble dénombrable $\mathcal{N}_0$ des nombres naturels. Elles ne se prêtent pas naturellement à un calcul fini.

D'après GÖDEL, la consistance ω est définie pour les propriétés $F(x)$ des nombres naturels. En effet, elle suppose le parcours complet des valeurs d'une fonction définie sur l'ensemble des nombres naturels. Alors que la consistance simple de ROSSER, pour l'arithmétique de PEANO, ne fait que substituer à la consistance ω le concept d'énumérabilité récursive qui lui est équivalent par le recours récursif $\mathcal{N}_0 = card\ N$.

Il est à remarquer que le produit de convolution polynomiale ou la diagonale de CAUCHY n'est pas source de paradoxe de l'autoréférence à travers la diagonale de CANTOR dans l'arithmétique de PEANO. En effet, GÖDEL accepte qu'il s'agit là d'un accident, si une formule obtenue par la substitution diagonale se transforme en un énoncé indécidable qui dit de lui–même qu'il est non prouvable : l'autoréférence ne serait qu'un produit dérivé de la procédure diagonale. GÖDEL suppose en effet la consistance $\omega -$ la consistance de ROSSER qui utilise la notion équivalente de fonction récursive sur $\mathcal{N}_0 = Card\ N$.Cequi permet, à la diagonalisation à la CANTOR, d'identifier certains énoncés qui portent sur le contenu arithmétique de la théorie

(la théorie – objet) avec les énoncés métalogiques de la théorie. Cela se passe dans le cas de l'énoncé indécidable du système formel S_2 de l'arithmétique de PEANO.

$$\forall_{\underline{x2}} \neg P_{\underline{r}}(\overline{k}, \underline{x_2})$$

où les expressions soulignées font partie du système formel et où $\overline{k}$ est $le\ ndg$ de la formule

$$\forall_{\underline{x2}} \neg P_{\underline{r}}(\underline{x_1}, \underline{x_2})$$

En effet, l'énoncé indécidable dit qu'il n'ya pas de preuve de lui-même dans S_2, mais l'énoncé est vrai dans T_2 (le modèle de l'arithmétique de PEANO). Le jeu de la substitution montre qu'on peut l'admettre, dans une logique interne de l'arithmétique ou une logique polynomiale, étant donné que la substitution par diagonalisation (de CAUCHY) indiquerait

$$\psi: \forall x Ax[n] = \prod_{n} \prod (a_0 x, b_0 x)$$

Il se fait que pour un énoncé A, on peut associer à cela son $ndg[n]$; ce qui donne par la suite une représentation polynomiale qui fera apparaitre cet n de la manière suivante :

$$C_n = a_0, b_n + a_1, b_{n-1} + a_n, b_0$$

où on a $\overline{C}_n 1 - C_n$,

Mais il fera entrer cela dans une computation sans paradoxe ; raison pour laquelle le $\overline{k}$ de l'arithmétique de PEANO, pointe réellement à un ensemble des nombres naturels se plaçant sur la codiagonale ou anticodiagonale des énoncés niés non récupérable sans paradoxes de l'autoréférence. Ainsi, on aura remarqué que la consistance externe dans l'arithmétique ensembliste N_0 est source de divers paradoxes rencontrés en logique mathématique.

Par contre, la consistance interne de l'arithmétique en général ou polynomiale le réalise et cela en s'évanouissant dans le calcul.

Fort de tous ces éléments, il est à retenir que dans sa preuve de complétude pour l'arithmétique de PEANO , GÖDEL avoue que son second résultat d'incomplétude, particulièrement les preuves sur la consistance, ne contredit pas les points de vue formalistes jusqu'alors soutenus par HILBERT ; étant donné qu'on peut concevoir qu'une démonstration finitiste « interne » ne peut pas être formulée dans le langage des systèmes formels de l'arithmétique de PEANO ou dans le langage des systèmes plus englobant telle que la théorie des ensembles ou l'analyse classique.

