# TAO is SILENT

*Les machines LISP chez NTT*

Jean-Pierre Briot
François-Xavier Testard-Vaillant

Le laboratoire des sciences de l'information est un des laboratoires de recherche fondamentale de NTT, situé dans le complexe de Musashino, légèrement à l'ouest de Tokyo *intra-muros*. Nous avons pu rencontrer Ikuo Takeuchi, qui y dirige une équipe depuis environ une vingtaine d'années et conçoit des ordinateurs spécialisés, adaptés au calcul symbolique, appelés machines Lisp. Ikuo Takeuchi est mondialement connu pour la fonction qui porte son nom. Cette fonction a pour caractéristique d'engendrer de très longs et volumineux calculs sur un calculateur naïf alors qu'elle est formellement équivalente à une fonction dont le temps de calcul est constant quelque soit sa donnée[1]. Après une digression sur la fonction `tak` et son histoire, I. Takeuchi[2] est entré dans le vif du sujet qui motivait notre visite, i.e. le projet SILENT de NTT. Ce projet prend la suite du projet TAO qui a conduit NTT à concevoir et construire une machine Lisp.

## CALCUL SYMBOLIQUE

Rappelons que par opposition au calcul numérique mis en oeuvre dans les applications scientifiques, le calcul symbolique manipule principalement des symboles formels. Son domaine d'application traditionnel et privilégié est l'Intelligence Artificielle pour la résolution de problèmes, la représentation des connaissances, la traduction etc. Cependant les applications informatiques devenant de plus en plus sophistiquées, cette approche tend à se généraliser (le système d'exploitation du Newton d'Apple est par exemple entièrement conçu au moyen d'un langage symbolique, Dylan). Les langages de programmation à objets sont d'ailleurs les représentants les plus récents du calcul symbolique. Deux autres grands représentants sont les langages de programmation fonctionnelle, le plus ancien et le plus connu étant Lisp, et les langages de programmation logique comme Prolog.

## MACHINES DÉDIÉES

Pour obtenir d'une part une efficacité optimale et d'autre part un environnement de développement adapté au mieux à un langage de programmation, les chercheurs ont commencé à s'intéresser au début des années 70 à des ordinateurs dédiés à un langage. Le premier prototype a été la machine Smalltalk, également premier environnement de développement à objets. L'idée sous-jacente est que l'ensemble de l'environnement de la machine, incluant donc le système d'exploitation, la gestion des fichiers, l'affichage etc., soit programmé dans un même langage, de surcroît de haut niveau et non pas un assembleur classique. Ce type de machine permet ainsi de disposer d'un environnement de programmation très sophistiqué, d'une efficacité optimale, et surtout d'accéder à tous les aspects de la machine, jusqu'au système d'exploitation, via un unique langage de programmation. Il n'y a donc aucune distinction stricte entre langage machine et langage utilisateur. Cette machine Smalltalk a entrainé le développement de machines Lisp au MIT[3]. La société Symbolics, issue du MIT, en est la plus illustre représentante. Des machines Prolog ont ensuite également été développées, notamment dans le cadre du projet Japonais des ordinateurs de 5ème génération : ICOT (1982-1992).

Signalons qu'une machine Lisp a également été développée en France au début des années 80 par un projet CGE+CNET : MAIA. Ce projet avait pour objectif d'étudier la réalisation d'une machine pour le calcul symbolique autour d'un processeur conçu à cet effet. Contrairement à son *alter ego* japonais, le projet français n'a donné lieu à aucune commercialisation. La tendance du marché des machines dédiées s'est en effet inversée à partir du milieu des années 80, comme nous allons le voir. Les constructeurs apparus tardivement sur ce marché, tels que Texas Instruments ou NTT, auront ainsi été les premiers à faire les frais d'un marché en réduction.

## TENDANCES

La période faste pour les machines dédiées est terminée depuis la fin des années 80. En effet, l'avènement de stations de travail a rendu courants :

- des vitesses de traitement très rapides, notamment avec les nouveaux processeurs RISC, c'est-à-dire à jeu d'instructions réduit et donc optimisé ;

- des environnements de programmation très sophistiqués bien que génériques. Le standard de multi-fenêtrage X-Windows, retombée du travail de pionnier des machines Lisp en est un exemple.

- des prix très attractifs, la production de masse aidant.

Le concept de machine dédiée est progressivement devenu moins pertinent. A quelques exceptions près comme l'introduction des

processeurs et terminaux X-Windows que l'on peut considérer comme des machines dédiées. Symbolics, le principal constructeur américain de machines Lisp et le dernier à rester en piste, a alors abandonné le développement de machines autonomes et s'est spécialisé dans des processeurs spécialisés[4] à adjoindre à une station de travail conventionnelle. C'est d'ailleurs, comme nous allons le voir, la nouvelle approche également suivie chez NTT. NTT, qui a récemment racheté Symbolics, est donc de fait le seul à contrôler le marché des machines Lisp. Bien qu'il soit quasi-inexistant, ce marché est potentiellement porteur à moyen terme.

