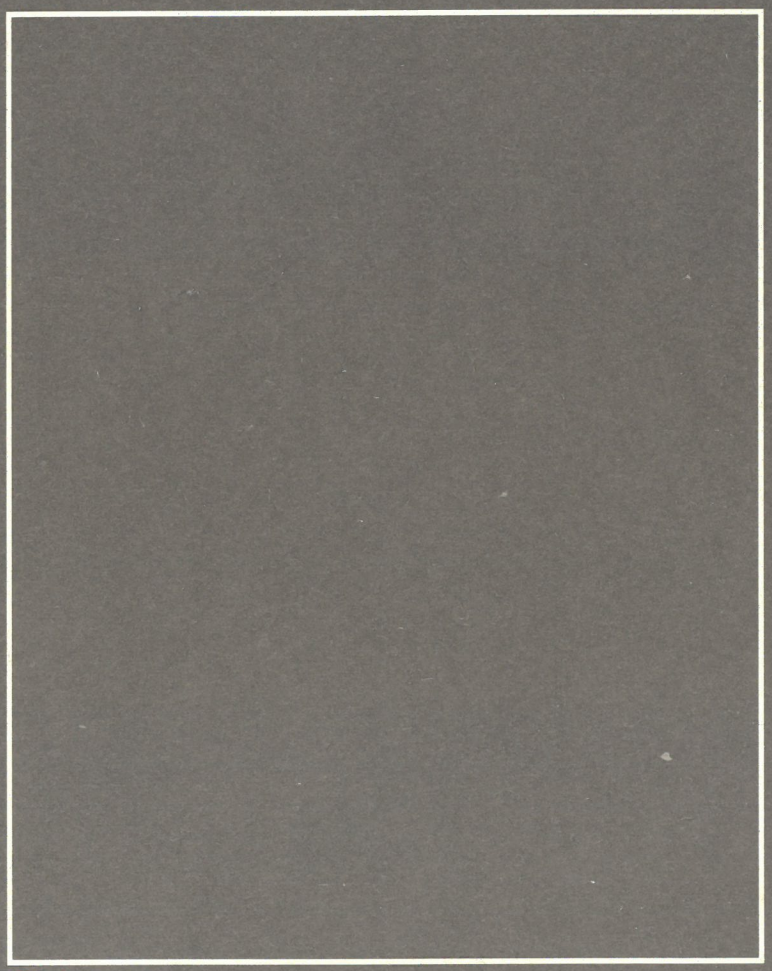
UNIVERSITA' DEGLI STUDI
DI MILANO
DIPARTIMENTO DI SCIENZE
DELL'INFORMAZIONE

BULL ITALIA
CENTRO STUDI
INFORMATICA
E AUTOMAZIONE

Note di Software



Anno 1989
44/45
Giugno - Ottobre

Note di software è una pubblicazione trimestrale, edita a cura del Dipartimento di Scienze dell' Informazione dell' Università di Milano e dal Centro Studi Informatica e Automazione della Bull Italia.

Note di software intende costituire uno strumento per la circolazione di informazioni, idee, metodologie ed esperienze nell'area delle tecnologie e della produzione del software.

La collaborazione a **Note di software** è libera. Chi desiderasse pubblicare un contributo è pregato di prendere contatto con il Direttore responsabile o con il Comitato di Redazione. In particolare si sollecitano contributi nella forma di monografie, da inviare in triplice copia, e che non dovrebbero superare le cinquemila parole.
I lavori ricevuti per la pubblicazione verranno revisionati dai membri del Comitato di Revisione Scientifica, che possono avvalersi della collaborazione di specialisti del settore. I lavori pubblicati riflettono comunque le opinioni dei loro autori.

Note di software viene distribuito ad Aziende, Enti e specialisti del settore che ne facciano richiesta alla Redazione.

Direttore Responsabile: Franco Filippazzi
Bull Italia, Centro Studi Informatica e Automazione, via Vida 11 - 20127 Milano

Comitato di Redazione: Stefania Bandini, Francesco Gardin, Roberto Polillo
Università di Milano, Dipartimento di Scienze dell' Informazione
via Moretto da Brescia, 9 - 20133 Milano, tel. (02) 7575233

Comitato di Revisione Scientifica: Alberto Bertoni, Gianluigi Castelli, Giovanni Degli Antoni, Giorgio De Michelis, Mario Italiani, Gaetano Aurelio Lanzarone, Giancarlo Martella, Marco Maiocchi, Giancarlo Mauri.

Autorizzazione del Tribunale di
Milano n. 87 del 23 febbraio 1977

Supporto tecnico - gestionale e distribuzione: Domenico Asinelli
Bull Italia, via Tazzoli 6
20154 Milano, Tel. 02 - 67792960

UNIVERSITA' DEGLI STUDI DI MILANO
DIPARTIMENTO DI SCIENZE DELL'INFORMAZIONE

BULL ITALIA
CENTRO STUDI INFORMATICA E AUTOMAZIONE

QUESTO NUMERO 1

- ORGANIZZAZIONE DELLA CONOSCENZA
  Gianni Degli Antoni 3

- HYPERTEXTUAL INTERFACE FOR AN OBJECT ORIENTED PROGRAMMING ENVIRONMENT
  Roberto Grande 7

- DIMOSTRAZIONE AUTOMATICA IN GEOMETRIA
  M. A. Alberti, M. Torelli 17

- HEURISTICS TO LOCATE THE BEST DOCUMENT SET IN INFORMATION RETRIEVAL SYSTEMS
  Dario Luccarella 28

- A KNOWLEDGE - BASED APPROACH TO AN OPERATING SYSTEM FOR A LARGE COMPUTER SYSTEM
  J. Takamura, K. Nobusawa, Y; Ebino, A; Date, E. Yoshikawa 36

- COMBINING THREE CONCEPTUAL MODELS: ABSTRACT DATA TIPES, LOGIC PROGRAMMING AND DATABASE
  Scott N. Woodfield, Jeannette M. Weston, David W. Empley 47

AVVENIMENTI

- NEURAL NETWORKS: COMMERCIAL PROSPECTS
  Paolo Stofella 56

- UMANISTI E INFORMATICA: UNA PANORAMICA DELLA CONFERENZA INTERNAZIONALE "THE DYNAMIC TEXT"
  Paolo Fezzi 58

- SISTEMI DI SUPPORTO ALLE DECISIONI: METODOLOGIE E STRUMENTI
  Giulio Falco 67

- SISTEMI ESPERTI IN BANCA E NELLE ASSICURAZIONI
  Roberta Gulden 70

QUESTO NUMERO

Nell'articolo di apertura, Gianni Degli Antoni presenta incisive riflessioni su un problema di fondo che pervade ogni ambito delle attività umane: la documentazione. Il computer è, fuori di ogni dubbio, lo strumento decisivo per la soluzione di questo problema. Ma ciò presuppone una razionalizzazione di tutto il processo documentario, secondo modelli funzionali agli obbiettivi perseguiti. In questo quadro assumono ruolo essenziale i concetti e le tecniche di organizzazione della conoscenza.

A questa tematica si può collegare l'articolo successivo, in cui Roberto Grande descrive una interfaccia ipertestuale per la realizzazione di un ambiente di sviluppo software object oriented. Il connubio tra tecniche ipertestuali e programmazione per oggetti crea un contesto operativo che favorisce la creatività del progettista.

Nel terzo articolo M. A. Alberti, E. Calzonari e M. Torelli riassumono i principali approcci alla dimostrazione automatica di teoremi in geometria. Vengono esaminati, in particolare, i metodi algebrici nel cui ambito gli Autori hanno effettuato verifiche e sperimentazioni dirette.

Dario Luccarella, nell'articolo successivo, esamina l'uso di strategie euristiche per localizzare "al meglio" i documenti cercati in un sistema di information retrieval. L'Aurore propone uno specifico algoritmo che ottimizza il processo di ricerca.

Il quarto articolo ha a che fare con un tema molto complesso quale è quello dei sistemi operativi di grandi mainframes. Il problema è stato affrontato alla NEC, in Giappone, con un approccio knowledge - based: i concetti fondamentali e i risultati conseguiti e vengono qui rappresentati da J. Takamura e colleghi.

Il sesto articolo si collega invece alle radici concettuali dell'informatica. Woodfield e coautori esaminano tre paradigmi dominanti nella computer science, mettendone in luce differenze e affinità. L'intento è di consentire l'uso più appropriato di ciascuno di essi a seconda i casi.

Il fascicolo si chiude con le consuete Rubriche, che in questo numero risultano particolarmente corpose.
E' nostro intendimento, infatti, dare adeguato spazio a questa componente della rivista affinchè costituisca un utile e concreto servizio offerto al lettore.
In particolare, vorremmo dargli, in questo modo, la possibilità di "partecipare" a convegni, seminari ed avvenimenti, scelti tra quelli di maggiore rilevanza e attualità.

LA REDAZIONE

ORGANIZZAZIONE DELLA CONOSCENZA

Giovanni Degli Antoni
Dipartimento di Scienze dell'Informazione
Università degli Studi di Milano

1. INTRODUZIONE

L' aumento della produttività nello sviluppo di documentazione e la comparsa di sempre più numerosi soggetti che producono documentazione sta portando alle sue estreme conseguenze: che la documentazione c'è e non si trova; ovvero c'è , ma è troppa e non si riesce a leggere; c'è e si trova, ma non contiene l'informazione desiderata; c'è ed è malfatta e quindi inutilizzabile.