1) Les symboles primitifs

On note qu'à chaque symbole μ du langage K correspond un entier positif impair. On voit clairement que les nombres $3,5,7,9,11\ et\ 13$ sont alors impairs. Quant aux nombres $13 + 8K, 7 + 8K, 1 + 8(2^3 + 3^k)\ et\ 3 + 8(2^3 + 3^k)$, ils le seront si et seulement si $8k\ et\ 8(2^3 + 3^k)$ le sont eux – mêmes. En clair, la somme des deux entiers positifs est impaire lorsque ceux – ci sont de parité différente. Et pourtant, $8k + 8(2^3 + 3^k)$ étant divisibles par 2, sont tous deux pairs.

En outre, la fonction $\mathcal{g}$ est monomorphique. Cela signifie qu'elle associe les nombres de GÖDEL $\mathcal{g}(\mu)$ différents à des symboles μ différents et cela à 3 niveaux.

D'abord, on constate que $3,5,7,9,11\ et\ 13$ sont tous différents l'un de l'autre.Ensuite, on aperçoit que $13 + 8K, 7 + 8K, 1 + 8(2^3 + 3^k)\ et\ 3 + 8(2^3 + 3^k)$ sont différents de $3,5,7,9,11\ et\ 13\ car\ k, n \geq 1$.Enfin, il est vrai que $13 + 8K, 7 + 8K, 1 + 8(2^3 + 3^k) et\ 3 + 8(2^3 + 3^k) e$ sont aussi tous différents l'un de l'autre, car ils sont différents relativement à la division mod 8.

2) Les expressions

A ce niveau, notons d'abord que la fonction $\mathcal{g}$ fait correspondre à toute expression $u_0 u_1 \ldots\ldots u_r$ constituée des symboles $u_0 u_1 \ldots\ldots u_r$ un nombre entier positif $\mathcal{g}(u_0 u_1 \ldots\ldots u_r)$, le nombre de GÖDEL est formé, de la manière suivante : $\mathcal{g}(u_0 u_1 \ldots\ldots u_r) = 2^{\mathcal{g}(u_0)} 3^{\mathcal{g}(u_1)} \ldots p_r^{\mathcal{g}(u_r)}$ où p_j est le j^{e}nombre premier avec $p_0 = 2$.

Suivant le procédé d'arithmétisation au niveau des expressions, la fonction $\mathcal{g}$ est monomorphique relativement aux expressions de la théorie K. Selon le constat, à des expressions différentes correspondent des nombres de GÖDEL différents. Ce constat se justifie par le théorème fondamental de l'arithmétique qui stipule, dans ce cas, l'unicité de la factorisation des entiers en puissances de nombres premiers.

En outre, les symboles et les expressions de la théorie K ont des nombres de GÖDEL différents. A l'opposé des symboles dont les nombres de GÖDEL sont impairs, les expressions ont par contre les nombres de GÖDEL pairs. On voit que $2^{g(u_0)}$ étant pair, car il est divisible par 2, le nombre de GÖDEL $2^{g(u_0)}3^{g(u_1)} \dots \dots p_r^{g(u_r)}$ associé à une expression $u_0u_1 \dots u_r$ est donc pair.

De ce qui précède, il y a lieu de constater que la fonction g permet de rendre discriminatoire entre l'utilisation d'un symbole en tant que tel ou en tant qu'expression. En clair, une expression peut être constituée uniquement d'un symbole u. Alors que, la parité de ce symbole varie selon qu'il soit utilisé en tant qu'expression.

3) Les suites finies des expressions

En créant un modèle arithmétique de PM équivalent à PM ayant des symboles, GÖDEL établit qu'à toute suite finie d'expressions $e_0, e_1, \dots, e_r$de K, correspond un nombre entier positif $g(e_0, e_1, \dots, e_r)$; le nombre de GÖDEL de cette suite finie d'expressions est définie comme suit :

$g(e_0, e_1, \dots, e_r) = 2^{g(e_0)}3^{g(e_1)} \dots \dots p_r^{g(e_r)}$ où p désigne le j^{e} nombre premier.

