Langages formels, Calculabilité, Complexité

Calculabilité distribuée

Yoann Bourse

2009-2010 : Semestre 1

Résumé

Nous étudierons ici un modèle de calcul distribué développé par *Anguin & all* [1] basé sur des interactions entre agents mobiles non différenciés de faibles ressources et donc ne pouvant communiquer que sur des distances très courtes. Des exemples concrets nous permettront d'étudier l'adéquation de ce modèle à la réalité ainsi que les calculs de base qu'il peut effectuer. Une formalisation de ce processus à travers le paradigme de protocole de population nous permettra d'énoncer des propriétés fondamentales de ce modèle : la calculabilité de tout prédicat défini dans l'arithmétique de Presburger, avec la complexité temporelle de $n^2 log(n)$ interactions aléatoires.

Table des matières

1 Applications du modèle **2**
- 1.1 Compteur dans une population animale 2
- 1.2 Utilisations pour l'homme et comparaisons 4
- 1.3 Calculs parallèles et operations booléennes 5

2 Modèle formel des protocoles de population **6**
- 2.1 Protocoles de population 6
- 2.2 Exécution et calcul 7
- 2.3 Schema de fonctionnement 8
- 2.4 Lien avec les machines de Turing 8

3 Calculabilité de l'arithmétique de Presburger **9**
- 3.1 Logique et arithmétique de Presburger 9
- 3.2 Portée calculatoire du modèle 9

4 Complexité pour des rencontres aléatoires **11**
- 4.1 Interactions aléatoires et complexité 11
- 4.2 Simulation de machines de Turing 11

1 Applications du modèle

La calculabilité distribuée est un terme générique pour désigner des calculs effectués à l'aide de plusieurs machines. Il en existe de nombreux paradigmes. Le modèle que nous étudions se place dans le cadre bien précis de populations d'agents non distinctement identifiés, mobiles, et disposant de faibles ressources. Ils ont donc une mémoire limitée et ils ne peuvent échanger des données que deux à deux, à très courte distance. C'est une modélisation parfaite de sondes ou de capteurs de coût moindre.
L'objectif de ce modèle est de mettre à profit cette population pour réaliser des calculs. Il s'agit donc pour un opérateur de fixer un état initial pour tous les agents, qui changeront d'état à la manière d'un automate lors d'une interaction avec un autre agent.
L'un des principaux avantages de ce modèle calculatoire est que les états de ces agents peuvent dépendre des circonstances environnementales et donc prélever et traiter des informations en même temps. Pour illustrer ce propos, on peut citer l'exemple classique de la calculabilité distribuée, à savoir une population d'animaux sur lesquels sont implantées des puces dont l'état initial est neutre, et qui passent dans un état d'alerte dès que l'animal sur lequel elles sont implantés tombe malade. On peut ainsi surveiller en temps réel le nombre d'animaux malades dans la population (voir I.1).
Il est crucial de remarquer que la population de support du calcul mène une existence à part entière : l'expérimentateur n'a pas de contrôle sur le mouvement des agents et donc sur leurs interactions. Il ne peut pas non plus repérer un agent en particulier car ils ne sont pas distinctement identifiables. Ces contraintes se répercuteront sur les protocoles utilisés, et soulèvent la question de la terminaison des calculs. On définira formellement l'aboutissement comme la stabilité de la sortie du processus dans II.

1.1 Compteur dans une population animale

Comme mentionné auparavant, le modèle de calculabilité distribuée que nous étudions convient particulièrement bien à des applications biologiques comme la surveillance de populations animales via de petits capteurs bon marché placés implantés aux animaux. On donne classiquement l'exemple d'un compteur d'animaux atteints par une maladie dans une population donnée afin de surveiller la pandémie. Nous allons prendre de la distance avec cet exemple et étudier une autre application possible des mêmes puces, en rapport avec l'actualité : la surveillance démographique d'une population animale en voie de disparition.
Nous pouvons imaginer que des biologistes désirent surveiller la pérénité d'une population d'une espèce en voie de disparition en influant le moins possible sur leur mode de vie. Pour analyser la vitesse d'extinction de la

