LIMBAJUL CYC (CYCL) ŞI LIMBAJUL NATURAL

Mironela Pîrnău

mironela@yahoo.com

Universitatea creştină „Dimitrie Cantemir", Bucureşti

Rezumat: Informaţiile computerizate sunt stocate în multe forme, incluzând datele structurate (baze de date), semistructurate (foi de calcul, pagini WEB) şi nestructurate (fişiere de tip text sau câmpuri de text). Cyc poate transforma aceste informaţii în cunoştinţe utile. Cyc tratează fiecare înregistrare într-o bază de date ca şi cum ar fi fost o declaraţie implicită, făcută într-o bază de date. Aceste declaraţii sunt apoi disponibile în timpul procesului de deducţie. În mod similar, câmpurile de text pot fi citite folosind procesorul limbajului natural pentru a vedea dacă ele conţin declaraţii utile. Uneori declaraţiile descriu despre ce este vorba în „text". Cyc poate folosi aceste informaţii pentru a localiza şi raporta orice informaţie pe care utilizatorul ar putea să o folosească pentru a răspunde la un anumit interogatoriu.

Cuvinte cheie: limbajul CycL, baze de date, set de instrucţiuni, formule.

1. Introducere în limbajul CycL

CycL (limbajul reprezentativ Cyc) este o modalitate de reprezentare largă şi extraordinar de flexibilă a limbajului. Este, în mod esenţial, o extensie a analizei predicatelor (FOPC–first order predicate calculus) şi se poate ocupa şi cu egalităţile, deducţiile, skolemizarea şi unele caracteristici de rang secundar. Procesarea limbajului natural (Cyc NL) este una din cele mai studiate şi mai mari provocări ale ingineriei software. Multe echipe au încercat să producă sisteme bazate pe limbajul natural, capabile să citească şi să înţeleagă un simplu text. Sistemul Cyc NL are trei componente: lexiconul, analizatorul sintactic gramatical şi interpretul semantic ceea ce face posibil ca Cyc să poată analiza negaţii şi cuantificatori datorită interfeţelor care permit oamenilor să facă deducţii şi să interogheze Cyc folosind limbajul natural în locul limbajului CycL.

1.1. Semantic Integration Bus

Informaţiile computerizate sunt stocate în multe forme, incluzând datele care sunt structurate (baze de date), semistructurate (foi de calcul, pagini WEB) şi nestructurate (fişiere de tip text sau câmpuri de text). Cyc poate transforma aceste informaţii în cunoştinţe utile. Cyc tratează fiecare înregistrare într-o bază de date ca şi cum ar fi fost o declaraţie implicită, făcută într-o bază de date. Aceste declaraţii sunt apoi disponibile în timpul procesului de deducţie. În mod similar, câmpurile de text pot fi citite folosind procesorul limbajului natural pentru a vedea dacă ele conţin declaraţii utile. Uneori, declaraţiile descriu despre ce este vorba în „text". Cyc poate folosi aceste informaţii pentru a localiza şi raporta orice informaţie pe care utilizatorul ar putea să o folosească pentru a răspunde la un anumit interogatoriu.

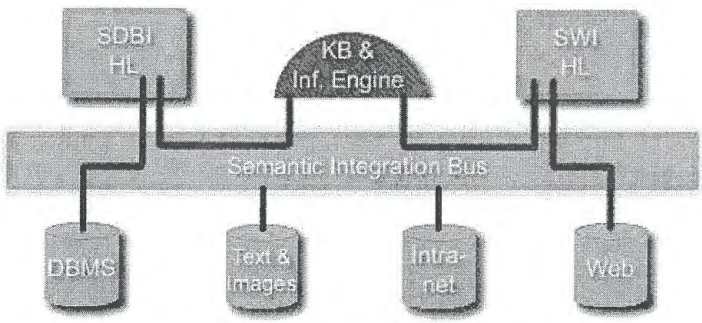

Figura 1. Semantic Integration Bus

În diagrama de mai sus, informaţiile stocate într-o bază de date sau pe web sunt făcute disponibile motorului de inferenţe ca declaraţii virtuale. Aceste seturi de declaraţii virtuale sunt administrate de module de niveluri euristice (heuristic level – HL). De exemplu, motorul de inferenţe (inference engine –

Inf. Engine) emite o transmisie pe Bus. Un modul HL recunoaşte că cererea are nevoie de o declaraţie care este stocată în spaţiul de cunoştinţe virtual. Modulul HL interceptează comanda, comunică cu baza de date (database – DB), site-ul web sau altă sursă de cunoştinţe, şi întoarce legăturile la motorul de inferenţe. Se continuă procesul de deducţie, combinând informaţii din surse multiple.