La comparsa delle tecnologie ottiche sembra volere contribuire al cambiamento verso una migliore documentazione. In taluni ambienti, laddove la dimensione della documentazione aveva superato cifre da fantascenza, è iniziata la conversione verso l'impiego delle memorie ottiche.
Un forte impeto ad una più razionale documentazione viene ovunque dalla comparsa degli ipertesti e dei sistemi ipermerdiali che integrano la documentazione (multimediale poichè eventualmente costituita da porzioni di suoni, immagini e filmati opportunamente integrati) con una opportuna interfaccia ipertestuale che presentando sul video parti di documenti, visualizza con adeguate simbologie iconiche/simboliche la presenza di documenti associati a elementi del documento presente sul video. I simboli di associazione sono essi stessi a loro volta bottoni per accedere ai documenti associati.
La possibilità indicata sta determinando numerosi sviluppi verso la realizzazione di opportuni sistemi di documentazione con caratteristiche molto variabili che vanno dalla proposta di sistemi come XANADU orientato alla realizzazione di opportuni sistemi di documentazione globale (o semplicemente aziendale), ad altri più adatti ad essere impiegati per scopi non globali su un limitato sottoinsieme della documentazione di una organizzazione.

La dipossibilità nel mercato di sistemi per la raccolta integrata di documentazione dotati di interfaccia ipertestuali, impone la necessità di mettere a punto metodologie nei cicli di vita della realizzazione di sistemi di documentazione ipermediale proposti ed intende suggerire che la organizzazione della conoscenza contiene tutti gli elementi per indicare la metodologia per la realizzazione di sistemi ipermediali.

2. IL CICLO DI VITA DELLA REALIZZAZIONE DI SISTEMI IPERMEDIALI

La costruzione di sistemi ipermediali è certamente caratterizzabile in termini di ciclo di vita.
Le numerose esperienze in corso in molti laboratori sono d'altra parte così variabili negli obbiettivi e nei metodi che difficilmente si può proporre un ragionevole ciclo di vita accettabile da tutti per applicazioni pratiche.Tuttavia alcuni elementi del ciclo di vita sono certamente indipendenti dalla particolare modalità realizzativa e ad essi è facilmente associabile una serie di raccomandazioni.
La costruzione di un qualsiasi sistema, e quindi anche di un sistema ipermediale, parte dalla constatazione delle esigenze.

Una organizzazionze o un individuo possono soffrire,ad esempio, per le difficoltà di recupero ed impiego della documentazione. Ciò è molto comune e suggerisce di analizzare il costo sociale aziendale di tale sofferenza. Si scoprirà facilmente, anche in assenza di cifre, che il costo è enorme in termini di produttività individuale perduta, in termini di qualità delle attività (che dipende strettamente dalla disponibilità e dalla qualità della documentazione presente), in termini di costi logistici ed organizzativi. Non si può trascurare poi l'aspetto sociale della documentazione cartacea che certamente sta contribuendo alla distruzione di risorse pregiate (la cellulosa) ed all'inquinamento per produrre tali risorse.

E' ben vero che una certa (dis)organizzazione dei documenti è funzionale alla determinazione di strutture di comunicazione che proteggono in vari modi l'individuo o il gruppo da eccessive ingerenze esterne. Ciò è certamente importante e non può essere trascurato se si desidera contribuire al cambiamento nella modalità tecno-logica della documentazione. La comprensione della (dis)organizzazione è dunque il primo elemento da mettere a punto per la costruzione di un ciclo di vita orientato alla costruzione di sistemi di documentazione ipermediali.
L'affermazione ha una validità che va oltre il singolo progetto: solo la esperienza di tanti piccoli progetti potrà confermare le numerose tesi implicite nella analisi effettuata.
Immediatamente dopo, nel ciclo di vita analizzabile tuttavia in buona misura indipendentemente dal punto precedente, dovrà essere studiato quale uso si fa della documentazione.

L'analisi dell'uso della documentazione non può essere effettuata da chiunque: occorre una seria esperienza nel settore di attività in considerazione. L'uso della documentazione legale o tecnica è sempre accompagnato da una tradizione che si è costruita con il tempo e che non può essere trascurata. E' vero che con sistemi di information retrieval ciò può sembrare trascurabile. Ma non è così. La difficoltà di accesso ai documenti opportuni nei vari contesti di lavoro determineranno il successo del sistema proposto o faranno rimpiangere la vecchia carta. Ciò implica che il secondo elemento del ciclo di vita sia rigorosamente analizzato da persone competenti.
Dovrà venir anche raccolto (anche questa attività si può fare in parallelo alle precedenti) il patrimonio di documentazione nel modo più esteso possibile. Non dovrà mancare una fase in cui si confronta il secondo elemento del ciclo di vita (l'analisi dell'uso) con la documentazione raccolta: il confronto sarà certamente creativo poichè suggerirà come migliorare entrambi gli aspetti.

A questo punto occorrerà un progetto di massima del sistema ipermediale.
Tale progetto va suddiviso nell'archivio multimediale (testi ed immagini sono ormai la norma) e nella interfaccia di interrogazione. L'interfaccia dovrà essere rigorosamente adatta all'archivio. Il che comporta dunque la scelta dell'ambiente.

Scelto l'ambiente si dovrà procedere alla progettazione della interfaccia: in questa fase farà la sua comparsa la competenza dell'ambiente raccolta nel secondo elemento del ciclo di vita, nonchè alcuni aspetti del primo elemento sotto forma di gestione dei diritti di accesso e di tutte le conseguenze che la gestione degli accessi può comportare.
La costruzione di modelli cartacei potrà aiutare molto la discussione fra i partecipanti al progetto. Tali modelli cartacei (riguarderanno solo testi ed immagini, con rinvii esterni ad altre entità) conterranno piccole frazioni della documentazione, bottoni vari sotto forma di rimandi, le prime schematizzazioni della interfaccia, e commenti vari.

L'adozione di modelli facilita alquanto la comprensione delle architetture da adottare: alcune saranno già implementate nel sistema scelto. Altre andranno implementate sotto forma di contesti e bottoni oltre che di gerarchia dei rinvii.
Queste ultime scelte determineranno la struttura concettuale del sistema.

Dalla struttura concettuale e dalla semplicità con cui tale struttura viene comunicata (non spiegata) al suo utente dipenderà la semplicità d'uso. Le spiegazioni sulla struttura concettuale andranno accuratamente evitate nella interfaccia poichè inevitabilmente generano lo spostamento della attenzione a livelli più alti del sistema progettato. Per mantenere la attenzione dell'utente al livello adatto alla acquisizione della struttura concettuale questi dovrà capire chiaramente cosa potrà fare in ogni contesto anche di fronte ad una grande varietà di scelte.
Si arriva quindi alla realizzazione del prototipo. Il primo eviterà accuratamente memorie ottiche e l'intero volume della documentazione. Ma nello scheletro complessivo di tutte le funzionalità dovrà essere presente. L'interfaccia non sarà necessariamente completa ma le parti implementate, seppure in forma prototipale, dovranno essere scelte molto affidabilmente in modo da evitare rifacimenti ed in modo da poter attendere ulteriori specificazioni secondarie tali da non inficiare il disegno complessivo e da rendere possibile il completamento della progettazione di tutti gli elementi durante la preparazione del prototipo.
Questo dovrà essere validato da utenti reali che dovranno essere messi di fronte a pezzi realmente completi e non a pezzi parziali. Questi utenti dovranno essere rigorosamente e criticamente ascoltati nella rimessa a punto degli elementi già predisposti.
Il retso è ovvio. L'esperienza del cilco di vita di altri prodotti (in particolare dello sviluppo software) suggerirà molte considerazioni sulla impostazione del progetto, sulla sua implementazione, sulla sua valutazione, eccetera.

3. INTERFACCIA E CONOSCENZA

Un aspetto che abbiamo trattato solo in dettaglio nel paragrafo precedente riguarda la natura della interfaccia.
L'adozione di interfacce iconiche rende possibile non solo la semplificazione dell'impiego dei sistemi, ma se ben concepita, anche la semplificazione della progettazione dei sistemi di documentazione.

Infatti, se si utilizzano tecniche di documentazione che modellano in forma ionica (grafica) direttamente le entità di cui si tende ad effettuare la documentazione o anche solo l'ambiente in cui tali entità vivono, si ha immediatamente una linea guida per la costruzione degli aspetti grafici della interfaccia e una linea guida per associare gli archivi ed i documenti direttamente a disegni e simboli grafemici che compaiono sul video.
In queste ipotesi la predisposizione di un vero e proprio disegno della interfaccia è già un primo passo verso l'organizzazione della conoscenza.
Per chi non ha mai verificato le ipotesi di cui sopra il discorso potrà sembrare concettuoso: la verità è che la linea metodologica che l'atteggiamento indicato suggerisce è molto fecondo e la qualità dei risultati risulta essere notevole anche con piccolo sforzo. La ragione di tutto ciò è forse dovuta alla abitudine di chiunque al ragionare per modelli. Il ragionamento per modelli contribuisce alla distinzione in parti della documentazione: al modello come tutt'uno ed alle sue parti sarà facile associare la relativa documentazione con o senza una dimensione storica, temporale o casuale. Questi ultimi aspetti si possono facilmente realizzare con opportuni dialoghi che mettano in evidenza alcuni elementi del ciclo di vita degli elementi considerati.
Così, se si tratta di condensatori per un sistema, si dovrà poter chiedere le caratteristiche di un progetto, chi li ha acquisiti, chi li ha testati,chi li ha valutati rispetto alla affidabilità, quali eventi (rotture) sono intervenute, quali modifiche successive, con rientro nel ciclo di vita.

La stessa documentazione informale potrà essere facilmente associata a precisi contesti: così mentre i documenti certamente dovranno esser reperibili con le usuali tecniche di ritrovamento della documentazione, nel caso di interfacce iconiche a forte contenuto cognitivo i documenti saranno associati agli elementi del modello rappresentati sul video. Il browsing del sistema, diventerà un modo per ripassare la necessità di interventi dell'utente sulla realtà modellata a cui l'utente stesso ha direttamente accesso e di cui è responsabile.
Una questione molto interessante e per la quale attualmente manca qualsiasi esperienza e che potrà modificare completamente l'accesso alla documentazione è

l'impiego della animazione. Poichè nelle ipotesi suggerite si intende associare la documenatzione agli elementi del modello dei sistemi che si intende documentare, non sarà difficile immaginare delle sequenze animate che rappresentino graficamente il sistema di interesse. Ciò è possibile per tutte le strutture organizzate, per le tecnologie, per le biotecnologie, per la medicina, per le opere di ingegneria e, seppure con un pizzico di fantasia, per le strutture matematiche.