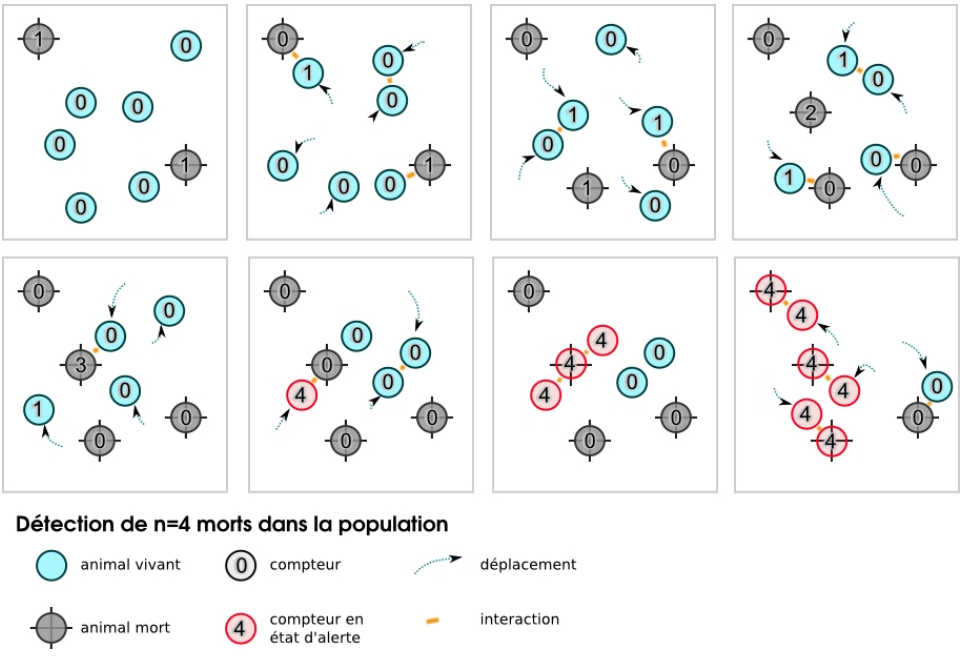
population, les scientifiques décident alors de doter chaque animal d'une puce capable de détecter cet animal est vivant ou non. Afin de limiter le coût totale de l'équipement d'une population potentiellement vaste, ces puces ne disposent que de faibles capacités de mémoire et ne sont capables de communiquer que sur une très courte distance. Elles sont paramétrées de la façon suivante :

– La valeur neutre de ces puces est 0.
– Lorsqu'un animal meurt, sa puce s'incrémente.

On peut imaginer un modèle plus complexe où la puce de toute femelle tombant enceinte serait décrémentée (la sonde pourrait mesurer ses taux hormonaux, par exemple).

Afin de compter le nombre total de morts dans la population, les puces mettent à profit la mobilité des animaux et interagissent selon le schéma :
Lorsque deux puces interagissent, l'une d'elle est remise à zéro, l'autre prend pour valeur la somme de leurs valeurs initiales.

On crée ainsi un compteur qui va peu à peu récolter les données de toutes les puces à mesure que se produiront des interactions entre individus. On peut imaginer que les scientifiques désirent être informés lorsque le nombre de morts dépasse un seuil critique fixé à n. Le paradigme suivant de lecture de la sortie convient :
Quand une puce atteint la valeur n, elle passe dans un état d'alerte qui se propage par interactions successives à toute la population



Les scientifiques n'ont alors plus qu'à lire le message d'alerte sur n'importe quel animal. On a donc réalisé un compteur, dont on voit qu'on peut sans peine le réinitialiser modulo une certaine valeur.

1.2 Utilisations pour l'homme et comparaisons

Les contraintes sur les puces que nous avons décrites ne limitent pas l'utilisation de ce modèle de calculabilité distribuée à la simple surveillance de populations animales. Les puces RFID (Radio Frequency IDentification) répondent en effet très exactement aux mêmes caractéristiques, et elles sont omniprésentes dans la société moderne (cartes de transport, passeports, téléphones GSM, inventaires, bibliothèques, vélibs...) et sont préssenties pour avoir encore d'avantage d'importance dans le futur (remplacement des codes-barres...). Ceci ouvre la porte à autant de nouvelles applications plus ou moins complexes dans la société humaine. Nous étudierons ici l'exemple d'une étude systématisée de la clientèle d'un magasin, dans le but d'adapter ses produits et sa stratégie commerciale en temps réel.
Pour que cet exemple reste simple, nous nous limiterons à la question du sexe majoritaire parmi les clients du magasin. Les puces utilisés peuvent être celles dé téléphones portables des clients ou de leur carte de fidélité. Elles peuvent être lues et initialisées par les portails anti-vols présents à l'entrée du magasin. Nous supposerons qu'il n'est jamais vide pendant sa période d'ouverture.

