Langages formels, Calculabilité, Complexité
Calculabilité distribuée

Louis Jachiet

20 décembre 2010

Résumé

Nous nous intéresserons ici au modèle des protocoles de population ou population protocols. Dans ce modèle le calcul est distribué à de petits agents, anonymes, sans contrôle sur leur déplacement. On s'attachera d'abord à définir le modèle et en donner un exemple. Ensuite on montrera que les prédicats définis dans l'arithmétique de Presburger sont définissables par des protocoles de population. Enfin nous donnerons quelques résultats exemplaires qui étendent le point de vue sur la calculabilité distribuée en général.

1 Le modèle des population protocols

1.1 Motivations et premier exemple

Imaginons que l'on veuille effectuer une surveillance sanitaire sur une population d'oiseaux. De quelles manières peut-on répondre à des questions comme : existe-t-il N oiseaux malades, avec N petit ?

Une première idée consiste à donner un identifiant à chaque oiseau, le doter d'un capteur qui détermine son état de santé ainsi que d'un petit émetteur. À chaque fois que celui ci passe devant une station relais, la station conserve son état de santé. Cette idée simple suppose la connaissance d'un identifiant unique à chaque oiseau et suppose que les oiseaux passent tous près de la tour et n'est donc pas celle envisagée car on veut que la mémoire soit une constante indépendante de la taille de la population.

L'idée du population protocols est d'éviter l'utilisation d'une "tour" ou même d'un système d'identifiants dont la taille mémoire grandirait avec la taille de la population. Pour cela chaque oiseau est doté d'un capteur et d'une minuscule unité de calcul. Le capteur contient un nombre entre 0 et N. Dès qu'un oiseau tombe malade son compteur passe à 1. Lorsque deux puces sont à une distance permettant la communication, l'une transfère à l'autre son état, et l'autre calcule son nouvel état : soit la somme des deux premiers soit N si le résultat est plus grand que N, la première puce repassant alors dans l'état 0. Dès qu'une puce passe dans l'état N, elle transmet à ses voisins l'état N sans repasser elle même dans l'état 0. Donc s'il y a N oiseaux malades dans une population les puces convergeront toutes vers l'état N. Il suffit donc, de ne lire que l'état d'un seul oiseau pour savoir si plus de N oiseaux sont malades.

2 Définition

Définition 2.1. *On définit une population comme la donnée d'un graphe orienté. Les nœuds correspondants à nos agents (ou capteurs ou puces selon le besoin) et les arcs correspondent aux interactions possibles. Les interactions, comme dans l'exemple plus haut, ne sont pas nécessairement symétriques.*

Définition 2.2. *Un protocole de population est la donnée de* $(I, O, Q, f_I, f_O, \delta)$ *avec :*

- I *un alphabet d'entrée*
- O *un alphabet de sortie*
- Q *un ensemble d'états*
- $f_I : I \to Q$ *une fonction d'initialisation*
- $f_O : Q \to O$ *une fonction de lecture de résultat*
- $\delta : Q \times Q \to Q \times Q$ *une fonction de transition quand deux agents se rencontrent.*

Définition 2.3. *Un $n-$uplets d'états est nommé configuration. Une suite de configurations est dite une suite de configurations valide si deux éléments consécutifs n et $n+1$ de la suite ne diffèrent que par deux éléments, mettons i et j, et que $\delta(a_i(n), a_j(n)) = (a_i(n+1), a_j(n+1))$*

Définition 2.4. *Un calcul par un protocole de population sur une population $a_1, ..., a_n$ est une valeur de chaque agent dans l'alphabet d'entrée et une suite de configurations valide.*

Définition 2.5. *On est calculable stablement s'il existe un protocole de population et si l'image par la fonction de lecture de résultat de tous les agents converge vers le même caractère limite qui ne doit pas non plus dépendre de l'ordre des interactions. À la restriction près que toutes les interactions doivent apparaitre un nombre infini de fois ; ce qui est le cas avec une probabilité 1.*

Note Dans ce cas, on peut définir la notion de résultat d'un calcul. Ce résultat étant la limite commune de la fonction de lecture de résultat

Retour à l'exemple : On avait $Q = (0, ..., N)$, $I = (Malade, Sain)$, $O = (Oui, Non)$, $f_I(Malade) = 1$, $f_I(Sain) = 0$, $f_O(N) = Oui$, $\forall i \leq N - 1 f_O(i) = Non$. Enfin on avait $\delta(a, b) = (min(a + b, N), 0)$. Il est clair que c'est un protocole qui calcule stablement. En effet s'il n'y pas N oiseaux malades alors la lecture renverra toujours Non et si au contraire il y a N oiseaux malades l'information se propagera.