Par rapport aux suites d'expressions, la fonction g est monomorphique. On a des suites d'expressions différentes ayant des nombres de GÖDEL différents, et cela se justifie toujours par le fondamental de l'arithmétique.

En plus, on se rend compte que le nombre de GÖDEL, d'une suite finie d'expressions, est différent des nombres de GÖDEL associés à des symboles ou expressions. Cette différence résulte du fait que tout symbole u a un nombre de GÖDEL impair, mais le nombre de GÖDEL associé à une suite finie, d'expression $e_0, e_1, \dots, e_r$ est nécessairement pair en raison du facteur $2^{g(e_0)}$. Aussi se comprend-t-elle que les nombres de GÖDEL, associés à une expression $u_0, u_1, \dots, u_r$ et à une suite finie d'expression $e_0, e_1, \dots, e_r$ sont tous deux pairs en raison de leur premier facteur qui est une puissance de 2. On trouve respectivement$2^{g(u_0)}$ et $2^{g(e_0)}$. S'agissant d'une expression, cet exposant correspond toutefois au nombre de GÖDEL d'un symbole. Donc, il est impair, contrairement à une suite d'expressions, cet exposant est pair, car il correspond au nombre de GÖDEL d'une expression.

En clair, on voit que la fonction permet à ce niveau d'établir une distinction entre un symbole utilisé en tant que symbole, et une expression ou suite finie d'expressions.

2.3.1 LE CODAGE

Le codage est à juste titre la numérisation, l'idée montrant un lien de cause à effet. Deux mécanismes sont opérés lors du codage à savoir : le formalisme et la numérisation [5].

2.3.1.1 LE FORMALISME

Toutes les opérations d'un système formel, qui sont la création d'un langage (termes et énoncés), les axiomes, les règles, leur enchaînement pour fournir des preuves (les théorèmes), toutes les opérations en bureautique, à savoir le renommage de variables et la substitution de termes pour des variables, peuvent être écrites en informatique. Cela suppose les caractéristiques de l'ordinateur illimitées c'est-à-dire n'ayant pas de limitation de mémoire. Ce travail est semblable à celui d'un logiciel de traitement de texte.

Remarquer que c'est une activité à caractère formel, bureautique : pas la moindre erreur, par exemple en confondant ''o'' et ''O'' [5].

2.3.1.2 LA NUMÉRISATION

Le langage se code par des nombres, en binaire ou hexadécimal qu'importe ; même les images et les sons sont codables. Le travail de GÖDEL en 1931, a consisté pour la première fois, à associer à toute expression du langage un « nombre de GÖDEL » $\ulcorner\alpha\urcorner$, qui est certes un codage. D'après GÖDEL$\ulcorner(\urcorner = 11\ulcorner)\urcorner = 13$, alors que les codes modernes ASCII des parenthèses sont respectivement 40, 41.

Le codage utilisé par GÖDEL est quelque peu obsolète, en ce sens qu'il ne se justifie qu'en l'absence de problème de taille (encombrement mémoire). Mais, outre cela, le codage de GÖDEL opère suivant les mêmes procédés que les codages modernes [5].

On voit également que les propriétés d'un système formel seront donc traduites à une fonction ou une opération des codes associés représentables dans l'arithmétique formelle, comme aussi tout système dans lequel il est possible de traduire l'arithmétique [5].

Particulièrement, c'est donc l'aspect réflexif du théorème qui indique qu'un système d'arithmétique peut se représenter lui-même, « parler » de soi, de même qu'un langage de programme ou un programme particulier de ce même langage [5].

2.3.1.3 DU CODAGE AUX THÉORÈMES D'INCOMPLÉTUDE

En 1931, GÖDEL prouve l'incomplétude de tout système axiomatique contenant la théorie des nombres. Il s'agit là de l'arithmétique théorisée. Depuis lors, on établit une séparation nette entre vérité et prouvabilité.

La métamathématique gödelienne va du codage aux théorèmes d'incomplétude prouvant ainsi la faillite du réductionnisme à fonder les mathématiques sur la seule logique formaliste.