La visualizzazione di sequenze (o processi) animati apre molte strade al brusing se si suppone che l'arresto della sequenza animata ad un certo istante permetta immediatamente l'accesso alla documentazione relativa alla parte di processo visualizzata all'istante di arresto della animazione. Si tratta dunque di una moviola concettuale per l'accesso alla documentazione.

Ciò comporta una chiara integrazione della documentazione con le fasi della animazione, un esplicita e chiara associazione della fase della animazione alla struttura concettuale del modello che si rappresenta ed infine l'associazione fra documentazione e modello.
Si può facilmente intendere per quale ragione sia stato impiegato constantemente il termine documentazione nella presente nota. L'idea di fondo è che la conoscenza sia l'associazionedi documentazione (formale o informale realmente irrilevante) agli elementi del modello a cui si riferiscono.

La comparsa dei sistemi ipermediali diventa quindi un serio approfondimento di cosa sia la conoscenza in tutte le sue versioni e la comprensione che con Mc Luhan il MEDIA E' IL PROCESSO.

In tale ordine di idee nessuna forma di documentazione può essere privilegiata. Certo che al momento della esecuzione con computer la documentazione dovrà essere accuratamente definita e quindi formale. Non sarà sfuggito al lettoreil raccordo fra la ingegneria della conoscenza e i temi discussi nella presente nota che intende essere un succinto insieme di suggerimenti volti a ripensare il ruolo della documentazione per la ricostruzione di più corrette ed agili strutture organizzate dove la conoscenza sia realmente lo strumento di gestione e non il supporto sd angherie prodotte al fine di difendere inutili e futili diritti di accesso ad un insieme confuso e disorganizzato di privilegi dovuti unicamente alla nostra incomprensione della organizzazione.

4. BIBLIOGRAFIA

[1] Ted Nelson, PROJECT XANADU, Literary Machines, 1987

[2] Giovanni Degli Antoni, ARTIFICIAL WORLDS, Atti del Convegno HUG, II° incontro, in corso di pubblicazione su Note di Software, N° 42/43 1989

[3] Giovanni Degli Antoni, IL COMPUTER E IL RAELE E L' ARTIFICIALE, Note di Software, N° 41, 1989

NOTE DI SOFTWARE 44/45

HYPERTEXTUAL INTERFACE FOR AN OBJECT ORIENTED PROGRAMMING ENVIRONMENT

Roberto Grande
Dipartimento di Scienze dell'Informazione
Università degli Studi di Milano

RIASSUNTO

In questo articolo sono presentati alcuni principi per la progettazione di una interfaccia di un ambiente di programmazione object-oriented, ed uno schema per l'applicazione della tecnologia degli Ipertesti alla realizzazione di tali principi. Come guida nella scelta di tali principi si è usata la metafora della Realtà Artificiale.

ABSTRACT

In this paper we present some design principles for an object oriented programming interface and a framework for applying the Hypertext technology to the realization of these principles. In pursuing this goal we are guided by the artificial reality metaphor.

1. INTRODUZIONE

The novel Hypermedia technology has generated a variety of environments designed to help people work with ideas in form of texts, images, schemata, project plans, programs, etc. They cope with new ideas, reducing the non-intellectual content of the creative process, as well with already consolidates ideas, providing support for their access, organization and management.

Like other ideas-intensive activities, software construction involves both creative and organizational issues. In this paper we address the problem of what kind of support a programming environment must provide to encompass the two previous aspects, borrowing some guidance from cognitive psychology and software engineering.

In particular we trust the artificial reality metaphor and the object-oriented design methodology as they provide adequate tools that extensively support modern software engineering method without neglecting psychological aspects of the programming process and the user interface.
Object-oriented design offers a language of description that serves as a bridge between the models in the human

This work was partly supported by Bull HN Italia

mind and those in computing systems (and as a means to organize such models), while the artificial reality metaphor provide a language of interaction that matches the human communication system to that of the computer.

The most promising media to support and integrate both this languages is offered by the hypermedia technology.
P.Garg and W.Scacchi, [9] point out the advantage of the combination of an hypertext system and software engineering tools, what they call a Software Hypertext System.
It provides aside the facilities for automated processing of information, the capabilities of hypertext systems for information management in large scale software engineering.

We agree with this vision and go further: not only hypertext systems are useful software management tools as they can be used for storing and retrieving informations, but also they are software development tools as they participate in the creative process of programming.

This paper is mainly concerned with the latter aspect. We don't deal with the whole software life cycle and with the project management activity. Indeed the focus of our attention is centered on the design and implementation phases.
Also we don't deal with a specific object oriented language or with a specific Hypertext System; we just present some good design principles for an object oriented programming interface and a framework for applying the Hypertext technology to the realization of these principles.

2. OBJECT TO THINK WITH

Human beings have developed a way of thinking about problems in which the mind automatically categorizes entities and builds a conceptual model. People attack new problems by seeing similarities and differences with old problems. Often new solutions can be produced by making slight modifications to old solutions; and similar solutions to several different problems can be generalized to a 'generic' solution.

The modules in which the conceptual model about a real world problem can be decomposed, usually include properties (data) and behavior associated with each module (operations). In principle, computer system should provide models that are compatible with those in the mind. They need a modelling paradigm to represent the richness of real world information.

Objects provide such a paradigm. An object 'encapsulate' data and the code that operates on that data in a single entity, thus reflecting our natural models of reality.
The object data model encourages people to design their application system using the natural concepts in the application's domain, rather than forcing them to translate application concepts into very different machine-oriented concepts.

3. OBJECT ORIENTED SOFTWARE ENVIRONMENTS

Object Oriented Software Environments (OOSE in short) provide a direct support for such a modeling style, both syntactical, by means of the programming language, and at run-time by means of the run-time environment.
Following B. Meyer [10] , object oriented programming is the "construction of software systems as structured collections of abstract data type implementations" realized by the 'class' construct.
Classes are modular units implementing meaningful abstractions; they describe uniform behavior and common properties of a set of objects.

From the programmer point of view an OOSE can be seen as a multi layered structure.

In the ***Class interface*** layer we define the public part of a class i.e. its interface, abstracting from implementation. In the interface we only specify operations and properties accessible to clients through that interface.

For each class there is an hidden layer in which we deal with the private part of a class i.e. the code for implementing operations (methods) in the class interface and the internal data structure. Herein we call this layer ***Class realization*** layer.

A class may inherit operations from 'superclasses' and may have its operations inherited by 'subclasses'. The ***Class Inheritance Specification*** layer is where we specify inheritance relationships. Subclasses inherit the interface as well as the implementation except for that part of code they explicitly redefine.

Class encapsulation and inheritance favors reusability: "encapsulation lets consumers assemble generic components directly into their applications, and inheritance lets them define new applications-specific components by inheriting most of the work from generic components in the library" [8].
Through the inheritance graph we can exploit communalities at all levels of abstraction saving coding effort and organizing the program structure in a natural modeling style (as cognitive psychologist found, this capability is useful in understanding concepts formation [2]).
Furthermore encapsulation and private/public separation in the class unit, promote modifiability: implementation changes in a class do not propagate to other class, owing to the information hiding mechanism.

In the program we only see class. Objects are run-time elements that will be created during a system's execution, as instances of classes. Object oriented computation consist in dynamically creating a number of objects (according to a pattern which is usually impossible to predict at compile time) that interact each other by sending 'messages' to perform operations.
The run-time support for such a computation is offered by what we call the ***Run-Time Environment*** layer. It can be seen as an abstract machine on top of the operating system, that takes care , among others, of memory management, garbage collection, object naming and persistence.

4. THE ARTIFICIAL REALITY METAPHOR

If a system must be entirely mastered by a single individual, it should be designed around a powerful metaphor that can be uniformly applied in all areas .
The object metaphor provide a means for mapping conceptual models in computational models and, ultimately, for mapping the reality, which our models are about, in the computer system.
Here we point out the fruitfulness of extending this metaphor to cover the interaction with computational models consistently with the way we interact with our mental models.

A programmer should be able to access to the software he is developing as to an artificial reproduction of the reality in his mind. We call this extended metaphor the 'Artificial Reality Metaphor' [5], [6].

Since all capability of a software environment is delivered through its user interface, we should design an interface that matches the human communication system to that of the computer. The general requirement that must be fulfilled is that every component accessible to the programmer should be able to present itself in a meaningful way for observation and manipulation.

5. HYPERTEXT OBJECT ORIENTED SOFTWARE ENVIRONMENT

The interface of a programming environment must deal with internal models (back end), external media (front end), and the interaction between these in both the human and the computer.
The Hypertext technology provide a powerful framework to support both back and front end with their interactions. We don't sustain a particular Hypertext system, neither propose a new one; herein the reader may refer to the HAM model [7] as it is general enough to underlie the subsequent argumentation. Again because of generality we chose the EIFFEL language [8] to expose a program example, but this choice is not mandatory.

5.1. The Back End

Back end is responsible for storing the program document in a collection of linked nodes in accordance with its relevant structure. Fundamental choices at this level are about type of nodes and topology of links; that's to say we must answer the question: to what extent are some program structures relevant for the access needs of a programmer?

5.2. Organizing Source Code

At a first view the source code is a sequential text file, but if we look at its implicit composition, the following

structures will appear:

- a sequence of nested blocks
- a parse tree [11]
- a call tree [1]
- a graph connecting reference statements-to definition statements for variables, (and other less remarkable structures), revealing its intimate hypertextual nature.

This is true for every program but is particularly meaningful for object oriented programs because, here, the main hypertextual structures are closely related to that of the conceptual model underlying the program.

