El Cau del Hàcker

Enric Plaza

La javamania

Els qui s'ocupen, per feina o per vocació, de la IA solen ser discutidors de mena, i un tels temes que més sobretaules de congressos ha ocupat ha estat la discussió sobre les avantatges de Lisp sobre Prolog—o viceversa. Aquesta ha esta una discussió interminable que tenia com a contrapunt els comentaris sobre la desgraciada necessitat de usar C i C++ "per exigències del guió" i també totes les variacions de llenguatges funcionals "purs" i lògics "extesos" que s'han arribat a dissenyar. Vet aquí que compareix **Java**, i el món mai més no serà igual.

Quan el mot *agent* va començar a dringar fort, ara fa un parell d'anys, semblava que el llenguatge **Telescript** era el que millor s'avenia a les noves idees de *programar Internet*: els agents com a codi mòbil. Tot i ser un disseny propietat de de General Magic Inc., els altres llenguatges per Internet, encara que oberts i no propietari, semblaven més acadèmics que industrials. La compareixença de Sun Microsystems proposant Java com a llenguatge obert i no propietari va canviar les coses. Després de dos anys, el disseny de Java, el suport de Sun, i una estratègia d'expansió ben pensada, han convertit Java en un llenguatge de programació amb què cal comptar: no només té un suport industrial gran i en augment, també la comunitat acadèmica s'hi ha llençat de ple amb un munt de projectes on intervé Java (i sobretot la *màquina virtual Java*).

Java tot curt:
dinàmic, multifilat, eficient, portable, segur, robust, interpretat, distribuït, orientat-objecte, arquitectura lment neutral i simple.

Per tal d'esbrinar si Java pot ser un llenguatge important dins la IA cal veure les característiques lingüístiques que l'han fet interessant per diverses comunitats i la mena d'aplicacions per les quals s'està usant. Un punt a favor de Java, assenyalat per diferents informàtics, és el de ser el primer llenguatge industrial dissenyat tenint en compte moltes de les porpostes modernes de llenguatges d'alt nivell—però, ai las, no pas totes! la eficiència ha obligat ha deixar fora força propostes. De fet, Apple ja havia fet una aposta semblant amd el desenvolupament de **Dylan**, un *DYnamic LANguage*[1] per desenvolupar aplicacions i eficient que recollia les característiques de Lisp i Scheme (incloent-hi funcions d'ordre superior i orientació-objecte, recollida de brossa i polimorfisme) compatibles amb una compilació en codi eficient. Fins i tot es va dissenyar per Dylan una sintaxi pseudo-C (els parèntesis són tabú) per atreure els programadors industrials avesats a l'excés de sintaxi. Java segueix la mateixa estratègia, per bé que decideix altres propostes: orientació-objecte pura, l'herència no és múltiple però admet un fort tipat usant interfícies, no admet funcions d'ordre superior però és multifilat.

Tanmateix, el tret per mi principal i més innovador de Java, que el fa molt millor que Dylan, és la proposta d'assolir la distribució (i la independència respecte de l'arquitectura) mitjançant una màquina virtual de byte-codi. Les màquines virtuals han existit des de fa temps—Prolog i Smalltalk s'han implementat a bastament damunt les seves màquines virtuals. Des de fa anys, Scheme s'ha compilat específicament sobre una màquina virtual de byte-codi. La proposta de Sun és de fer de la màquina virtual Java (JVM) de byte-codi la plataforma que permet, sobre l'Internet Protocol, enviar codi (objectes Java compilats) a qualsevol lloc i sobre qualsevol plataforma que suporti la JVM. Cap altra proposta de llenguatge de programació per Internet, com Obliq[2], Agent-TCL, o Safe-Python[3], havia proposat fer servir una m'aquina virtual d'aquesta manera. A més la JVM i la recollida de brossa s'usa com a mecanisme de seguretat pel fet que el codi mòbil no té access directe al SO client.

La porposta Java+JVM ha creat molt enrenou en el món informàtic industrial, però també en les comunitats de llenguatges de programació. La actualització del SO Macintosh de 1997 inclourà la JVM. També inclourà a més d'OpenDoc, la tecnologia de components ara recolzada per Netscape i el consorci CORBA. Alguns fabricants de xips estan estudiant le possibilitat de que els seus processadors, ara molt optimitzats a C, donguin suport a les necessitats de Java. L'hegemonia d l'aliança Wintel és el focus de tots aquests arrengleraments de les indústries informàtiques.

---

[1] <http://www.cambridge.apple.com/dylan/dylan.html >

[2] <http://www.research.digital.com/SRC/Obliq/Obliq.html>

[3] <http://minsky.med.virginia.edu/sdm7g/Projects/Python/SafePython.html>

Màquina Virtual Java:
La JVM és un intèrpret que emula una CPU; les instruccions que executa generen crides al hardware del SO client; els programes Java es compilen a la JVM, que accepta 248 byte-codis; la JVM és una ALU apilada amb variables globals i locals.