3 Théorème (*Angluin et al 2004*)

Théorème 3.1. *Les protocoles de populations peuvent calculer stablement les prédicats du premier ordre de l'arithmétique de Presburger.*

La preuve qui suit n'est pas tout à fait celle de Angluin bien qu'elle s'en inspire. Il s'agit d'une preuve personnelle. Pour bien la comprendre un exemple est nécessaire, il y en a un à la fin.

3.1 L'arithmétique de Presburger

L'arithmétique de Presburger est l'arithmétique de Peano privée de la multiplication. Cette arithmétique permet de définir les formules $\phi_n : \forall x \exists y z$ $x = n * y + z \wedge z < n$. En considérant ces formules comme des axiomes, on peut définir les fonctions $Q_n(x) = \lfloor x/y \rfloor$.

3.2 Élimination des quanteurs sur les formules de l'arithmétique de Presburger

Lemme 3.2. *Dans le langage $L = \{0, 1, +, \prec, Q_n\}$, l'arithmétique de Presburger admet l'élimination des quanteurs.*

Ce lemme est long à démontrer et principalement technique. Sa preuve a été faite en TD de logique (TD9). Il est admis ici.

Note Comme l'arithmétique de Presburger admet l'élimination des quanteurs dans le langage L, on se fixe ce langage et on utilise des formules sans quanteurs.

3.3 Expressions dans l'arithmétique de Presburger

Une formule est, avant tout, composée d'expression qu'il va falloir calculer avec notre modèle. Pour cela on représente les entiers de manière distribuée. Si pour calculer notre formule on a besoin de manipuler n entiers alors les états de nos agents seront les éléments de $\{-M, ..., -1, 0, 1, ..., M\}^n +$ 2 où M est une constante du protocole choisie à l'avance et suffisamment grande pour nous permettre les calculs. (En fait, il suffit que M soit supérieur à tous les p dans les $Q_p(...)$ qui apparaissent dans la formule et au nombre de variables.)
On manipule des entiers de $\mathbb{Z}$, cela nous permet de transformer les tests $a = b$ en $a - b = 0$ et $a < b$ en $a - b < 0$.

On introduit un nouveau langage d'expression arithmétique. Dans ce langage les expressions peuvent réutiliser le résultat d'un calcul précédent. Les résultats sont stockés dans des variables (v_i) et chaque variable n'a le droit d'apparaitre qu'une seule fois dans un calcul. Les expressions sont de la forme : $v_i := v_j + v_k$, $v_i := Q_p(v_j)$, $v_i := -v_j$, $i > j, k$ ou enfin $v_i := x_j$ où x_j est une variable libre de l'expression et donc une donnée du problème. Quand v_j apparaît dans l'expression de v_i on dit que v_i est capturé par v_j. Il est clair que toutes les expressions peuvent être calculées de cette manière si l'on représente l'expression sous la forme d'un arbre où chaque nœud sera remplacé par une variable.

À l'aide de ce langage on introduit simplement un langage de formule qui est juste le langage d'expression, plus les conjonctions et disjonctions de tests du type (=0) ou (>0) que l'on ne s'autorise que pour des variables non capturées.

3.4 Calcul d'un prédicat à l'aide d'un protocole de population

L'idée principale c'est que les entiers sont représentés de manière distribuée sur l'ensemble des agents. Chaque agent contient une petite partie

(entre $-M$ et M) de chaque variable. On a besoin d'un nombre $O(n)$ d'agents où n est la plus grande coordonnée de l'entrée.

3.4.1 Lecture de l'entrée

Comme l'alphabet est fini et qu'il n'existe pas d'ordre entre les agents, on est obligé de donner l'entrée en unaire. Notre alphabet d'entrée est donc $\{0,1\}^n$, où n est le nombre de variables libres dans la formule. La fonction d'initialisation associe à $(x_1, ..., x_n)$ l'état ayant des 0 sur chaque composante sauf celles où $v_i := x_j$ avec $x_j = 1$.

3.4.2 Transitions

Bien sûr dans l'automate final il s'agira de transitions, mais expliquons de manière imagée ce qu'il se passe lors d'une transition. Quand deux agents se rencontrent, ils font pour chaque variable v_i, la somme de ce chacun possède de cette variable, puis ils appliquent les règles de simplification suivantes, jusqu'à ce que plus aucune règle ne puisse s'appliquer :