## SILENT

Nous allons maintenant présenter le développement des machines Lisp chez NTT. La machine prototype actuelle, appelée SILENT est en cours de fabrication. Elle succède à la première machine Lisp conçue par l'équipe de I. Takeuchi : ELIS.

### ELIS

Ikuo Takeuchi a dirigé le développement de machines Lisp aux laboratoires NTT depuis de nombreuse années. ELIS a été développée au début des années 80. Son langage de programmation se nomme TAO. Le langage TAO était présenté comme une unification de Lisp, Prolog et Smalltalk résumée par la somme harmonique :

Pour être plus précis TAO était un langage fonctionnel de type Lisp qui intégrait les principes de la programmation logique et de la programmation par objets. Cette machine a été commercialisée par NTT vers 1985, sans succès. Malgré son échec commercial, NTT n'a pas abandonné le projet mais au contraire soutient une deuxième mouture.

### SILENT

La machine successeur de ELIS se nomme SILENT. Ikuo Takeuchi propose lui-même deux explications au nom choisi. C'est une référence/hommage au livre «TAO is Silent» du philosophe Raymond Smullyan, introduction «lumineuse» au TAO (d'après Takeuchi). De plus, une fois inversé, SILENT, devient un acronyme de The New ELIS.

Cette nouvelle machine est destinée à ne pas être autonome, mais sera utilisée comme unité de calcul symbolique spécialisée, associée, via une interface SCSI[8], à une station de travail classique comme par exemple une station Sparc de Sun. Ainsi il n'est plus question comme pour ELIS ou les premières machines Lisp de reconstruire un environnement complet doté de multi-fenêtrage et d'une gestion saine de fichiers mais de déléguer ces fonctionnalités à la station de travail hôte et à son système d'exploitation.

### CARACTÉRITIQUES DU PROCESSEUR

Donnons quelques caractéritiques du processeur telles qu'elles ont été précisées par Takeuchi. Le processeur spécialisé de SILENT, conçu par M. Yoshida, est de type CISC et utilise la technologie CMOS 0,7 micron gate array (100k gates). SILENT est micro-programmable. La taille mémoire est de 160 Mo. Un mot mémoire fait 40 bits, dont 8 bits de tag. Le cycle est de 30 nano-secondes. Le processeur est en cours de fabrication et devait être disponible au mois d'Avril 1993. Il sera inséré dans la carte mère, de SILENT, que nous avons pu voir.

### APPLICATIONS

SILENT est capable de traiter une requête temps réel sous 100 micro-secondes. Les applications envisagées sont des applications temps réel[9] de haut niveau : contrôleur ou routeur de réseau de communications intelligent, robotique, infographie. Ainsi, concernant ce dernier aspect, constatant que la programmation d'applications de synthèse/traitement d'images devient de plus en plus sophistiquée, I. Takeuchi considère qu'elle bénéficiera pleinement d'un processeur symbolique spécialisé.

SILENT est destinée à être connectée notamment à une autre machine dédiée au traitement d'images et développée à NTT. Il s'agit de la machine TARAI, composée de 3 processeurs (3 étant un nombre "idéal" relatif aux 3 dimensions graphiques et aux 6 degrés de liberté utilisés en général en robotique, selon I. Takeuchi). La réalisation de TARAI est prévue vers la mi-1994.