2.3.1.4 DE L'ARITHMÉTISATION (LE CODAGE) DE LA THÉORIE FORMALISÉE

Pour GÖDEL, un système formalisé est capable d'exprimer tous les concepts arithmétiques et d'établir toutes les relations arithmétiques courantes. Par après, il est possible d'arithmétiser ce calcul. Cela veut dire que les formules et les preuves de son système peuvent être codées en nombres entiers. Donc, elles sont arithmétisées, selon une méthode permettant d'établir une correspondance bi-univoque entre les expressions du système et un certain sous ensemble de $\mathbb{N}$.

Le codage dans la métamathématique de GÖDEL établit une correspondance bi-univoque entre les expressions du système et un sous-ensemble de $\mathbb{N}$. La description technique du codage s'effectue de telle manière que GÖDEL assigne un nombre unique (*nombre de GÖDEL*) à chaque signe élémentaire du système :

Constances : → nombre entier de 1 à 10,

variables numérique x, y → nombre premier supérieur à 10,

variables propositionnelles p, q, →carré d'un tel nombre

variables de prédicats P, Q →cube d'un tel nombre

formule ou suite de signes produit des nombres premiers élevés à une puissance dont l'exposant est le nombre de GÖDEL de chaque signe.

Démonstration ou suite finie des formules → produit des nombres premiers élevés à une puissance dont l'exposant est le nombre de GÖDEL de chaque signe.

Ainsi il y'a lieu de déterminer le nombre de GÖDEL de toute expression et, dès qu'un nombre est donné, il est possible de déterminer s'il s'agit d'un nombre de GÖDEL et de retrouver l'expression qu'il représente.

Prenons l'exemple : le nombre 100.

$100 > 10$, donc 100 n'est pas le nombre de GÖDEL d'une constante

100 n'est pas un nombre premier supérieur à 10 et 100 n'est ni le carré ni le cube d'un nombre premier supérieur à 10, donc 100 n'est pas le nombre de GÖDEL d'une variable.

$100 = 2^2.5^2$ On voit sauter le nombre 3. C'est pour dire que les nombres premiers ne sont pas successifs. Cela ne correspond pas à la règle posée.

De même qu'il est possible à ce stade de mentionner que toutes les assertions métamathématiques portant sur la théorie formalisée s'érigent en assertions métamathématiques portant sur des nombres entiers, il en résulte que ces relations sont transformées, quant à elles aussi, en relation portant sur les nombres. Donc les assertions métamathématique se trouvent transformées en relations arithmétiques (opérations).

A titre d'exemple

(p v p) → p
8 11^2 2 11^2 9 3 11^2

Ce qui donne en effet le nombre de GÖDEL$2^8.3^{11^2}.5^2.7^{11^2}.11^9.13^3.17^{11^2}$

Ainsi, la proposition "$(p \vee p)$ est une partie de $(p \vee p) \longrightarrow p''$ devient" b est un facteur de a", en désignant par b, le nombre de GÖDEL de $(p \vee p)$, soit $2^8.3^{11^2}.5^2.7^{11^2}.11^9$, et par a le nombre de GÖDEL ci-haut.

Le fondement théorique du codage dans la théorie de démonstration de GÖDEL trouve sa justification dans le fait que ces métamathématiques se trouvent entièrement arithmétisées. Donc, il est possible d'entreprendre l'étude de questions métamathématiques par une analyse des propriétés arithmétiques et des relations entre certains entiers. Aussi se permet-on d'affirmer qu'aux assertions métamathématiques vraies correspondent des formules arithmétiques vraies.

Ce codage est un procédé qui permet à GÖDEL de rédiger des assertions arithmétiques qui se réfèrent à elles – mêmes, c'est-à-dire les autoréférences typiques semblables aux paradoxes de RICHARD et/ou au paradoxe du menteur. L'assertion que construit GÖDEL pour prouver un énoncé à la fois vrai et non prouvable est de ce type. Elle utilise le fait qu'un nombre du codage peut ou non correspondre à ce qu'exprime le nombre du codage qu'il code (paradoxe de RICHARD).