- Si $v_i := Q_p(v_j)$: Si $v_j \geq p$ et $v_i < 2M$ alors on passe à $v_j := v_j - p$, $v_i := v_i + 1$. Inversement si $v_j \leq -n$ et $v_i > -2M$ alors $v_j := v_j + p$, $v_i := v_i - 1$.
- Si $v_i := -v_j$: On pose $K = max_{|k|}\{k * v_j < 0 \text{ et } |v_j + k| \leq 2M \text{ et } |v_i + k| \leq 2M\}$ et alors $(v_i, v_j) = (v_i - K, v_j + K)$.
- Si $v_i := v_j + v_l$: On choisit d'abord $K = max_{|k|}\{k * v_j > 0 \text{ et } |v_j - k| \leq 2M \text{ et } |v_i + k| \leq 2M\}$ et ensuite $K' = max_{|k|}\{k * v_l > 0 \text{ et } |v_l - k| \leq 2M \text{ et } |v_i + k + K| \leq 2M\}$ et on pose $(v_i, v_j, v_l) = (v_i + K + K', v_j - K, v_l - K')$

Enfin ils doivent se partager le total et donc pour cela on favorise un des deux agents en le remplissant le plus possible.

Si nos agents sont A_1 et A_2 on a alors pour tout i si $v_i \geq 0$: $v_i(A_1) := min(v_i, M)$, $v_i(A_2) := max(0, v_i - M)$. Inversement si $v_i < 0$: $v_i(A_1) := max(v_i, -M)$, $v_i(A_2) := min(0, v_i + M)$

Pour finir il faut définir la transition pour les deux dernières coordonnées la première sert à savoir si on est un élément maximal et le second sert à savoir si du point de vue du dernier élément maximal la formule était vraie ou non. Donc si les deux éléments étaient non maximaux les deux dernières coordonnées sont inchangées. Si au moins un des deux était marqué comme maximal alors l'élément privilégié est marqué maximal et l'autre non. Tous

deux marquent la valeur de vérité à vraie si elle l'est pour les quantités qu'ils possèdent de chaque variable (seul les variables non capturées sont utilisées dans la formule).

3.4.3 lecture du résultat

Le résultat est simplement la lecture de la dernière coordonnée

3.5 Convergence et preuve du théorème

Lemme 3.3. *Le protocole de population respecte les propriétés suivantes :*

1. *Pour chaque coordonnée i il existe une étape de calcul telle que pour tous les agents v_i soit nulle ou si $v_i := Q_p(v_j)$ alors $|v_i| < p$*
2. *Il existe une étape de calcul à partir de laquelle pour chaque i tous les agents ont le même signe de v_i*
3. *Il existe une étape de calcul à partir de laquelle il n'y plus qu'un seul agent qui est maximal, et il a, pour chaque i un $|v_i|$ maximal.*

On peut considérer que δ n'est pas $Q \times Q \to Q \times Q$ mais $Q \times Q \to \{Q, Q\}$. Aussi les propriétés de convergence s'étudient non pas en tant que n-uplets mais en tant qu'ensemble à n éléments

Démonstration. D'abord on voit que la somme en valeur absolue sur tous les agents et sur toutes les coordonnées décroit. En effet à chaque fois on ne rajoute dans une variable que si on en a pris plus dans une autre. Comme on a par choix de M, $M \geq n$, la capacité de stockage dans une variable est $M * nombreagents$ alors qu'il y avait au départ un total de moins de $n * nombreagents$. Il est donc impossible qu'une variable soit saturée. Pour chaque agent et pour chaque coordonnée capturée v_i, il ne peut rester de cette quantité que si c'est un Q_p et $|v_i| < p$. D'une part si la propriété est vraie pour tous les $j \leq i$ à une étape de calcul elle restera vraie tout le temps car toutes les variables sont capturées par des variables d'indices supérieurs. Sinon s'il existe un agent tel qu'il existe une coordonnée v_j soit contre-exemple, on prend j minimal. On a alors forcément que la coordonnée qui la capture est saturée sinon elle serait transmise au moindre contact avec un agent non saturé mais il est impossible que cette coordonnée qui capture soit saturée pour tous les agents. □

Démonstration. D'après 1, on voit que les valeurs capturées se stabilisent et donc lors des rencontres des agents on a par construction pour chaque coordonnée un des agents qui va accumuler la quantité que les autres possèdent de cette variable et on aura au final trois types d'agents : ceux qui ont cette coordonnée saturée, ceux qui ont cette coordonnée nulle, et un seul agent qui possède le reste. □

Démonstration. Cette propriété est claire : lors de chaque rencontre un des deux agents est supérieur à l'autre en considérant une rencontre avec chaque agent on aura bien, par transitivité, que un des agents sera supérieur à tous les autres. □

Comme il existe un élément maximal et que toutes les agents s'accordent à ce qu'il dit, il suffit de voir que l'élément maximal donne la bonne valeur de vérité. Comme c'est un élément maximal et que tous les autres ont sur chaque coordonnée le même signe. L'élément maximal à sur chaque coordonnée une valeur strictement positive si et seulement si la variable est strictement positive. Donc la valeur de vérité de l'élément maximal correspond à partir d'une certaine étape de calcul à celle de la formule.