2. Setul de instrumente Cyc

Sistemul Cyc include, de asemenea, o varietate de instrumente de interfaţă, care permit utilizatorului să caute, să editeze sau să extindă baza de cunoştinţe Cyc pentru a pune întrebări motorului de inferenţe, şi pentru a interacţiona cu limbajul natural şi modulele de integrare a bazelor de date. Cel mai comun şi mai des utilizat instrument este browser-ul HTML, care permite utilizatorului să vizualizeze baza de cunoştinţe în mod hypertext. Paginile HTML, care descriu termenii Cyc, sunt generate din mers de sistemul Cyc. Fiecare pagină descrie un termen Cyc afişând toate declaraţiile în care este implicat termenul respectiv, organizat potrivit unei scheme standard. Fiecare utilizare a unui termen cyc este un link HTML la o pagină HTML care descrie acel termen, astfel încât navigarea în baza de cunoştinţe este uşoară urmărind o reţea de relaţii. Browser-ul HTML, de asemenea, include facilităţi pentru căutarea şi editarea bazei de cunoştinţe şi pentru punerea unei întrebări motorului de inferenţe. Alte instrumente de interfaţă HTML includ:

- un browser ierarhic care afişează orice subramură a unei structuri arborescente Cyc în format outline;
- un editor de lexicon, care asigură o modalitate de utilizare prietenoasă de a edita şi extinde lexiconul Cyc;
- un analizator semantic CycL, care permite utilizatorului să experimenteze cu facilităţile limbajului natural analizând arbitrar în şirurile de text;
- o interfaţă pentru baze de date, care asigură o interfaţă cu modulul de integrare semantică Cyc;
- un browser WordNet, care permite utilizatorilor să vizualizeze WordNet în relaţie cu ontologia Cyc;
- un generator care generează reguli în limbajul natural.

Majoritatea constantelor Cyc au un nume unic, cum ar fi #$Masa obiectelor sau #$Bill. Constantele Cyc primesc prefixul #$. Aceste caractere sunt, uneori, omise în documente care descriu limbajul CycL şi, de asemenea, pot fi omise de unele instrumente ale unor interfeţe. Numele constantei care urmează după #$ trebuie să îndeplinească următoarele reguli:

- toate numele de constante Cyc trebuie să aibă o lungime de cel puţin două caractere neincluzând caracterele prefixului #$;
- numele constantei poate include orice literă mare sau literă mică, orice cifră, şi simbolurile „-„ sau „_”, „?”; nici un alt caracter cum ar fi „!”, „&”, sau „@” nu sunt permise;
- numele constantelor Cyc ţin cont de literele mici şi literele mari: #$Foo nu este acelaşi lucru cu #$foo; cu toate acestea este interzisă de sistem distingerea a două constante doar pe baza literei mari sau mici;
- toate numele predicatelor Cyc trebuie să înceapă cu literă mică;
- toate constantele nepredicat trebuie să înceapă cu literă mare; numele constantelor nepredicat pot, de asemenea, începe cu un caracter numeric (de exemplu, #$3Mcorporation); de asemenea, se pot începe constante de tip predicat cu caractere numerice, dacă cineva aduce un argument destul de convingător pentru alegerea acestei variante;
- toate constantele de tip Cyc trebuie să fie compuse din unul sau mai multe cuvinte care au înţeles logic într-o secvenţă, fără pauze cu excepţia liniuţelor sau sublinierilor (de exemplu: #$isa şi #$maşinisport); o secvenţă de caractere numerice poate fi considerată un cuvânt, de exemplu #$birouldinfaţădincorpul123; cu excepţiile menţionate mai sus pentru numele predicatelor, toate cuvintele (ce nu conţin caractere numerice) trebuie să înceapă cu literă mare; un acronim poate fi considerat cuvânt, dar toate caracterele sale vor fi de acelaşi fel (de exemplu: se va scrie cu litere mici dacă acronimul începe cu numele unei constante predicat şi cu litere mari dacă acronimul începe altfel.); liniuţele sunt folosite pentru a despărţi părţi ale numelor care restricţionează sau redefinesc înţelesul numelor, de exemplu #$Fruct - Cuvântul.

Înţelesul unei constante în limbajul CycL este determinat de declaraţiile făcute în baza de cunoştinţe. În mod convenţional, alegem nume pentru constantele Cyc ce au un înţeles pentru utilizator (de exemplu: #$Roşu sau #$Culoarearoşie), dar Cyc nu înţelege aceste şiruri de caractere. Baza de cunoştinţe Cyc este

constituită dintr-un număr mare de declarații. Atunci când o formulă este introdusă cu succes în baza de cunoștințe, este stocată ca una din acestea. Fiecare declarație este compusă dintr-un număr de elemente:

- o formulă;
- o microteorie;
- o valoare de adevăr;
- o direcție (sau nivel de acces);
- un sprijin.