5.3. Class Nodes

As we have seen, an object oriented source code must be viewed as a collection of class definitions.
The internal class structure is not flat; thus storing the whole class code in a single unstructured node could result in a lost of flexibility accessing it. This consideration suggests to implement the class node as a composite node with each subnodes storing a substructure. Here the relevant substructures are provided by the layered decomposition of classes previously introduced.
A possible implementations assign a subnode to the *class interface* and one to the *class realization* . We prefer this solution rather than attaching each method to a different node, because it fits the right level of granularity for the needs of both the programmer and the compiler (the natural unit of incrementality for an object oriented compiler is the class body). Furthermore this approach reduces the number of subnodes and the visual overhead for the programmer who, otherwise, could drown in a sea of windows (if we associate a subwindow to each subnode).

5.4. Link Types

There are four types of links that serve to organize source code:

- *seeMethod/links:* from *class interface* subnode to *class realization* subnode, both belonging to the same composite node. They are traversed acting on the operation names (buttons) listed in the interface and cause the access (for editing or visualization) to the corresponding methods in the *class realization* .

- *ClassReference/links:* from a *class realization* subnode to the composite node of another class. They are activated operating on the class name (button) referenced in a type declaration statement, and lead to the corresponding class node.

- *MessageReference/links:* from a *class realization* subnode to another *class realization* subnode (may be the same). Similar to the *seeMethod/links,* they allow the access to a method referenced in a 'message sending' statement like *object.message.*

- *Inheritance/links:* from a class composite node to another; they realize the relationship between a subclass and its superclass. The resulting graph completely describes the *Class Inheritance Specification* layer.

5.5. Organizing Documentation

A network of related heterogeneous informations illuminates the meaning of a program, especially in the development phase (when a professional programmer wants to see something like cross reference informations, storage maps, data/flow control/flow informations, accumulated timing informations, ..., all what a good compiler provides in a compilation protocol and that can be discharged when the program is completed). In this respect software and documentation should not be seen as distinct products.

We can individuate three kind of such informations:

- those produced by the programmer as informal synthetic commentary of his work; they address high level topics like: the structure of the inheritance graph, the meaning of a class interface, the overall implementation choices about a class,...

- those produced by the compiler in a compilation protocol

- those interleaved by the programmer in the source code as informal comment about single program statements or formal assertions (if available in the programming language), about properties of class operations

(routine pre and post condition, class invariant); assertions are a mean for documenting correctness arguments and may be monitored at run time by the compiler [10].

The first kind of informations will be placed in nodes edited by the programmer and linked to the appropriate nodes or subnodes.

The second kind of informations will be automatically placed by the compiler in default nodes linked to nodes that store the correspondent source code; they can be visualized and eventually edited for inserting comments.

The latter will be edited together with the source code and can be visualized along with the class code, or masked out.

To the *documentation /links* could be attached special attributes indicating the kind of informations they refer to, and some flag attributes used to filter out unwanted informations.

5.6. Run-time environment

This layer is represented in the hypertext as a single compound node. Dynamic images of all the internal structures that are affected at run-time and that should be monitored during debugging, are stored in its subnodes .

6. THE FRONT END

The front end must optimally deliver what back end stores offering to the user multiple approach to the program and integrating multiple tools from the underlying environment.
The integration of several capabilities will provide a substantial improvement to the creative process by eliminating the discontinuity between shifting from one tool to another and reducing the delay in the administrative steps of that process.

We will propose some good design principle for our hypertextual front end, resting on the Artificial Reality metaphor as an implicit guide.

6.1. Visualization and Direct Manipulation

A mere textual interaction with a program provide little leverage for managing the complexity of a large software system.
There are several good reasons for exploiting visual information richer than plane texts:

- visual content of conceptual models
- high bandwidth in communicating informations
- random access
- exploitation of the sensorial analogy guided by usefuls metaphors
- direct referencing by pointing
- compact coding through icons

Certain primitives for visualization are offered directly as elements of structured programming languages.
e.g.: Pascal like languages visualize:

- syntax by expressive reserved words (IF-THEN-ELSE)
- structure with bracketing construct(BEGIN-END), indentation and modularization concepts
- meaning of declared objects through identifiers of unlimited length

but highlighting structure by correct placing of brackets and choice of expressive identifiers is left to the discretion of the programmer. Furthermore the visual content of these primitive is a too poor support for direct manipulation.

The kind of visualization needed allows the programmer the correct association of visual entities to the design notions that, idealistically, had to be manipulated directly.
The correct associations is facilitated if the design notions used are adequately covered in what the cognitive psychologist call the mental model of the design task.
As we have seen, the design notions promoted by Object Oriented Programming provide an adequate coverage.

The strength of direct manipulation arise from providing:

- rapid, incremental and reversible actions
- feedback

- exploiting of the sensorial analogy

In the hypertextual front end icons, buttons (active icons), windows are the visual entities on which we can operate by direct manipulation.

We associate a different window to each layer of a class and to each documentation node. To the run-time layer are associated several subwindow (one for each sub-node), that can be organized in a suitable framework by the programmer, depending on the dynamic structure that he need to monitor.

Within the same window the programmer can chose among different layouts of the program or documentation text in accordance with its aesthetical preferences (e.g. indentation schemata, highlighting of reserved words or of user defined identifiers...).

6.2. Integrating Editor and Browser

If our attention have to be focused on the relevant structures of a program the syntax of the programming language should provide a guide during the editing phase. To a little extent this kind of facility is jet provided by structure-oriented program editors, allowing the programmer not to bother with missing END and semi-colons. Herein we propose to generalize that idea to structure of every level. So , for example, after editing the class interface window the programmer will find in the class realization window a skeleton, to fill in, presenting the header of each procedure listed in the interface.

The best way to realize the proposed generalization to higher level structure is by integrating the facilities of editor and browser in a single tool. This mean that in each step of the navigation process through a partially developed program we can inspect already entered structures or edit new structures with the mode proper to the abstraction level at which we are working.
Due to the compatibility of hypertextual and conceptual structure of an object-oriented program, navigating in the hypertext correspond to exploring the underlying conceptual model at several abstraction levels.

This approach is particularly valuable for object-oriented programming as it tend to blur the distinction between design and implementation. The only difference is the level of abstraction: during design certain details may be left unspecified, but in an implementation everything should be expressed in full. The integration of editor and browser allow the process of filling in details to be continuous from system architectures to working program.

In order to expose the main functionalities that should be provided by the editor/browser we consider how we can manipulate the fragment program listed in (fig.1). At the abstraction level of the inheritance graph, we associate a different icon to each class; classes of similar type are identified by similar icons with different names (fig.2).

```
class   POLYGON    export
    vertices, translate, rotate, perimeter....

feature
    ...
    vertices:   LINKED_LIST[POINT];
    traslate (a,b: REAL )    is
        do...end;
    ...
    perimeter:    REAL  is
        local
            this, previous:    POINT
        do
            from
                vertices.start;  this := vertices.value
            until
                vertices.islast
            loop
                previous := this;
                vertices.forth;
                this := vertices.value;
                Result := Result + this.distance(previous)
            end
            Result := Result + this.distance(vertices.first)

        end


class   RECTANGLE   export
    vertices, translate, rotate, perimeter, diagonal,
    side1, side2, ...
inherit
    POLYGON  redefine    perimeter
feature
    side1, side2 :   REAL;
    diagonal:   REAL;
    perimeter :  REAL   is
        do
            Result := 2* (side1 + side2)
        end
end
```

Fig.1

A new class can be entered by editing an icon with its name; by double clicking on this icon we zoom in its class interface window and can fill in a predefined skeleton by entering the exported operations identifiers (fig.3). Again double clicking on an operation identifier we open the class realization window (placing the cursor on the appropriate position inside the skeleton with headers of the operations to fill in) (fig.4) and connect, in the back end, the interface subnode with the realization subnode through a *seeMethod/link* .

Instead of further finish the definition of this class by implementing each operation, we can prefer to come back to the graph inheritance level with a simple backtracking command, and carry on the design leaving details unspecified.

If we suppose to have already implemented the POLYGON class we can easily define the subclass RECTANGLE by adding a new class icon at the inheritance graph level, and edit an arrow from RECTANGLE to POLYGON (fig.2). This will cause the automatic generation of the corresponding *Inheritance/link* in the back end.
Furthermore the appropriate **export** and **inherit** statement, will be automatically entered in the class interface and class realization windows.

The level of specification of each class (e.g. only the name, the interface, the implementation) is represented in the class inheritance graph by different color intensities of the icon, allowing to grasp on fly the program development state.

Double clicking on the class name occurring in a type definition or in an **inherit** statement we can reach the appropriate class definition following a *ClassReference/link* or an *Inheritance/link.* Similarly double clicking on the operation name that occurs in a message sending statement we traverse a *MessageReference/link* reaching the proper method definition. *Inheritance/links* can also be traversed starting from a superclass to its subclasses entering a 'Heirs' command.

To complete the picture of the editor/browser's functions we mention the capability of accessing all pieces of documentation available, depending on the class and the level in which we are working.

The potentiality of this editor/browser will be particularly useful if we have tools for importing and structuring in the back end, external libraries of reusable software. Reuse is a largely exploited technique in object-oriented design and must be supported by good documentation and good database searching tools.