Toutes les puces portent initialement la donnée du sexe de leur propriétaire (♂ou ♀). Pour comparer le nombre d'individu des deux sexes, nous utilisons le protocole suivant :

- La rencontre de deux personne du meme sexe ne change rien :
  (♂,♂)⇒(♂,♂) ; (♀,♀)⇒(♀,♀)
- Deux personnes de sexe opposées s'annulent et leur puce passe dans un état neutre (○) :
  (♂,♀)⇒(○,○) ; (♀,♂)⇒(○,○)
- Une rencontre avec un état neutre n'a pas de conséquences :
  (♂,○)⇒(♂,○) ; (♀,○)⇒(♀,○) et symétriquement

De cette façon, les sexes opposés s'anihilent deux à deux et il ne reste plus de symboles que du sexe majoritaire. Se pose alors le problème de lire ce résultat. Pour cela, nous allons utiliser un deuxième champ mémoire de la puce afin d'élite un meneur [1] . Il sera désigné par le symbole * et contiendra le résultat final.

Initialement, toutes les puces sont dans l'état meneur. A chaque interaction entre deux meneurs, seul l'un des deux le restera. De plus, tout meneur

[1] L'élection d'un meneur est un processus classique des protocoles de notre modèle de calculs distribués, en particulier pour enregistrer le résultat de comparaisons.

neutre cèdera son privilège de meneur à toute puce non-neutre qu'il rencontrera. De cette façon, toutes les puces non-neutres deviendront un jour meneur et seront prises en comptes, sauf si cela n'est pas nécessaires (si elles font partie de la majorité). On a donc le protocole suivant, où ⚤ représente ♂ ou ♀.

- Rencontre entre deux meneurs : (⚤*,⚤*)⇒(⚤*,⚤) ; (○*,○*)⇒(○*,○)
- Les règles de comparaison ne changent pas :
  (♂,♀)⇒(○,○) et symétriquement
  (♂*,♀)⇒(○*,○) et symétriquement, de même si c'est ♀ qui porte *.
  (♂*,♀*)⇒(○*,○) et symétriquement
- Les meneurs neutres ne sont pas prioritaires : (○*,⚤)⇒(○,⚤*)

Ainsi il suffira de lire la puce du meneur (typiquement le dernier client de la journée) afin de connaître le sexe majoritaire parmi les clients, en supposant un nombre d'interactions suffisantes. On peut ainsi réaliser des comparaisons et des tests d'égalité.

1.3 Calculs parallèles et operations booléennes

On peut imaginer vouloir affiner cette étude de clientèle en repérant par exemple l'age des clients. Par soucis de clarté, nous n'utiliserons que deux tranches d'ages : adolescent • et adulte ⊗, les protocoles réels pouvant être bien plus complexes.
On repère alors une personne par une paire de symboles ♂• ou ♀⊗. On peut alors comparer en parallèle les sexes et âges des clients en utilisant des règles de la forme :
(♂* ⊗ * ,♀* ⊗ *) ⇒ (○* ⊗ * ,○⊗)
Que nous n'allons pas expliciter ici par soucis de lisibilité. C'est ainsi que l'on peut réaliser plusieurs calculs en parallèle, ce qui permet ensuite de créer des combinaisons booléennes dans l'interprétation du meneur final (on peut par exemple déclarer que la clientèle correspond à la clientèle espérée si on a un meneur final ♀∧•).

2 Modèle formel des protocoles de population

Pour raisonner théoriquement sur ce type de protocoles et en cerner l'étendue des possibilités, il nous faut formaliser ce modèle de calculabilité distribuée. Nous nous inspirerons du modèle proposé par *Anguin & all* [1].

2.1 Protocoles de population

Definition 2.1. ***Population***
Une population $\mathcal{P}$ est un ensemble de n éléments appelés agents.

Definition 2.2. ***Interactions***
Sur une population $\mathcal{P}$, on définit une relation binaire $\leftrightarrow$ de $\mathcal{P} \times \mathcal{P}$ telle que $p_1 \leftrightarrow p_2$ représente le fait que p_1 interagisse avec p_2, c'est à dire que p_1 et p_2 sont à un moment donné à portée de communication.