Preuve du Théorème : Ce protocole de population converge vers la valeur de vérité de la formule qu'on a choisi quelconque donc les protocoles de population calculent les prédicats de l'arithmétique de Presburger ! ■

3.6 Exemple

Prenons par exemple le problème de savoir si $y < Q_3(x)$. On pose alors :
- $v_1 := x$
- $v_2 := Q_3(v_1)$
- $v_3 := y$
- $v_4 := v_2 - v_3$

On a $M = 4$. Les états sont donc $\{-4, -3, -2, -1, 0, 1, 2, 3, 4\}^4\{V, F\}\{0, 1\}^2$. Je supprime les états dont toutes les coordonnées sont à 0 car ils ne jouent plus aucun rôle, ils s'accorderont nécessairement avec ce que dit l'unique agent qui est maximal.

Voici sur une population le résultat :

états	12 34 V M	12 34 V M	12 34 V M	12 34 V M	12 34 V M	12 34 V M
étape 1	10 00 F 1	10 00 0 1	10 00 F 1	10 10 F 1	10 10 F 1	10 00 F 1
étape 2	**00 00 F 0**	**20 00 F 1**	10 10 F 1	10 00 F 1	10 10 F 1	10 00 F 1
étape 3		20 00 F 1	10 00 F 1	10 00 F 1	**20 0 -1 F 1**	**00 00 F 0**
étape 4		**00 01 V 1**	**00 00 F 0**	10 00 F 1	20 0 -1 F 1	
étape 5		**10 01 V 1**		**00 00 V 0**	20 0 -1 F 1	
étape 6		**00 00 V 1**			**00 0 0 V 0**	

Donc la réponse est V.

3.7 Autres résultats

Théorème 3.4 (Angluin (2006))**.** *L'ensemble des prédicats calculables par un protocole de population sont exactement ceux de l'arithmétique de Presburger.*

Ce théorème renforce donc le précédent qui ne nous donnait qu'une borne inférieure sur la calculabilité des protocoles de population

La calculabilité distribuée peut s'aborder sous des formes diverses mais des limitations peuvent intervenir et ce de façon diverses : la capacité individuelle de stockage qui est très limité face à la taille de la population et donc l'impossibilité par exemple d'avoir des identifiants uniques. C'est cette limite qui est notamment à l'origine de l'impossibilité pour les protocoles de population de calculer plus que l'arithmétique de Presburger. Un autre type de limitation mais qui n'a pas été abordé ici vient des problèmes de synchronicité. En effet ce problème est majeur dans le cas d'allocation de ressources. Le problème du dîner des philosophe en est une illustration.

Le dîner des philosophes

- On a N philosophes autour d'une table
- Chaque philosophe a devant lui un plat
- Entre deux assiettes se trouve une seule fourchette

Chaque philosophe pense pendant un temps indéterminé, puis il a faim pendant un certain temps T et tente après d'attraper une première fourchette puis une seconde, s'il réussit il mange sinon il recommence à avoir faim et attendre. Le problème de savoir comment ordonnancer les philosophes pour que au bout d'un temps fini chaque philosophe qui a faim soit nourri. Ce problème peut paraitre simple mais :

Théorème 3.5. *Il n'existe pas d'algorithme déterministe qui résolve le problème du diner des philosophes avec le même algorithme pour tous les philosophes s'ils sont synchrones.*

Démonstration. En effet pour tout algorithme déterministe et synchrone, les philosophes devraient faire exactement les mêmes gestes et donc s'il y arrivaient, manger en même temps. Il ne peuvent pas tous manger en même temps donc il n'existe pas d'algorithme. □

4 Références principales

Angluin et al., 2004 Dana Angluin, James Aspnes, Zoë Diamadi, Michael J. Fischer et René Peralta (2004). Computation in networks of passively mobile finite-state sensors. In PODC '04 : *Proceedings of the Twenty-Third Annual ACM Symposium on Principles of Distributed Computing, pages 290–299. ACM Press.*

Angluin et al., 2006 Dana Angluin, James Aspnes et David Eisenstat (2006). Stably computable predicates are semilinear. *In PODC '06 : Proceedings of the Twenty-Fifth Annual ACM Symposium on Principles of Distributed Computing, pages 292–299. ACM Press..*

Koegler , 2006 *Théorie algorithmique des jeux.*

D Angluin, J Aspnes, MJ Fischer Self-stabilizing population protocols. *ACM Transactions on, 2008*