6.3. Integrating Editor and Compiler

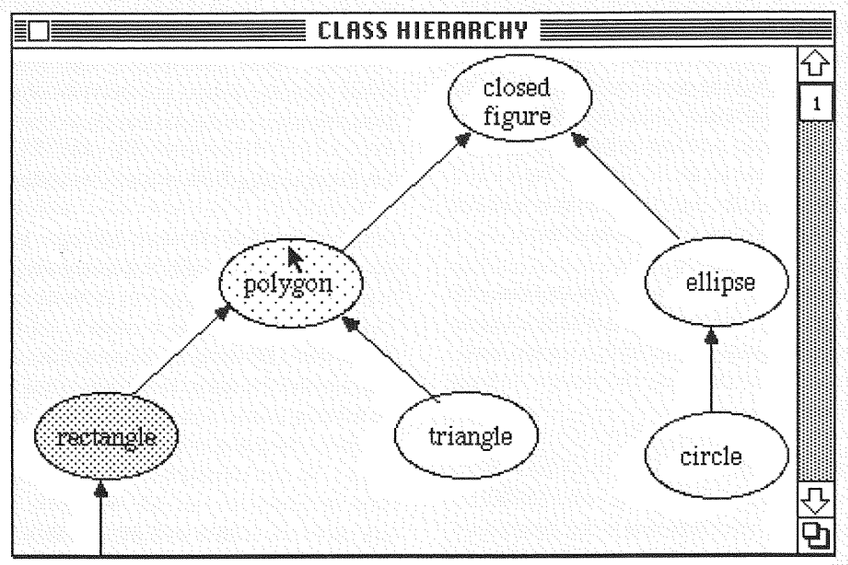
What we manipulate when editing a program are not passive cluster of words but living fragments that participate (even if incomplete) in an overall semantic.
The lack of immediate feedback, on these fragment, diminishes control over the programming process. To enhance the liveliness feeling we should anticipate, in the editing phase, some checks that are typical of the compile phase (e.g. type consistency check, assertions check). So the variables names occurring in an assignment statement are not just identifiers but belong to a type network with a precise semantic and should be checked against this semantic.

6.4. Debugger

The debugger access the run-time environment enabling to monitor through suitable animation the internal structures that the programmer belive are useful to visualize. The display should be customized by programmer opening and resizing a variable number of window , one for each interesting structure (e.g. fig. 5). The topics about the kind of animation to exploit and the control to provide to the user on the animation are extremely challenging. We refer to [3] and to [4] for the interesting approach and a valuable review.

7. CONCLUSION AND ACKNOWLEDGEMENT

In this paper we ave illustrated a convenient marriage. Object Oriented design promote the creative process by means of reusability and modifiability; hypertext enhance it by means of tools integration, retrievability and meaningful presentation of the informations. The author is grateful to Pr. Gianni Degli Antoni for his comments and suggestions, which significantly helped to develop a comprensive vision of the hypermedia technology.



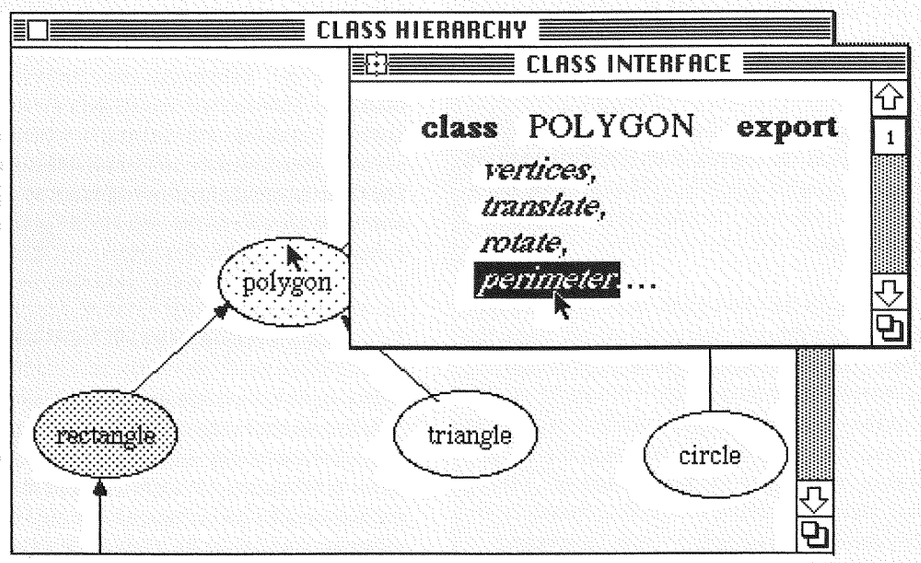


Fig. 2 e 3

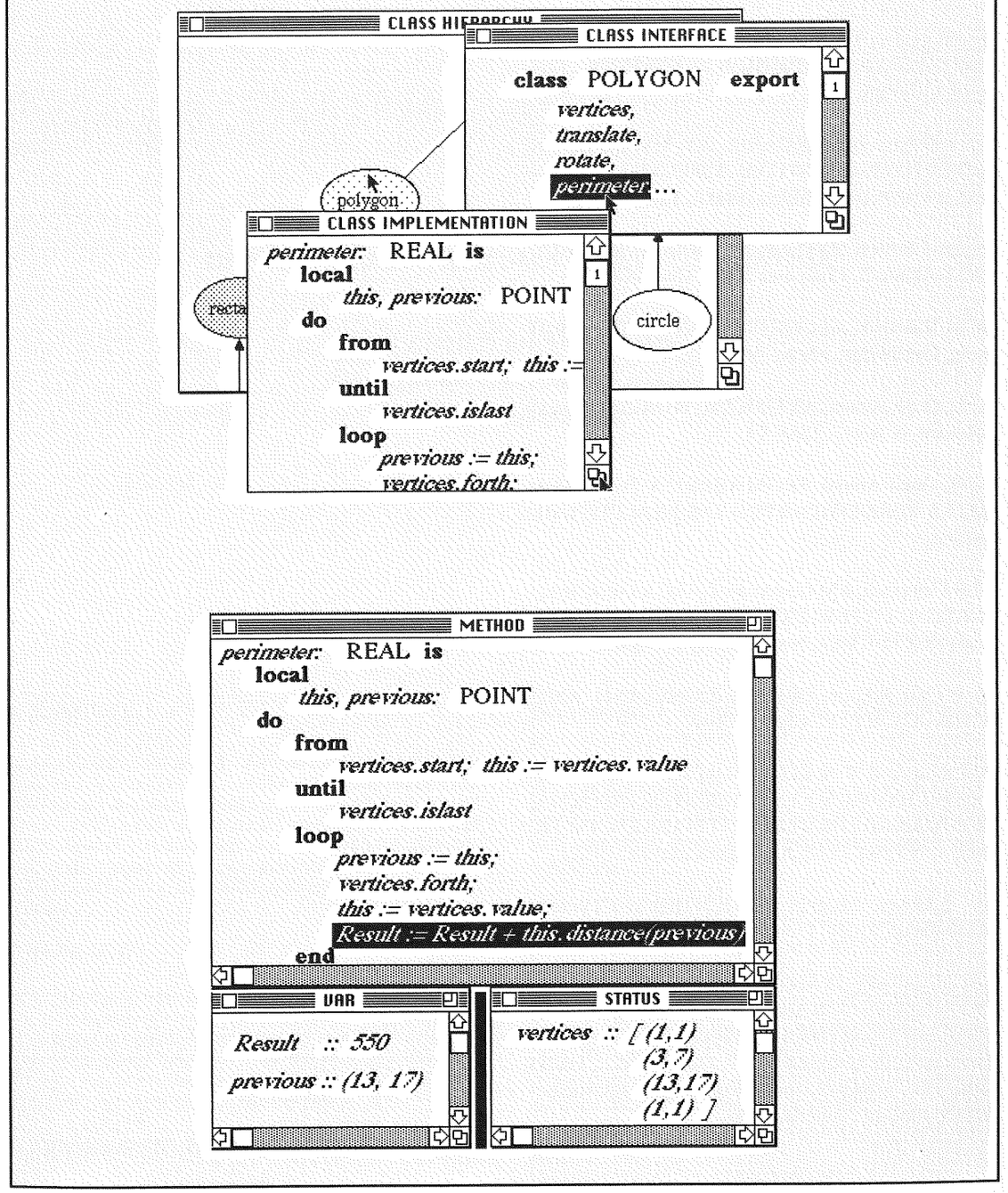


Fig. 4 e 5

8. REFERENCES.

[1] J.Bigelow, V.Riley, MANIPULATING SOURCE CODE IN DYNAMIC DESIGN, in HYPERTEXT 87, North Carolina, 1987

[2] A.Boder, HYPERCARD: AN EXAMPLE OF HYPERTEXT COGNITIVE ASPECTS, conference held at University of Milan, March, 1988

[3] M.H.Brown, EXPLORING ALGORITHMS USING BALSA-II, Computer 21(5), May, 1988

[4] M.H.Brown, ALGORITHM ANIMATION, MIT Press, Cambridge, Mass., 1988

[5] G. Degli Antoni, ARTIFICIAL WORLDS, Note di Software, N°42/43, 1988-89

[6] G. Degli Antoni, FROM REALITY TO HYPERMEDIA THROUGH ARTIFICIAL REALITY, forthcoming

[7] B.Campbell, J.M.Goodman, HAM: A GENERAL-PURPOSE HYPERTEXT ABSTRACT MACHINE, in HYPERTEXT 87, North Carolina, 1987

[8] B.Cox, B.Hunt, OBJECT, ICONS, AND SOFTWARE-ICS, BYTE, August, 1986.

[9] P.Garg and W.Scacchi, ON DESIGNING INTELLIGENT HYPERTEXT SYSTEMS FOR INFORMATION MANAGEMENT IN SOFTWARE ENGINEERING, in HYPERTEXT 87, North Carolina, 1987

[10] B.Meyer, OBJECT - ORIENTED SOFTWARE CONSTRUCTION, Prentice Hall, 1988

[11] J.Stelovsky, D.Ackermlann, P.Conti, VISUALIZATION OF PROGRAM STRUCTURES: SUPPORT CONCEPTS AND IMPLEMENTATIONS, in "Visualization in Programmin g", P. Gorny, M.J. Tauber (Eds.), 5th Interdisciplinary Workshop in Informatics and Psychology, Austria, 1986

DIMOSTRAZIONE AUTOMATICA IN GEOMETRIA

M.A. Alberti, E. Calzolari, M. Torelli

Dipartimento di Scienze dell'Informazione

Università di Milano

RIASSUNTO

Dopo un'introduzione generale alla dimostrazione automatica in geometria vengono presi in considerazione alcuni aspetti della dimostrazione con metodi algebrici e ci si sofferma in particolare sul metodo di Wu e sulle semplificazioni introdotte da Chou, che gli hanno consentito di dimostrare oltre 500 teoremi di geometria elementare. A conclusione si presentano delle considerazioni comparative con i metodi che ricorrono alla costruzione di una base di Gröbner.