Ainsi, la formule $(\exists x)(x = s\ y)$ a pour nombre de GÖDEL qu'on peut calculer[4], la formule $(\exists x)(x = s\ m)$ a un nombre de GÖDEL qu'on peut calculer, mais qu'il est aussi possible de se contenter de définir ou de caractériser métamathématiquement, par la manière de sa formation et sans ambigüité : c'est le nombre de GÖDEL de la formule obtenue en substituant m à y (la variable dont le nombre de GÖDEL est 13) dans la formule dont le nombre de GÖDEL est m. Il peut être noté par $Sub\ (m, 13, m)$.

La preuve proprement dite,

Elle se développe comme suit :

GÖDEL construit une formule arithmétique G qui affirme d'elle – même qu'elle n'est pas prouvable. Pour cela, il associe la formule G à un certain nombre h, et la construit de façon qu'elle corresponde à l'assertion : "la formule associée au nombre h n'est pas prouvable" ou "aucune preuve ne peut être donnée de la formule qui porte le nombre de GÖDEL h".

De quelle manière procède GÖDEL ?

Pour ce faire, GÖDEL utilise un fait de système de codage, il existe une relation arithmétique entre le nombre de GÖDEL d'une preuve et le nombre de GÖDEL de sa conclusion.

Etayons ce procédé par un exemple ;

Si m est le nombre de GÖDEL de la formule $(\exists x)(x = s\ y)$,

Si n est le nombre de GÖDEL de la formule $(\exists x)(x = s\ 0)$,

On déduit que la seconde formule découle de la première. La suite des 2 formules est donc une preuve de la seconde. Si l'on associe à la suite le nombre $k = 2^m.3^n$, alors, il ya une relation arithmétique définie entre k, nombre de GÖDEL de la preuve, et n, nombre de GÖDEL de la conclusion. Généralement, ce type de relation se laisse traduire par $D\overline{em}(k, n)$.

En effet, la formule $Dem\ (k, n)$, signifie que l'assertion "telle suite de formules est une preuve de telle conclusion" est vraie.

De même, l'assertion métamathématique :"la suite de formule qui porte le nombre k n'est pas une preuve de la formule qui porte le nombre n", est représentée par une formule comme dans la théorie formalisée et arithmétisée de l'arithmétique.

En généralisant cette formule et en la faisant précéder de $(\forall k)$, on a alors l'assertion métamathématique qui s'énonce comme suit :

---

[4]*Ce nombre de GÖDEL est comme suit :*

$(\ \exists x\ )(\ x\ =\ s\ \ y\ \ )$

8 411 9 8 11 5 7 13 9

$2^8x3^4x5^{11}x7^911^8x13^{11}17^5x19^723^{13}x29^9$

Remarquons que tout entier naturel n'est pas à considérer comme le nombre de GÖDEL d'une formule. Comme on fait qu'une machine de TURING est une entité correspondant à un ensemble de formules, il y a lieu de lui assigner un nombre de GÖDEL.

"Pour tout k, la suite de formule qui porte le nombre de GÖDEL,k n'est pas une preuve de la formule qui porte le nombre de GÖDEL n", veut dire que "la formule qui porte le nombre de GÖDEL n n'est pas prouvable" ou "aucune preuve ne peut être donnée de la formule qui porte le nombre de GÖDEL n".

Pour GÖDEL, *n* constitue un cas particulier de $sub\ (m, 13, m)$, le nombre de GÖDEL de la formule obtenue en substituant m à y (dont le nombre de GÖDEL est 13) dans la formule dont le nombre de GÖDEL est m.

La formule :

$(\forall k)\neg Dem(k, sub(m, 13, m))$ représente l'assertion métamathématique :

"La formule qui porte le nombre de GÖDEL $sub(m, 13, m)$ n'est pas prouvable". Elle peut être formalisée et porter un nombre de GÖDEL *h* qui est calculable.