Cette relation n'est pas réflexive, ni a priori symétrique : on remarque donc la présence d'un initiateur et d'un récepteur dans les interactions. Cela traduit le fait que dans la réalité, les deux agents qui se rencontrent ne peuvent émettre ni recevoir le signal exactement au même moment. On peut toutefois simuler des relations symétriques en choisissant au hasard l'initiateur. Dans beaucoup d'applications, on suppose également le graphe de cette relation complet sur les paires ordonnées distinctes, c'est à dire que $\forall (p_1, p_2) \in \mathcal{P}^2, p_1 < p_2 \Longrightarrow (p_1 \leftrightarrow p_2)$.

Definition 2.3. ***Protocole de population***
On définit un protocole de population sur les alphabets d'entrée A_e et de sortie A_s donnée par la réunion de :

- *Un ensemble fini Q d'états*
- *Une fonction d'entrée $E : A_e \to Q$*
- *Une fonction de sortie $S : Q \to A_s$*
- *Une fonction de transition $\delta : Q \times Q \to Q \times Q$*

La fonction δ décrit le comportement de deux agents lors de leur rencontre. Les fonctions E et S permettent de contrôler le déroulement du protocole.

Definition 2.4. ***Configuration de population***
Une configuration de la population $\mathcal{P}$ dans un protocole donné est une fonction $C : \mathcal{P} \to Q$ associant à chaque agent de $\mathcal{P}$ un état de Q.

Definition 2.5. ***Transition***
Une transition d'un protocole de population est une paire de configurations $C \xrightarrow{p_1 \leftrightarrow p_2} C'$ telle que :
Pour le couple $p_1 \leftrightarrow p_2$, on ait $(C'(p_1), C'(p_2)) = \delta(C(p_1), C(p_2))$,
et C' coincide avec C pour les autres agents de $\mathcal{P}$ qui sont donc invariants.

2.2 Exécution et calcul

Nous allons pouvoir utiliser le protocole de population défini ci-dessus pour effectuer des calculs grâce aux définitions suivantes.

Definition 2.6. ***Affectation d'entrée***
Une affectation d'entrée est une fonction $W : \mathcal{P} \to A_e$ qui définit un mot d'entrée w [1] *sur l'alphabet d'entrée dont chaque lettre correspond à un agent.*

Definition 2.7. ***Configuration initiale***
Pour tout mot $w \in (A_e)^{|\mathcal{P}|}$, on définit la configuration initiale du protocole de population sur $\mathcal{P}$ d'entrée w comme : $\forall p \in \mathcal{P}, C_w(p) = E(W(p))$

Definition 2.8. ***Exécution***
L'éxécution d'un protocole sur $\mathcal{P}$ avec l'entrée w est une suite de transitions : $C_w = C_0 \to C_1 \to \cdots \to C_n \to \cdots$

Definition 2.9. ***Calcul***
Une éxécution est un calcul si pour toute transition $C \to C'$, si C apparait un nombre infini de fois dans l'exécution, alors C' aussi.

Definition 2.10. ***Affectation de sortie***
Pour toute configuration C, on définit une affectation de sortie $Y_C : \mathcal{P} \to A_s$ qui définit un mot de sortie y_C sur l'alphabet de sortie par la fonction de sortie S du protocole, formellement $\forall p \in \mathcal{P}, Y_C(p) = S(C(p))$