ABSTRACT

After a general introduction automated theorem proving in geometry, some features of the algebraic proof methods are considered. Wu's method is examined in detail together with Chou's simplifications which enabled Chou to prove over 500 theorems in elementary geometry. In the conclusion some comparative remarks are made with proof methods costrusting a Gröbner basis.

1. INTRODUZIONE

Trovare una procedura di calcolo per provare la verità di enunciati, ovvero dimostrare teoremi automaticamente, è un problema da tempo affrontato, inizialmente con metodi più o meno euristici, successivamente con metodi esatti, come il principio di risoluzione di Robinson del 1965 o la procedura di Knuth e Bendix del 1967, infine nuovamente con metodi euristici o tecniche procedurali sviluppate in Intelligenza Artificiale.

Uno dei primi dimostratori automatici fu realizzato, proprio nel campo della geometria, nel 1958: la Geometry Theorem Machine (GTM) di Gelernter [11] [12]. Era stato preceduto qualche anno prima dalla Logic Theory Machine di Newell, Simon e Shaw: un programma in grado di dimostrare teoremi nel calcolo proposizionale, in particolare alcuni dei teoremi contenuti nel libro "Principia Mathematica" di Whitehead e Russel. Ad esempio il seguente teorema fu dimostrato in 8 passi:

$$((\neg Q \Rightarrow \neg P) \Rightarrow (P \Rightarrow Q)).$$

Da allora sono stati realizzati dimostratori in logica, geometria, teoria dei gruppi, teoria degli insiemi, analisi, topologia ed anche nei settori di sintesi e verifica di programmi, robotica ed altri. Per riferimenti si consultino [21, 9, 20, 3]. Alcuni sistemi sono completamente automatici, cioè forniscono la risposta ad un problema direttamente, altri invece forniscono prove interattivamente. In alcuni casi i dimostratori hanno risolto problemi ancora aperti (per esempio un problema sui reticoli modulari e altri problemi in teorie finitamente assiomatizzabili). Un caso interessante che non riguarda la dimostrazione bensì la formazione di congetture è costituito dal programma AM di Lenat [19], che è in grado di formare concetti nuovi a partire dai precedenti, di trovare degli esempi, quindi di formulare congetture, controllando l'esistenza di regolarità tra questi.

Tornando al sistema GTM realizzato da Gelernter e dai suoi collaboratori Rochester, Hansen e Loveland, esso dispone di un algoritmo per costruire prove di teoremi in una teoria assiomatica che costituisce un frammento della geometria piana. Dapprima si definisce la teoria assiomatica, dandone il linguaggio, gli assiomi ed infine le regole per derivare i teoremi. Quindi si procede all'indietro (deduzione di tipo backward chaining): si pone la tesi del teorema come obiettivo (goal), si cercano uno o più sotto obiettivi, che rendano vero il goal precedente, e si procede in questo modo fino a giungere alle ipotesi o a teoremi già noti. La dimostrazione è quindi rappresentata da un albero AND-OR, la cui radice è la tesi del teorema e le cui foglie sono gli assiomi o le ipotesi del teorema. Per contenere l'esplosione combinatoria dell'albero si fa uso del *diagramma* del problema: una figura che rappresenta il problema su cui sia possibile fare *misure* per scartare alcuni sotto goals che si rivelassero falsi (idea questa applicata in seguito in altre aree).

Ad esempio diamo la traccia semplificata della dimostrazione del seguente teorema: due vertici di un triangolo sono equidistanti dalla mediana relativa al lato compreso tra questi. Il problema è rappresentato dal diagramma della figura 1, dalle

ipotesi: BM = MC, BD ⊥ AM, CE ⊥ ME e dalla

tesi: BD = EC

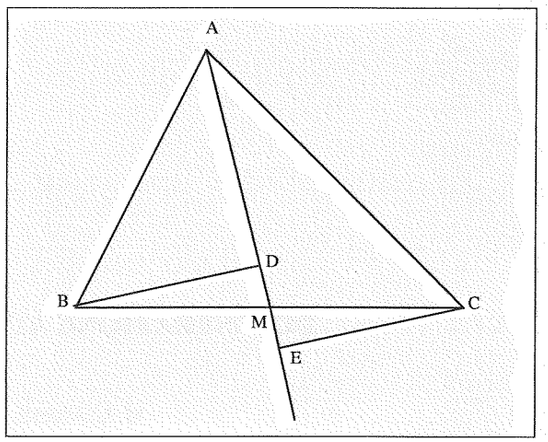


Fig.1. Il diagramma del problema geometrico

Soluzione	Commento
angolo DMB = angolo EMC	angoli opposti al vertice
angolo BDM = angolo CEM	angoli retti
BM = MC	per ipotesi
CEM è un triangolo	deducibile dal diagramma (vedi testo)
BDM è un triangolo	deducibile dal diagramma (vedi testo)
triangolo CEM = triangolo BDM	2° criterio di congruenza
BD = EC	elementi corrispondenti di triangoli congruenti

Per ricavare la soluzione la GTM procede all'indietro ponendosi come goal

G_1 segmento BD = segmento EC.

Tra le condizioni che implicano l'uguaglianza di segmenti la GTM trova quella che afferma che due segmenti sono uguali se sono parti corrispondenti di triangoli congruenti. Poichè il sistema GTM desume dal diagramma che BMD e CEM non sono terne di punti collineari, e quindi definiscono triangoli, esso procede ponendosi il secondo goal parziale

G_2 triangolo CEM = triangolo BDM.

Il sistema tenta ora di provare l'uguaglianza dei due triangoli secondo i diversi criteri di congruenza: fallendo con il 1° procede con il 2° ponendosi gli ulteriori

goals

G_3 segmento BM = segmento MC
G_4 angolo EMC = angolo DMB
G_5 angolo CEM = angolo BDM.

Il goal G_3 è nelle ipotesi, i goals G_4 e G_5 sono soddisfatti per i criteri di uguaglianza tra angoli noti al sistema (angoli opposti al vertice, tutti gli angoli retti sono uguali, ecc.). Quindi il sistema conclude che essendo verificati G_4 G_4 G_5 allora lo è G_2 e di conseguenza G_1.

Naturalmente il sistema ha però costruito un albero di ricerca ben più complesso, per esempio cercando di provare G_1 con diversi sotto goals

$G_{2.1}$ triangolo CEA = triangolo BDA
$G_{2.2}$ triangolo CEA = triangolo BDM
$G_{2.3}$ triangolo CEM = triangolo BDA.

Ciascuno di questi sotto goal deve venire verificato; l'analisi diretta comporterebbe la ricerca di tutti i criteri di congruenza e di conseguenza una crescita onerosa dell'albero della soluzione. Il diagramma consente invece di scartare tutti i tre goals come falsi, semplicemente controllando numericamente l'uguaglianza delle misure dei segmenti.

Il lavoro di Gelernter fu ripreso ed ampliato per un sottoinsieme della geometria euclidea piana (in particolare costituito da triangoli congruenti, segmenti ed angoli uguali, linee parallele e parallelogrammi) da Goldstein [13]; l'approccio di deduzione è ancora di tipo backward chaining, ma l'enfasi è posta sulla ricerca di un formalismo per rappresentare conoscenza matematica sotto forma di programmi, inoltre il sistema contiene un generatore di mosse plausibili per guidare la prova. Successivamente Nevins [22] estende il metodo di Gelernter adottando una tecnica mista di deduzioni backward e forward.

In tempi recenti Anderson e collaboratori hanno realizzato il Geometry Tutor [1], ambiente interattivo per controllare prove di teoremi, costituito da tre componenti: il modulo delle regole corrette e scorrette, quest'ultime dedotte dall'analisi di protocolli prodotti dagli studenti, il modulo per dirigere la strategia d'insegnamento ed il modulo per l'interfaccia grafica. Sostanzialmente, un albero di goal e sotto goals viene prodotto e visualizzato graficamente, ad ogni passo lo studente può ragionare e fare inferenze partendo dalle ipotesi (forward chaining) o dalla tesi (backward chaining); il grafo viene costruito aggiungendo affermazioni sullo schermo opportunamente legate da frecce ed è completo quando l'enunciato da provare è connesso con inferenze corrette alle ipotesi. L'enfasi del lavoro è sull'aspetto didattico.

Infine Coelho e Pereira [8] riconsiderano il metodo di deduzione a partire da una base di conoscenze con un sistema in Prolog, con il dichiarato scopo di raggiungere una migliore comprensione delle possibilità di Prolog, come strumento per l'automazione del ragionamento; limiti e punti critici del sistema e del metodo sono messi in evidenza.

Il vantaggio dell'approccio di deduzione backward o forward di Gelernter e dei successivi autori citati consiste nell'essere abbastanza simile al modo naturale di dimostrare i problemi, importante caratteristica quando si vuole enfatizzare l'aspetto didattico e interattivo del sistema; d'altra parte i limiti di questo approccio consistono nella esplosione combinatoria dell'albero di soluzione, che rende tali sistemi inutilizzabili per problemi appena più complessi.

Un altro metodo, sostanzialmente più potente, fu introdotto dal matematico cinese Wu negli anni '70. I suoi lavori cominciarono a circolare tradotti in inglese solo nei primi anni '80 [25, 24]. L'approccio introdotto da Wu è algebrico e si basa sul calcolo dell'insieme caratteristico di Ritt: dimostrare il problema geometrico equivale ad eseguire con successo una opportuna manipolazione sul sistema di equazioni polinomiali che rappresentano il problema in un dato sistema di riferimento. Ai lavori di Wu si sono aggiunti quelli di Chou [4, 5], che hanno contribuito a semplificare e a diffondere il metodo, esemplificandolo con la soluzione di più di 500 teoremi di geometria euclidea e non [6].

Altri metodi algebrici, che fanno uso delle basi di Gröbner, sono stati investigati da Kutzler e Stifter [18] e messi anche sperimentalmente a confronto con il metodo di Wu [7]. In entrambi i casi si controlla che il polinomio che rappresenta la tesi appartenga all'ideale generato dai polinomi delle ipotesi sotto opportune condizioni.