LaJVM és el principal interés dels dissenyadors i implementadors de llenguatges. Donat que la JVM és disponible pels entorns Windows, Unix i Apple això cobreix la major part de les platafirmes instal.lades: qualsevol llenguatge que compili al byte-codi de JVM pot aprofitar el mateix mecanisme de distribució—a més de la independència de la plataforma.

Una altre possibilitat engrescadora és el xip RISC de Java dissenyat per Sun, l'especificació del qual, **picoJava**,és oberta i Sun llicenciarà també a fabricants externs. El disseny de picoJava té en compte les noves necessitats de Java: la recollida de brossa i la sincronització de fils d'execució. A més picoJava, en implementar directament les instruccions d'una JVM, és una arquitectura de pila: les variables s'emmagatzement localment a la pila, i les crides a mètodes i les operacions de byte-codi també passen les dades mitjançant la pila. La majoria de compiladors C tradueixen el codi font en un llenguatge de pila, però després cal que converteixin aquest llenguatge intermedi en codi RISC nadiu. Per tal com els RISC habituals només simulen la màquina de pila, els compiladors han d'analitzar el flux de dades i mantenir-ne les més essencials en els registres de la CPU. Aquesta estructura de regitres estàtics és eficient per aplicacions de càlcul científic, però és ineficient ineficient per codi amb crides a petits procediments que constantment engeguen i terminen. Segons Sun, el xip de Java, amb la pila i una "smart cache") pels registres de la pila, pot canviar les coses. De fet, el principal ús del xip Java (i també per dissenyar Java) és el de desenvolupar aplicacions *embedded*: sensors, PDAs, telèfons cel.lulars, equips tele-Web, i terminals d'Internet. El primer model del xip, **microJava**, es fabricarà el 1997 i costarà de $25 a $50. Més endevant, el model **ultraJava** costarà uns $100 i s'usarà en ordinadors personals—possiblement incorporarà descompressió JPEG i d'altres optimitzacions per gràfics presents ara als xips RISC UltraSparc de Sun.

Els programadors i implementadors de Lisp i d'altres llenguatges han discutit a bastament de Java en els *newsgroups* comp.lang.*. A comp.lang.lisp es torna a parlar de les màquines Lisp, i del perquè del seu fracàs, i del del xip de Lisp, tot en el context de la javamania. Triumfarà Java on Lisp fracassà? Perquè fracassaren les màquines Lisp? Són les condicions objectives (l'aliança de tots contra Miscorsoft i Intel) diferents? Després de molts anys encorsetats en un entorn C-Unix els Lispers poden tenir l'esperança d'un món (o cibermón) diferent? JavaSoft, la companyia de Sun que desenvolupa Java, ha contractat a Guy Steel Jr com a "distinguished engineer" per l'equip de desenvolupament de Java. Guy Steel Jr és un dels pares de CommonLisp.

El xip Java:
Sun Microsystems i Patriot Scientifics fabricaran xips amb un processador que implementi els byte-codis Java per equips ultralleugers simples i terminals internet.

No tot és xerrar, més aviat l'activitat ha estat frenètica. En Per Bothner[4] ha desenvolupat Kawa, un compilador de Scheme a la JVM escrit en Java. Per bé que **Kawa** no és un compilador optimitzador i que no permet `call/cc` és interessant com a demostració de viabilitat. A més, el problema més greu, la representació de les clausures (les funcions i el seu entorn lèxic), sembla que Java 1.1 resol el problema amb la introducció de les *inner classes* (la possibilitat de definir objects dins objectes, formalment anàloga a tenir entorns lèxics imbricats). **Jess** és un port a Java de CLIPS (una eina de desenvolupament de sistemes experts lliure i gratuita de la NASA[5] implementada en C) realitzat per Ernest Friedman-Hill[6] dels Sandia National Laboratories. També hi ha compiladors de CommonLisp a JVM i un grup de "funcionals purs" ha implementat **Pizza**, un superconjunt de Java que es tradueix (mitjançant macros) a Java i que admet funcions d'ordre superior i polimorfisme.

Finalment, però no per això menys important, el nom de *Java* és potser un dels més ben triats per promocionar un llenguatge de programació: és curt, evocador i natural. No crec que de mantenir l'antic nom, Oak, Sun haguès tingut aquest èxit.

---

[4] <http://www.cygnus.com/~bothner>

[5] <http://www.jsc.nasa.gov/~clips/CLIPS.html>

[6] <http://herzberg.ca.sandia.gov/jess>