## TAO

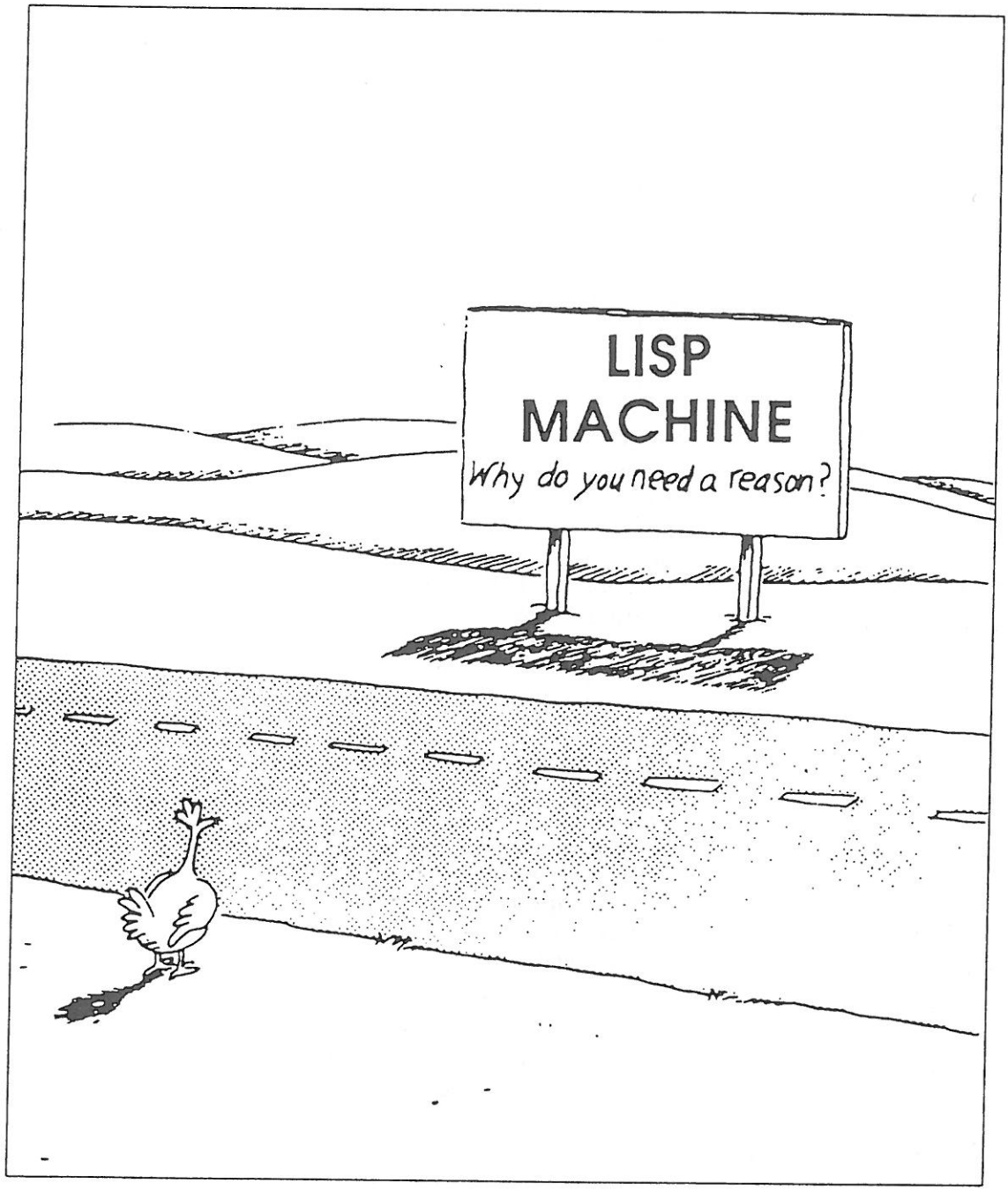
TAO est le langage machine de SILENT. Chaque primitive possède un temps d'exécution déclaré dans le manuel. Ainsi par exemple, l'opération "car"[10] s'effectue en 3 cycles et donc 90 nano-secondes, incluant un contrôle de type. Le vitesse d'exécution tombe à 500 nano-secondes si la donnée n'est pas présente dans la mémoire cache.

TAO est un dialecte de Lisp, très proche de Common Lisp, le standard de fait de Lisp. Comparé au premier TAO de la machine ELIS, le TAO nouveau est présenté par Takeuchi comme améliorant l'intégration des 3 types de programmation : fonctionnelle, logique et à objets. TAO n'est pas défini comme un langage de programmation directement à l'usage de l'utilisateur. Les primitives de gestion des objets et de la concurrence en particulier sont d'assez bas-niveau. L'objectif est d'offrir au concepteur de langages les primitives de base pour pouvoir construire des langages de niveau plus élevé.

### OBJETS

Ainsi en particulier, TAO offre des objets très primitifs, en fait constitués d'environnements lexicaux (au sens de Lisp lexical, tel que Scheme). Les primitives d'auto-référence, d'écriture dans les environnements, de définition d'opérations privées à l'objet et enfin de gestion d'erreur en cas d'invocation d'une opération

absente sont proposées mais pas plus. Il n'y a donc pas de classes ni d'héritage. Cependant on peut construire de tels mécanismes de plus haut niveau à l'aide de ces objets et des mécanismes primitifs d'invocation et de délégation. Signalons que Takeuchi n'envisage pas de développer lui-même une telle extension, et notamment pas d'implémentation du standard de fait en ce qui concerne les couches objets en Lisp, c'est-à-dire CLOS[11] (Common Lisp Object System).




## LOGIQUE

TAO offre également le nécessaire pour une programmation de type Prolog. Le préfixe "_" (souligné) permet de retourner la position (mémoire) d'une expression, en plus de sa valeur. Ceci permet d'implémenter les variables logiques. Un mécanisme de retour en arrière (*backtrack*) est également fourni. La syntaxe des crochets permet de faire la distinction entre appel fonctionnel "(...)", démonstration de but logique "{...}", ou transmission de message à objet "[...]".