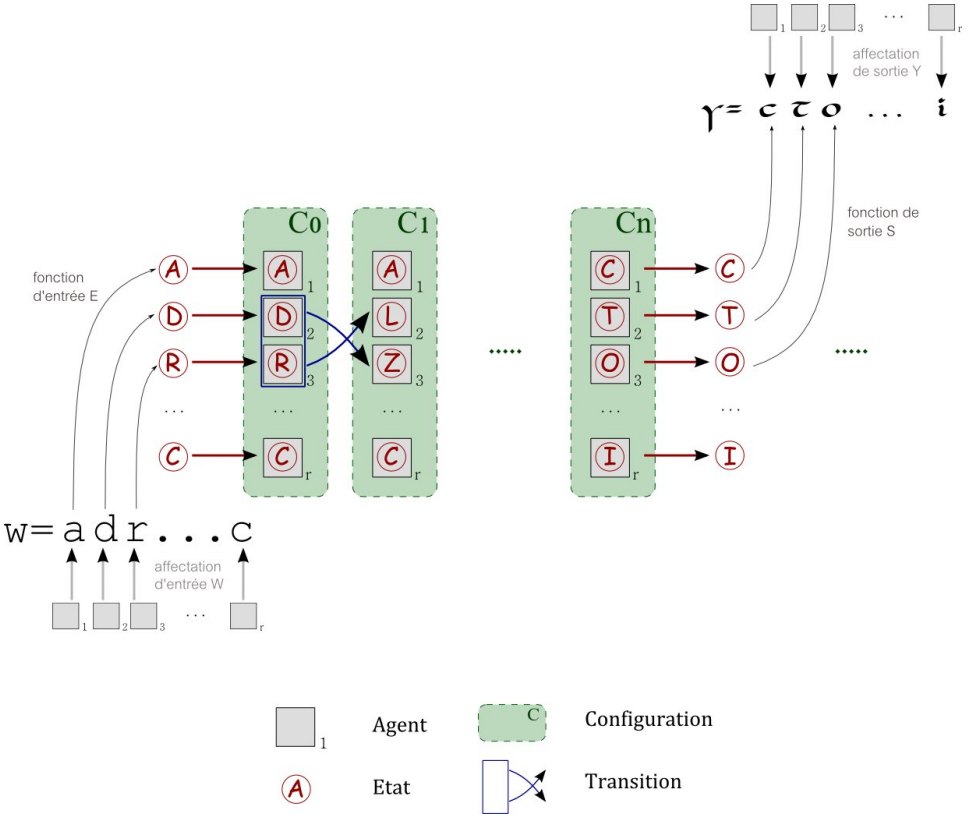
Definition 2.11. ***Configuration stable***
La configuration C est stable si pour toute configuration C' accessible depuis C ($C \to \cdots \to C'$) le mot de sortie reste le même, soit $Y_C = Y_{C'}$.
Si un calcul contient une configuration stable, il est dit ***convergent****.*

Il est important de noter qu'on ne demande pas l'égalité des configurations qui suivent C mais juste celle des mots de sortie.

Definition 2.12. ***Calcul d'une relation et d'une fonction***
Si pour tout mot d'entrée $w \in (A_e)^{|\mathcal{P}|}$, tous les calculs possibles dans $\mathcal{P}$ convergent, on dit que le protocole $\mathcal{P}$ ***converge toujours****. Il calcule alors stablement la relation $R : (A_e)^{|\mathcal{P}|} \times (A_s)^{|\mathcal{P}|}$,*
$R(w, y) \Leftrightarrow$ Il existe un calcul dans $\mathcal{P}$ de mot d'entrée w se stabilisant sur le mot de sortie y
Si de plus w n'est en relation qu'avec un unique y, $\mathcal{P}$ calcule stablement la fonction $\mathcal{F} : \mathcal{F}(w) = y$

[1] L'ordre des lettres est alors donné par l'ordre sur l'ensemble fini des agents. C'est aussi le cas pour le mot de sortie.

2.3 Schema de fonctionnement



2.4 Lien avec les machines de Turing

On peut considérer le modèle des protocoles de population comme une variante très éloignée des machines de Turing à une bande. En effet, chaque agent a un comportement similaire à une case de la bande, il enregistre une information qui peut être modifiée par la suite. Ainsi, le modèle de protocole de population est analogue à une bande finie, dont chaque case mène une existence indépendante. La valeur d'une case est alors modifiée suivant la fonction δ à chaque rencontre d'une autre case. Dans le cas particulier où la relation d'interaction forme une ligne, on peut même retrouver une bande linéaire.

3 Calculabilité de l'arithmétique de Presburger

Nous allons voir que la portée calculatoire du modèle de protocole de population coincide exactement avec la restriction de l'arithmétique sur les entiers définie par Mojzesz Presburger [2], que nous allons donc formuler et étudier dans un premier temps.

3.1 Logique et arithmétique de Presburger

Definition 3.1. ***Arithmétique de Presburger***
L'arithmétique de Presburger est une théorie égalitaire du premier ordre définissant 0, l'opération successeur et l'addition. $(\mathbb{N}, +)$ *en est un modèle.*

On peut remarquer qu'il s'agit de l'arithmétique usuelle de Peano après suppression de la multiplication. Cette théorie est cohérente, complète, et, contrairement à l'arithmétique de Peano, décidable.
L'opérateur successeur définit un **ordre total discret noté** $<$ que l'on peut sans problème formaliser. Une théorie logique du premier ordre peut utiliser les quantificateurs $\forall$ et $\exists$, ce qui semble peu compatible avec notre modèle, d'où la nécessité de la modifier quelque peu.