Inoltre Kapur fornisce un metodo che dimostra per assurdo i teoremi, sempre facendo uso delle basi di Gröbner [15, 16], metodo che approfondisce in [17], in cui tra l'altro discute anche gli altri approcci che usano le basi di Gröbner e li confronta criticamente con il metodo di Wu-Chou.
I più recenti sviluppi insieme con analisi critiche sono contenuti in [14].

2. ASPETTI GENERALI DEI METODI ALGEBRICI

Dimentichiamo per un momento la geometria e supponiamo di avere il seguente sistema di equazioni, in cui a, b e c sono *parametri*, ossia valori non specificati, mentre x, y, z e t sono le incognite:

(1) $x^2 + y^2 = b^2 + c_2$
(2) $cy = -bx$
(3) $2z = a + b$
(4) $2t = c.$

In questo caso è molto facile risolvere il sistema. Infatti le soluzioni sono:

$x = \mp c;$ $y = \pm b;$ $z = (a + b)/2;$ $t = c/2.$

Avendo risolto simbolicamente il sistema, possiamo rispondere a quesiti del tipo " vale la relazione seguente ? "

(5) $4t^2 + 4z^2 = x^2 + y^2 + a^2 - 2ay.$

Basta infatti sostituire nella (5) le soluzioni trovate, ottenendo la relazione:

$$c^2 + a^2 + b^2 + 2ab = c^2 + b^2 + a^2 - 2a\ (\pm b)$$

da cui si deduce che la (5) è vera se e solo se $y = -b$ (e di conseguenza $x = c$).

Le equazioni (1)..(5) sopra viste sono la traduzione in termini algebrici del seguente problema geometrico: dato il triangolo ABC, si costruiscano sui lati AC e BC i quadrati ACDE e BCFG; se M è il punto medio di AB, allora DF = 2 CM.

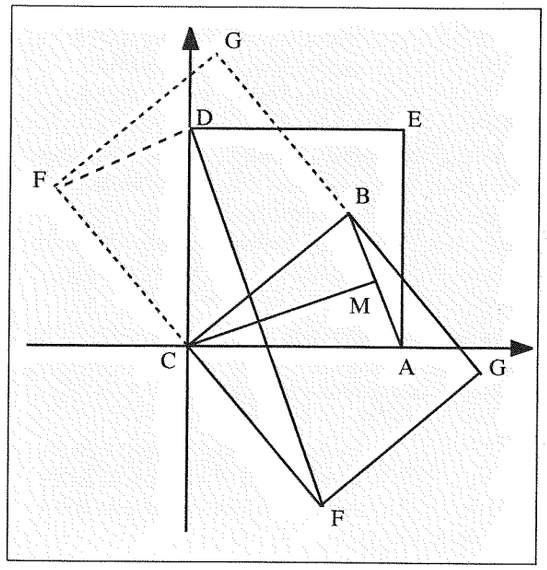


Fig. 2. Teorema dei due quadrati

Infatti scelto un sistema di riferimento cartesiano e posto A = (a, 0); B = (b, c); C = (0, 0); D = (0, a); F = (x, y); M = (z, t); l'equazione (1) esprime l'uguaglianza CF = BC; la (2) è la condizione di perpendicolarità tra CF e BC; (3) e (4) traducono in termini algebrici la condizione "M è il punto medio di AB"; infine, la (5) asserisce che $4\ CM^2 = DF^2$.

Osservando la figura 2 possiamo notare che è possibile costruire *due* quadrati sul lato BC (così come è possibile costruirne due sul lato AC: nella figura è mostrato solo il quadrato avente come vertice D = (0, a), e non quello con D = (0, - a)). I due quadrati sono quello tracciato con linea piena, per il quale vale (5), e quello tratteggiato, per cui si ha invece DF = AB.

Sempre osservando la figura, possiamo anche *congetturare* che CM è perpendicolare a DF, e *dimostrarlo* a partire dall'equazione che esprime la condizione di perpendicolarità:

(6) $zx = t(a - y).$

Semplicemente effettuando le sostituzioni $x = c$ e

$y = -b$, si ottiene infatti l'identità
$c\ (a + b)/2 = (c/2)\ (a - (- b))$ (nel caso $x = - c$ e $y = b$ si ha invece CM parallelo a DF).

Tuttavia, per verificare la validità di (5) non è necessario risolvere il sistema delle (1)..(4). Infatti si può, ad esempio, sostituire direttamente la (4) nella (5), pur di scrivere $4t^2$ come $(2t)^2$; analogamente si può eliminare la variabile z, sostituendo la (3) nella (5); infine si può eliminare $x^2 + y^2$ sostituendo la (1) nella (5), e ottenere: $c^2 + (a+b)^2 = b^2 + c^2 - 2ay$. In questo modo però non si riescono ad eliminare *tutte* le variabili dalla (5): si può sostituire y con x, tramite la (2), ma non si elimina la variabile x oppure la variabile y.

Così facendo, abbiamo considerato le equazioni (1)..(4) alla stregua di *regole di riscrittura* che abbiamo applicato nella (5). Abbiamo altresì constatato che il sistema (1)..(4) non è *completo*, cioè non consente di ricavare *tutte* le espressioni che posso ottenere dal sistema interpretando le (1)..(4) come equazioni. In conseguenza di ciò, una stessa espressione può essere riscritta in modi diversi, sino a giungere ad espressioni irriducibili differenti. Per esempio, data la relazione $c^2(x^2 + y^2)$, si può sostituire $x^2 + y^2$ con $b^2 + c^2$, tramite la (1), ottenendo $c^2(b^2 + c^2)$; ma riscrivendo la relazione data come $c^2(x^2 + y^2) = c^2x^2 + (cy)^2$ si può sostituire cy con -bx, tramite la (2), ottenendo $x^2(b^2 + c^2)$. Per essere reso rigoroso questo discorso dovrebbe essere formalizzato e ampiato specificando il verso delle regole di riscrittura, esplicitando l'assioma di commutatività e altri dettagli importanti. Interessa qui dare una prima idea: per approfondimenti si può consultare per esempio [2].

Le regole di riscrittura (1) e (2) costituiscono una *coppia critica,* e per poter ottenere da esse la stessa espressione irriducibile dobbiamo *completare* l'insieme delle regole introducendo una nuova regola che renda identiche le due espressioni: nel nostro caso tale regola può essere $x^2 = c^2$.

Nel caso di un sistema di equazioni polinomiali, un insieme di polinomi "equivalente" all'insieme di partenza, ma completo, si chiama *base di Gröbner:* un insieme cosiffatto consente di trasformare qualsiasi polinomio in qualsiasi altro ad esso equivalente rispetto alle equazioni di partenza. Esistono algoritmi che costruiscono una base di Gröbner a partire da un insieme finito di polinomi, ma pur usando degli accorgimenti, quale quello di semplificare via via l'insieme di polinomi, tali algoritmi sono in generale piuttosto dispendiosi. In ogni caso, è chiaro che predisporre un apparato in grado di manipolare un polinomio *qualsiasi* è, in linea di principio, più dispendioso che non partire dallo specifico polinomio (o insieme di polinomi) esprimente la tesi, e manipolarlo direttamente.

Proviamo ad esplorare questa via. Supponiamo di avere n equazioni $H_1 = 0$ .. $H_n = 0$ esprimenti le ipotesi, e l'equazione $G = 0$, che esprime la tesi. Se riusciamo a trovare 2n polinomi P_i e Q_i, $1 \le i \le n$, tali che P_1 .. $P_n\ G = Q_1H_1 + .. + Q_nH_n$, e *se i* P_i *non si annullano,* allora $H_i = 0$, $1 \le i \le n$, implica $G = 0$.

Il modo "naturale" per arrivare a un tale risultato è il seguente: cerco di determinare P_1, Q_1 e R_1 in modo che $P_1\ G = Q_1\ H_1 + R_1$, quindi itero il procedimento sui successivi resti R_i ottenuti, in modo che $P_{i+1}\ R_i = Q_{i+1}\ H_{i+1} + R_{i+1}$, per i da 1 a n-1. Se la tesi è vera, mi aspetto di trovare un ultimo resto $R_n = 0$, pur di aver fatto le cose per bene, cioè di aver "diviso" ogni R_i per H_{i+1} in modo da ottenere un resto R_{i+1} di grado inferiore a quello di H_{i+1} rispetto a una variabile specificata. Come vedremo nella sezione successiva, che illustra in dettaglio il metodo di Wu-Chou, la scelta delle variabili rispetto a cui dividere va fatta con oculatezza, e la situazione è in generale più complicata di quello che il nostro esempio può far supporre.

Riprendiamo comunque l'esempio: in esso
$G = x^2 + y^2 + a^2 - 2ay - 4t^2 - 4z^2$, per la (5),
$H_1 = x^2 + y^2 - b^2 - c^2$, per la (1), e quindi possiamo scrivere, per esempio,
$G = H_1 + R_1$ con
$R_1 = G - H_1 = a^2 + b^2 + c^2 - 2ay - 4t^2 - 4z^2.$

Ora $H_2 = bx + cy$ e $-cR_1 = 2aH_2 + R_2$ con
$R_2 = -2abx - a^2c - b^2c - c^3 + 4t^2c + 4z^2c.$
$H_3 = 2z - a - b$ e $R_2 = (2z + a + b)\ cH_3 + R_3$, da cui
$R_3 = -2abx - c^3 + 4t^2c + 2abc$ e finalmente
$H_4 = 2t - c$, $R_3 = (2t + c)\ cH_4 + R_4$, da cui
$R_4 = 2ab\ (c - x).$

In conclusione abbiamo $cG =$
$cH_1 - 2aH_2 - (2z + a + b)\ cH_3 - (2t+c)\ cH_4 - 2ab\ (c-x),$

e quindi effettivamente $G = 0$ è una conseguenza di $H_1 = H_2 = H_3 = H_4 = 0$, *purché* sia $x = c \neq 0$. Ricordiamo che c è un parametro, che può quindi essere scelto arbitrariamente: in particolare, c è l'ordinata del vertice B del triangolo: se $c = 0$ il triangolo collassa sul lato AC. La condizione $c \neq 0$ è quindi una condizione di *non degenerazione* del problema, che peraltro in questo caso specifico non inficia la tesi. Al contrario, x è una variabile dipendente dai parametri, e quindi in generale non può assumere valori specificati; nel caso in esame, però, $x = c$ è compatibile con le ipotesi, ma corrisponde a uno solo dei due possibili valori di x, come già sapevamo.