3 LES NOMBRES DE GÖDEL

Soit SFG une théorie. D'abord on assigne à chaque symbole primitif de SFG un nombre par une fonction g :

(	)	,	¬	→	∀
3	5	7	9	11	13

x_k	a_k	f_k^n	A_k^n
13+8K	7+8K	$1+8(2^n3^k)$	$3+8(2^n3^k)$

Ce qui revient en effet à dire que le procédé d'arithmétisation de GÖDEL procède par association d'un nombre à chaque symbole primitif du langage.

0	‘	¬	∨	∀	(	)
1	3	5	7	9	11	13

Cela vaut de même pour chaque symbole de variable : $si\ \ p_n$ est le nième premier.

x_1	x_2	…	x_n	y_1	y_2	…	y_n	z_1	z_2	…	z_n	…
17	19	…	P_n+6	17^2	19^2	…	P_n^2+6	17^3	19^3	…	P_n^3+6	…

Le nombre associé à une expression $S_1S_2 \ldots S_K$ comprenant les symboles primitifs S_i est $2^{n1}X3^{n2}X5^{n3}X \ldots Xp_k^{nK}$;

où $n_1, n_2, \ldots, n_k$ sont les nombres associés aux symboles $S_1, S_2, \ldots, S_k$. Donc, par exemple ;

Le nombre de $y_1(x_1) \vee \neg\, y_1(x_1)\ est 2^{17^2}3^{11}5^{17}7^{13}11^{7}13^{5}17^{17^2}19^{11}23^{17}29^{13}$

Un formalisme est un système des symboles assujettis à certains principes de manipulation qui ressemble à un objet concret[5] dont il est possible d'examiner complètement la structure. L'étude des propriétés du formalisme exige de recourir à

[5] L'opposition concret-abstrait concerne tantôt l'ontologie c'est-à-dire les objets physiques, perceptibles, sensibles et cela par comparaison aux entités abstraites, tantôt les notions. Ici, il y a chez GÖDEL opposition entre les propriétés des combinaisons de signes envisagés comme

une métalangue qui va comporter des expressions désignant les signes de ce formalisme et les combinaisons des signes du formalisme, douées de signification des prédicats correspondant aux divers types de suites de signes du système formel, ainsi que des relations exprimant les propriétés du formalisme.

L'arithmétisation d'un tel formalisme requiert quelques exigences[6] :

- On coordonne à chaque signe un nombre premier.
- On donne un procédé coordonnant un entier bien déterminé à toute suite de signes ou expressions du formalisme.

Si S_y est une expression du système formel formée d'une suite de k signes et si n_1, n_2….n_n sont des nombres premiers coordonnés à ces signes, alors le nombre coordonné à l'expression S_y est :

$2^{n1} \times 3^{n2} \times \ldots\ldots xp_k^{nk}$;

où p_k est le $k^{ème}$ nombre premier différent de 1

Et si S_p est également une suite de k, expressions du système formel, et si n_1, n_2, ……n_k sont les nombres coordonnés à ces expressions, alors le nombre coordonné à Sp est :

$2^{n1} \times 3^{n2} \ldots . xp_k^{nk}$

Le nombre coordonné à Sp est :

$2^{n1} \times 3^{n2} \times \ldots x\ p_k^{pnk}$

Par le théorème sur l'unicité de la décomposition d'un nombre en ses facteurs premiers, on montre évidemment que la correspondance à établir est biunivoque. A tout élément de système formel (signe, expression ou suite d'expressions) correspond un nombre entier et à un seul nombre entier correspond un élément de système formel au plus [6].

Ainsi, on appelle une telle correspondance correspondance de GÖDEL et l'on appelle le nombre entier, coordonné à un élément E_C du système formel par une correspondance de GÖDEL, le nombre de GÖDEL (ou ndg) ou nombre – G de E_C (ou ndg de E_C).

Précisons en effet [6] que l'opération consistant à passer d'un élément appartenant à un système formel, à son ndg, est une fonction métathéorique. Alors que l'opération consistant à passer du ndg d'un élément à cet élément lui-même, est une fonction inverse de la fonction métathéorique.