## CONCURRENCE

SILENT est une machine mono-utilisateur mais multi-tâche. TAO intègre donc des mécanismes de concurrence : processus (activités), sémaphores (primitives de synchronisation), boîtes aux lettres (primitives de communication), mémoires tampons (comme les boîtes aux lettres mais de taille bornée). Ces primitives sont de facture très classique, et de relativement bas niveau. Signalons par contre qu'un processus possède une queue de tâches. En ce sens un processus TAO est relativement évolué, proche d'un objet ou/et d'un séquenceur,

puisqu'il exécute successivement plusieurs tâches communiquées. Chaque processus possède également une queue d'évènements. Ceci permet donc le traitement d'interruptions.

Le temps de basculement (*context switching*) d'un processus à un autre est de 5 micro-secondes. L'ensemble des processus partage une pile circulaire (!) de 128 Kmots. La pile d'exécution contient l'ensemble des piles propres à chaque processus. Selon I. Takeuchi, la limite supérieure du nombre de processus est 50. Cette limitation indique que le grain de concurrence visé reste relativement important, et donc ne peut couvrir les programmes concurrents à grain très fin comme la mise en œuvre d'un éventuel langage d'acteurs sur TAO.

### GESTION DE LA MÉMOIRE

Il existe un pseudo-processus ayant un statut particulier. Il s'agit du récupérateur de mémoire (*garbage collector*) nécessaire dans les systèmes à gestion implicite de la mémoire, notamment Lisp qui en a d'ailleurs été l'inventeur. L'algorithme choisi reste de facture classique (*mark and sweep*).

## CONCLUSION

La machine SILENT et son langage de programmation TAO représentent un travail de qualité produit par une équipe réduite. La machine annonce des performances intéressantes sinon époustouflantes. La facture reste très classique, ce qui souligne le côté artisanal et autarcique de ce développement. Ainsi on peut remarquer que des résultats récents sur des algorithmes récupérateurs de mémoire optimisés pour le temps-réel ou des standard de mécanismes à objets comme CLOS n'ont pas été envisagés. L'aspect quelque peu idiosyncrasique de la syntaxe, notamment les notations "condensées" pour affectations dans les objets à base de "!", confirme cette impression de relative autarcie.

Cette constatation ne doit pas masquer une évidence : l'équipe de I. Takeuchi acquiert depuis maintenant plus de dix ans de l'expertise dans le domaine de la conception de machine numérico-symbolique dont les applications à moyen terme sont nombreuses, e.g. infographie, robotique, réalité virtuelle... Il est intéressant de constater que NTT continue à soutenir l'équipe de Takeuchi, à notre avis sans optique de commercialisation à court terme. La commercialisation a déjà été tentée une première fois au milieu des années 80 avec la première machine ELIS, mais a abouti à un échec, le marché s'étant rétréci fortement. Donc, le projet SILENT s'intègre dans le cadre d'une politique de positionnement à long terme. La technologie et l'équipe qui l'entretient et la développe restent ainsi activées et soutenues dans l'optique de retombées quasi-certaines mais d'échéances inconnues. De plus cette technologie n'est maintenant plus connue que de NTT, en acquérant Symbolics, le premier et dernier fabricant spécialisé de machine Lisp, il est vrai beaucoup plus petit que NTT. Ceci illustre clairement la stratégie Japonaise d'investissement à long terme, que l'on peut opposer notamment à l'optique de rentabilité à court terme des Etats-Unis.

Nous tenons à remercier Nicolas RAGUIDEAU, du *Switching Communication Labs*, qui a organisé notre visite. Nous remercions tout spécialement Ikuo TAKEUCHI qui nous a accueilli très chaleureusement et longuement.

❑

1 La fonction récursive "tak" est une des fonctions classiques d'évaluation de performances (benchmarks) de systèmes Lisp.

2 Takeuchi signifie littéralement le cœur du roseau en japonais. I. Takeuchi possède des cartes de visite sur la face japonaise desquelles son nom n'apparaît pas : il le calligraphie pour chacun de ses visiteurs.

3 Massachusets Institute of Technology, Boston, USA.

4 Processeur "Ivory"

5 Lisp est un langage de programmation symbolique.

6 Prolog est un langage de programmation logique.

7 Smalltalk est un langage de programmation à objets.

8 C'est I. Takeuchi lui-même qui a codé l'interface SCSI.

9 Le délai standard est plutôt autour de 20-40 msec. mais les applications envisagées par NTT sont telles que ce délai de 100 msec. semble suffisant.

10 Accès au premier élément d'un doublet.

11 Voir l'ouvrage de référence de G. Steele.