Definition 3.2. ***Prédicat***
Un prédicat ϕ_F *d'une théorie est une fonction à valeurs dans* $0, 1$ *dont les variables sont dans le modèle de la théorie (ici* $\mathbb{N}$*).*
Toute formule F *construite avec les éléments du langage d'une théorie (ici 0,* $+$ *et successeur), des variables et des quantificateurs définit le prédicat* ϕ_F *suivant :*
Pour toute assignation $X_1 \leftarrow u_1, \ldots, X_n \leftarrow u_n$ *des variables libres (non-quantifiées) de* F*,* $\phi_F(u_1, \ldots, u_n)$ *est égale à la valeur de vérité de la formule* F *quand les variables libres* X_i *sont interprétées par les* u_i*.*

Lemme 3.1. ***Arithmétique de Presburger étendue***
On peut ajouter à l'arithmétique de Presburger la relation binaire $\equiv_m$ *de congruence modulo une constante* m *sans modifier l'ensemble des prédicats qu'on peut y définir.*
Tout prédicat de l'arithmétique de Presburger peut alors être définie dans ce langage sans quantificateur.

Démonstration. L'origine de ce lemme est obscur. Presburger [2] en donne quelques indications, et Kracht [3] en présente une preuve détaillée. □

3.2 Portée calculatoire du modèle

Il existe de nombreuses conventions qui permettent de définir les entrées et les sorties d'un protocole de population en fonction d'un calcul à effectuer.

On peut représenter un entier N par un symbole que l'on affecte initialement à exactement N agents, ou en assignant directement la valeur N à un agent donné (sous la forme d'un vecteur de 0 et 1). C'est le rôle de l'affectation d'entrée. De même, l'affectation de sortie permet de choisir une convention de sortie. [1]

Definition 3.3. ***Prédicat calculable par un protocole de population*** *Un protocole de population calcule stablement le prédicat ϕ_F si, pour toute entrée $(u_1, \ldots, u_n)$, la sortie du protocole est $11\ldots1$ si et seulement si $\phi_F(u_1, \ldots, u_n) = 1$ et $00\ldots0$ si et seulement si $\phi_F(u_1, \ldots, u_n) = 0$.*

Lemme 3.2. ***Combinaison booléenne***
Toute combinaison booléenne de prédicats calculables stablement par un protocole de population est calculable stablement par un protocole de population.

Démonstration. On le démontre par récurrence, le cas de la fonction à deux variables étant réglé en combinant les états $Q_{\mathcal{A}}$ et $Q_{\mathcal{B}}$ des protocoles $\mathcal{A}$ et $\mathcal{B}$ astucieusement pour définir les états du protocole total $Q = Q_{\mathcal{A}} \times Q_{\mathcal{B}}$. La définition des transitions et la stabilité en découle. Une preuve complète est apportée par Anguin & al [1]. Ce propos est illustré en I.3. □

Théorème 3.3. ***Calculabilité de l'arithmétique de Presburger***
Tout prédicat défini dans l'arithmétique de Presburger est calculable stablement par un protocole de population.

Démonstration. Soit ϕ prédicat de l'arithmétique de Presburger. On peut le convertir en ϕ', équivalente, sans quantificateurs, qui est une combinaison booléenne de relations $=$, $\equiv_m$ et $<$ entre des termes de la forme $\sum_i \alpha_i x_i + c$. L'égalité peut être réécrite comme combinaison de deux inégalités.
A partir des exemples de la première partie, on peut extrapoler des protocoles de population qui calculent ces cas de base. Ils sont formellement définis dans l'article d'Anguin & al [1]. La cloture par opérations booléenne permet de conclure.

□

Ce théorème se verra complété par une réciproque deux années plus tard par Anguin & al [4] :

Théorème 3.4. ***Portée calculatoire des protocoles de populations***
L'ensemble des prédicats calculables stablement par des protocoles de population est l'ensemble des prédicats de l'arithmétique de Presburger.