Ces deux fonctions sont évidemment les *fonctions de GÖDEL* représentées par les notations abrégées suivantes :

fonction directe : Ngd (E_C) : ndg de l'élément E_C ;

fonction inverse : Ngd^{-1} (a) : élément du système formel dont le ndg est a.

Si les expressions de la suite $E_1, E_2, \ldots, E_k$ ont respectivement les nombres $N_1, N_2, \ldots, N_k$, le nombre associé à la suite est $2^{N1}x3^{N2}x5^{N3}x2^{N1}x \ldots xp_k^{nK}$. Le nombre associé à la suite de (deux) formules $y_1(x_1) \vee \neg\, y_1(x_1) et\ y_1(0') \vee \neg\, y_1(0')$ est

$$2^{2^{17^2}3^{11}5^{17}7^{13}11^7 13^5 17^{17^2}19^{11}23^{17}29^{13}} 3^{2^{17^2}3^{11}5^{17^3}11^{13}13^7 17^5 19^{17^2}23^{11}29^{1}31^{3}37^{13}}$$

Notons que le nombre associé à une expression E d'après cette procédure, est appelé comme tel, le nombre de GÖDEL $(le\ ndg)$ de E, ou en abréviation $\lceil(E)$. Le théorème fondamental de l'arithmétique nous apprend qu'un nombre a une et une seule décomposition en nombres premiers. Il est donc évident de dire que $\lceil$ est une fonction injective. De toute façon, on peut dire que $le\ ndg$ d'une expression est suffisant à la détermination de la suite de symboles primitifs composant E.

Ensuite, on définit une fonction Γ qui attribue un nombre à toute expression $\mu = \mu_0\mu_1 \ldots . \mu_r$ de SFG(les μ_i sont des symboles primitifs) de la manière suivante :

$\mu_0\mu_1 \ldots . \mu_r) \equiv 2g^{(\mu_0}3g^{(\mu_1)} \ldots Pr^{g(\mu_r)}$

---

types et les propriétés abstraites appartenant comme les combinaisons des signes, Chez HILBERT, cette opposition s'applique à des propositions, à des raisonnement où à des inférences plutôt qu'à des entitésVoir : J. LARGEAULT, Logique mathématique, textes, p.258.

où p_r désignele i-ièmenombres premiers.

Ainsi, on peut arithmétiser de nombreuses conceptions métamathématiques telles que:

(1) N(n) est le nombre de GÖDEL de la formule obtenue du nombre de GÖDEL n ayant une seule variable libre en substituant la variable libre au nombre n.

(2) Dem (x,y) si et seulement si x est le nombre de GÖDEL d'une preuve de la formule du nombre de GÖDEL y.

4 CONCLUSION

La métalogique et la métamathématique constituent le champ d'approfondissement et de pertinence de la métathéorie. Partant des axiomes d'un système formel, il y a lieu de déduire des théorèmes faisant partie dudit système. Les métathéories touchant les propriétés syntaxiques ou sémantiques des systèmes formels concernent des formules indécidables dans le cadre de ses systèmes.

REFERENCES

[1] J. Largeault : *Logique mathématique*, *textes*, Pairs : Armand Colin ,1972.

[2] Benis Sinaceur, « Récursivité », Encyclopédie Philosophique Universelle. Les Notions Philosophiques, Tome 2, Paris : PUF, pp.2188-2191,1990.

[3] Elliot Mendelson : *Introduction to Mathematical Logic*, 4ème édition, §3 – 4, Princeton: Van Nostrand,1964.

[4] Jocelyne Couture, *Arithmétisation de la syntaxe,* 2013.
[Online] Available : *http://www.math.uqam.ca/~belair/seminaire%20de%20logique%20/exp.24fev06-MB* (août 2013).

[5] J.-Y. Girard, *Proof Theory and Logical Complexity*, vol.1, Bibliopolis, 1987.

[6] J. Ladrière : *Les limitations internes des formalismes. Etude sur la signification du théorème de Gödel et des théorèmes apparentés dans la théorie des fondements des Mathématiques*, Nawelaerts/Gauthier-Villards, Paris/Louvain, 1957.