Formulele sunt formulele clasice Cyc, folosite pentru a declara fapte în baza de cunoștințe Cyc. Fiecare declarație este conținută într-o microteorie. O anumită formulă poate fi declarată într-una sau mai multe microteorii; în acest caz, vor exista declarații care au acea formulă în fiecare din aceste microteorii. De unde provin informațiile pentru declarații? Acest lucru depinde de originea declarației. Dacă o declarație este adăugată la baza de cunoștințe de către motorul de inferență ca rezultat al executării unei reguli, codul motorului de inferență decide ce microteorie concluzie ar trebui adăugată și o înregistrează. Dacă o declarație este rezultatul unei persoane sau a unui program extern de declarare a unei formule în baza de cunoștințe, la acel moment declaratorul trebuie să specifice în ce microteorie trebuie introdusă formula. Unele interfețe pentru introducerea cunoștințelor nu au nevoie ca utilizatorul să specifice o microteorie pentru noi declarații, și atunci se va încerca găsirea uneia potrivite fie se va folosi #$BaseKB ca default.

2.1. Valorile de adevăr

Fiecărei declarații îi este atașată o valoare de adevăr care îi indică gradul de adevăr. Limbajul CycL conține cinci posibile valori de adevăr, din care cele mai comune sunt: „default true" și „monotonically true".

Declarațiile care sunt „monotonically true" sunt declarate ca fiind adevărate în orice situație. În cazul unei declarații „monotonically true" cu variabile universale în formulă, dacă un obiect este găsit pentru care declarația nu este adevărată, este semnalată o eroare. În cazul unei declarații de bază, care este „monotonically true", dacă negația acelei formule este declarată în timpul inferării (în cadrul aceleiași microteorii) este semnalată o eroare. Declarațiile care sunt „default true", în mod contrastant sunt adevărate în majoritatea cazurilor și pot fi rescrise. Dacă negația unei declarații de bază este declarată în aceeași microteorie în timpul inferenței nu este semnalată nici o eroare.

2.2. Direcțiile (nivelurile de acces)

Direcția este o valoare asociată cu fiecare declarație care determină când să fie executat procesul de deducție care implică declarația respectivă. Sunt trei valori posibile pentru direcții: înainte, înapoi și cod. Procesul de deducție care implică declarații cu direcția înainte este executat la momentul declarării (adică atunci când este adăugată o nouă declarație bazei de cunoștințe), în timp ce inferențierea pentru declarații cu direcția înapoi este amânată până când se produce un interogatoriu care permite inferențierea înapoi. Modulele HL au fost scrise pentru a suplini nevoia de inferare folosind declarațiile. Declarațiile cod nu pot fi editate prin interfața HTML. O modalitate de a vizualiza direcțiile ca o ierarhie este:

- :cod;
- variabilele declarate :cod sunt suportate direct de modulele HL;
- :înainte (forward);
- variabilele declarate înainte sunt folosite în procesul de deducție înainte. Toate declarațiile :cod sunt folosite implicit la acest nivel pentru că modulele HL care le suplinesc sunt folosite aici;
- :înapoi (backward);
- variabilele declarate înapoi sunt folosite numai în inferarea înapoi; toate declarațiile :înainte sunt, de asemenea, folosite la acest nivel și modulele HL suplinind declarațiile :cod sunt din nou folosite la acest nivel.

Fiecărei declarații îi este atașat un „sprijin", care constă din una sau mai multe justificări care formează sprijinul pentru prezența declarației în baza de cunoștințe. În multe cazuri, cel puțin una din justificările sprijinite este locală, indicând că declarația a fost adăugată la baza de cunoștințe dintr-o sursă exterioară (de cele mai multe ori de un factor uman). În alte cazuri, o justificare sprijinită este o sursă care indică faptul că declarația a fost inferată și care subliniază pasul final al unui argument sau strategie de

inferare, care sprijină declarația. Sprijinul este un element al declarației, care nu trebuie specificat de factorul care introduce cunoștințele în momentul introducerii lor. Este creat și updatat automat.

3. Concluzii

Fundamentarea limbajului Cyc a fost lungă și anevoioasă, dar a început să dea roade. Începută ca proiect în 1984, Cyc este acum o tehnologie de lucru cu aplicații în multe domenii ale lumii reale. Baza vastă de cunoștințe a Cyc permite executarea de sarcini care sunt cu mult peste capabilitățile altor tehnologii software.

Bibliografie

1. **MARCUS, M.P.**: A Theory of Syntactic Recognition for Natural Language, Cambridge, 1980.
2. **MCGRAW, K.L., K. HARBISON-BROGGS**: Knowledge Acquisition: Principles and Guidelines, Prentice Hall, Englewood-Cliffs, NJ., 1989.
3. **PARTRIDGE, D.**: Engineering Artificial Intelligence Software, Intellect, Oxford, 1992.
4. **SCHUMACHER, M.**: Objective Coordination in Multi-Agent System Engineering, Design and Implementation, Lecture Notes in Computer Science, , Springer - Verlag, 2001, Vol. 2039.
5. **TEODORESCU, H.N., D. MLYNEK, A. KENDEL, H. J. ZIMMERMANN**: Inteligent Systems and Interfaces, Kluwer Academic Publishers, Boston, 2000.
6. [6] http://www.cyc.com