[1] La convention utilisée ici (*convention de prédicat totale*) invalide toute sortie qui n'est pas de la forme $aa\ldots a, a \in 0, 1$.

4 Complexité pour des rencontres aléatoires

Les calculs pouvant être infinis, la stabilité peut a priori être atteinte après un temps arbitrairement grand, d'où la nécessité de se pencher sur la question. Cependant, les interactions étant décrites par la relation $p_1 \leftrightarrow p_2$ si et seulement si ils interagissent à un moment quelconque, la prise en compte du temps n'est pas à proprement parler possible : il n'y a pas de différence entre des interactions fréquentes et rares. Toutefois, on peut l'estimer en comptant le nombre d'interactions nécessaires à la satbilisation du protocole. Pour ce faire, il faut étudier plus précisément la relation d'interaction.

4.1 Interactions aléatoires et complexité

Nos considérations se porteront sur la relation d'interaction complète la plus simple possible :

Definition 4.1. ***Automate conjugant***
Un protocole de population est un automate conjugant si la paire d'agents sur le point d'interagir est choisie parmi toutes les possibilités avec une probabilité uniforme et indépendante.

Dans ce modèle, nous pouvons établir le résultat suivant :

Théorème 4.1. ***Complexité de calcul d'un prédicat***
Tout prédicat calculable stablement par un protocole de population est calculable avec la probabilité 1 par un automate conjugant en un nombre total de $O(n^2 log(n))$ *interactions.*

Démonstration. La démonstration de ce théorème est complexe et ne sera pas étudiée ici. Elle est proposée par Anguin & al [1].

□

4.2 Simulation de machines de Turing

Dans ce même article, les auteurs extrapolent et proposent une méthode basée sur l'élection d'un meneur (*cf. 1.2*) pour simuler avec grande probabilité un compteur dont la valeur est la somme des valeurs portée par tous les agents. Le meneur incrémente ou décrémente le compteur en faisant l'opération sur un des agents. Pour tester si le compteur est nul, le meneur cherche dans la population un agent non nul. Il se prononcera après un nombre donné de rencontres avec le même agent, marqué par le meneur.
D'autre part, on peut montrer que les machines de Turing peuvent se réduire aux compteurs (*par la réduction de Minsky [5] par exemple, c'est à dire en réduisant la bande de la machine de Turing en deux piles*). Ainsi il apparait qu'un automate conjugant permet de simuler une machine de Turing, malgré une faible probabilité d'erreur.

Théorème 4.2. ***Simulation d'une machine de Turing***
Toute machine de Turing de complexité spatiale logarithmique sur un alphabet unaire, et de complexité temporelle $O(n^d)$ dans le pire des cas est simulable sur des entrées de taille inférieures à n par un protocole sur une population de n agents, avec pour tout c une complexité temporelle de $O(n^{d+2}log(n) + n^{2d+c+1})$ contrebalancée par une probabilité d'erreur de $O(n^{-c}log(n))$.

Démonstration. La démonstration, également complexe, peut être trouvée dans [1].

□

Nous voyons donc que les possibilités de calcul des protocoles de populations approchent celles des machines de Turing, en présentant l'avantage majeur de pouvoir traiter des informations de l'environnement. Ce modèle semble d'autant plus judicieux pour les applications pratiques qu'il repose sur des capteurs de moindres coûts, parfaitement déployables à grande échelle, comme la puce RFID, déjà omniprésente dans le monde actuel.

Références

[1] Dana Angluin, James Aspnes, Zoë Diamadi, Michael J.Fischer et René Peralta, "Computation in networks of passively mobile sensors"
(2005) PODC'04 ACM 290-299.

[2] Presburger "Über die vollständigkeit eines gewissen systems der arithmetik ganzer zahlen, in welchem die addition als einzige operation hervortritt"
(1929) Comptes Rendus du I congrès de Mathématiciens des Pays Slaves

[3] Kracht, M. : The Mathematics of Language, Studies in Generative Grammar
(2003) Vol. 63. Mouton de Gruyter. ISBN 3-11-017620-3

[4] Dana Angluin, James Aspnes et David Eisenstat "Stably computable predicates are semilinear"
(2006) PODC'06 ACM 292-299.

[5] Minsky, M.L. : Computation : Finite and Infinite Machines.
(1967) Prentice-Hall Series in Automatic Computation, Englewood Cliffs, N.J.

Merci à Xavier Koegler pour son aide et ses conseils.