3. IL METODO DI WU

Il metodo di Wu è un metodo di tipo algebrico:le ipotesi e la tesi dell'enunciato espresso in forma geometrica sono rappresentate da equazioni algebriche a coefficienti interi. Il problema geometrico espresso in forma algebrica sarà quindi composto da un sistema di equazioni che rappresentano le ipotesi, $H_1 = 0 .. H_t = 0$, e da un' equazione (o più di una), $G = 0$, che rappresenta la tesi.
Naturalmente non tutti gli enunciati geometrici si possono esprimere in questa forma: per esempio non si potranno prendere in considerazione problemi che coinvolgono relazioni d'ordine.

La rappresentazione algebrica del problema si ottiene fissando un sistema di riferimento cartesiano ed esprimendo i vincoli del problema con relazioni tra le variabili che rappresentano le coordinate dei punti. Occorre a tal fine distinguere le variabili indipendenti da quelle dipendenti, e disporre di tante equazioni $H_i = 0$ quante sono le variabili dipendenti.

Dimostrare un teorema risulta quindi equivalente a verificare che gli zeri del sistema associato alle ipotesi sono anche zeri dell'equazione che rappresenta la tesi, sotto ulteriori condizioni di non degenerazione che vengono ricavate automaticamente.
Il metodo si fonda sui seguenti punti:

1. costruire l'insieme caratteristico (concetto dovuto a Ritt [23]) del sistema di polinomi delle ipotesi
2. calcolare il resto finale delle pseudo divisioni successive del polinomio G rispetto ai polinomi dell'insieme caratteristico determinato al punto 1
3. analizzare la riducibilità dei polinomi dell'insieme caratteristico.

3.1 Costruzione dell'insieme caratteristico

A partire dai polinomi delle ipotesi $H_1..H_t$ si costruisce un insieme di polinomi $F_1..F_t$, detto insieme caratteristico, che verifica le seguenti proprietà:

1. i polinomi $F_1 .. F_t$ sono in forma triangolare rispetto alle variabili dipendenti $x_1 .. x_t$, ordinate secondo un ordinamento totale
   $x_1 < x_2 < .. < x_t$:
   $F_1 (u_1 .. u_d\ x_1) = 0$
   $F_2 (u_1 .. u_d\ x_1\ x_2) = 0$
   . . .
   $F_t (u_1 .. u_d\ x_1\ x_2 .. x_t) = 0$
2. ogni polinomio F_j è ridotto rispetto ad ogni polinomio $F_1 .. F_{j-1}$ (si dice che F_j è ridotto rispetto a F_i se si verifica che in F_i il grado massimo con cui compare la variabile x_i è maggiore del grado massimo con cui x_i compare in F_j.)"
3. valgono le seguenti relazioni tra gli zeri dell'insieme caratteristico $F_1 .. F_t$ e quelli delle equazioni polinomiali associate alle ipotesi $H_1..H_t$:
   zeri (CHS / J) $\subseteq$ zeri (IP) $\subseteq$ zeri (CHS)
   zeri (IP) = zeri (CHS / J) $\cup \cup_i$ zeri (IP_i)
   dove CHS è l'insieme caratteristico $F_1 .. F_t$, IP è l'insieme $H_1 .. H_t$, J è il prodotto degli *iniziali* I_i dei polinomi F_i espressi come polinomi nella sola variabile x_i:
   $F_i = I_i\ x_i^m + a_{m-1}\ x_i^{m-1} + .. + a_0$
   e IP_i è l'insieme ottenuto aggiungendo I_i all'insieme $H_1..H_t$.

Per ogni insieme finito di polinomi è possibile trovare un insieme caratteristico e Wu fornisce un algoritmo di calcolo in [25]. Recentemente G.Gallo [10] ha dimostrato che tale algoritmo è doppiamente esponenziale, ma che in generale il problema della determinazione di un insieme caratteristico è semplicemente esponenziale.

3.2 Calcolo del resto finale delle pseudodivioni

Dopo aver calcolato l'insieme caratteristico $F_1 .. F_t$, si procede nel calcolo del resto finale delle pseudodivisioni successive del polinomio della tesi G rispetto ai polinomi $F_1 .. F_t$. Cominciamo con il definire un algoritmo

per il calcolo della pseudo divisione tra polinomi. Siano assegnati due polinomi
$A = a_n v^n + .. + a_0$, $B = b_k v^k + .. + b_0$: indicato con gr(A,v) il grado del polinomio A rispetto alla variabile v, si calcola lo pseudo resto tra A e B, rispetto a tale variabile, con la seguente procedura:

```
procedura  PsResto (A, B, v)
      R := A
      k := gr (B, v)
      while gr (R, v) ≥ k  do  begin
                m  : = gr (R, v)
                c_m : = monomio principale di R
                R  : = b_k * R - c_m * B * v^(m-k)
      end
output  R
```

Al termine della procedura vale la seguente formula:
$b_k^s A = Q B + R$, dove $s \leq n - k + 1$ e
$gr (R, v) < gr (B, v)$.

Diamo ora una procedura per il calcolo delle pseudo divisioni successive del polinomio G rispetto al sistema di polinomi $F_1 .. F_t$ in forma triangolare:

```
procedura PsDivSucc ( G, F_1 .. F_t )
      R_t := G
      for  k :=t  downto 1   do
R_(k-1):= PsResto ( R_k, F_k, x_k )
output R_0
```

Il risultato R_0 è il resto finale delle divisioni successive. Al termine della procedura si può affermare che esistono t interi non negativi s_j e t polinomi Q_j tali che:

a. $I_1^{S_1} 1 .. I_t^{S_t} t\ G = Q_1 F_1 + .. + Q_t F_t + R_0$
b. $gr (R_0, x_j) < gr (F_j, x_j)$ $\quad j = 1 .. t.$

Se il resto finale R_0 che compare nella formula dei resti a. è nullo allora posso concludere che l'enunciato è un teorema con l'aggiunta delle condizioni di non degenerazione $I_1^{S_1} \neq 0 .. I_t^{S_t} \neq 0$ ottenute automaticamente dal metodo.

3.3 Analisi della riducibilità di $F_1 .. F_t$

Se il resto finale non è nullo non si potrà subito concludere che l'enunciato è falso: si dovrà procedere all'analisi della riducibilità dei polinomi associati alle ipotesi in forma triangolare.

Infatti, se ogni polinomio F_j è irriducibile nell'estensione $Q(u_1..u_d)\ [x_1..x_{j-1}] / (F_1..F_{j-1})$, allora si potrà dire che l'enunciato è falso (sempre supponendo che le ipotesi non siano inconsistenti).

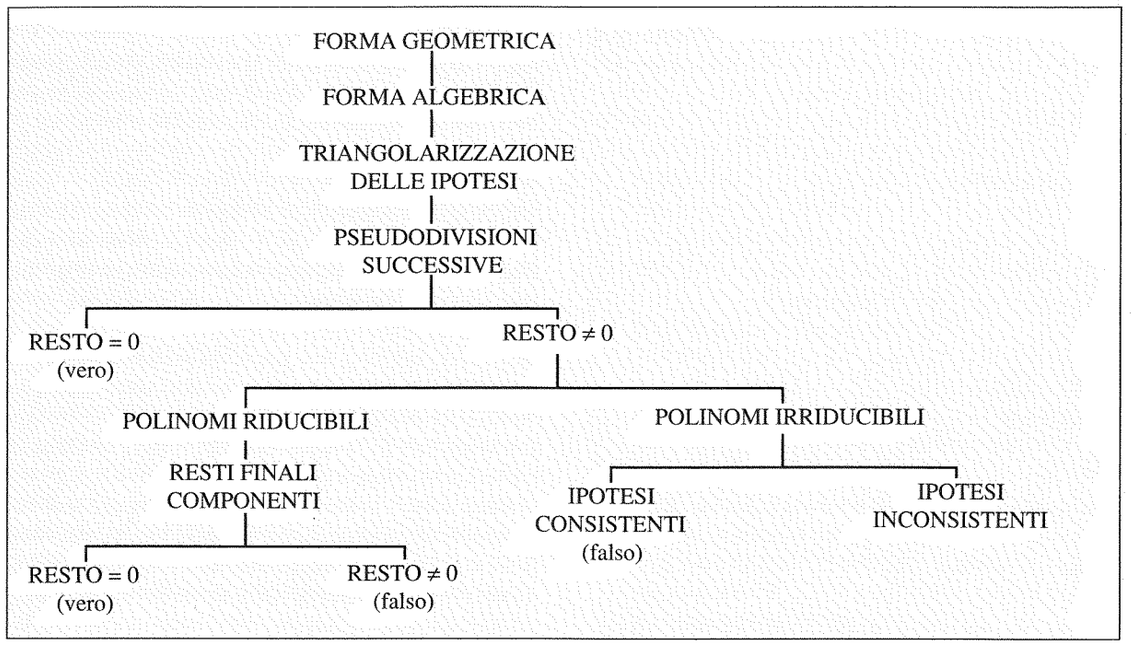
Se invece esistono dei polinomi riducibili allora si dovranno prendere in considerazione tutte le componenti possibili del sistema dei polinomi $F_1 .. F_t$ (per esempio, se F_2 è riducibile, $F_2 = F_2'F_2''$, allora si considerano le due componenti $F_1 F_2' .. F_t$ e $F_1 F_2'' .. F_t$) e riapplicare il metodo per calcolare i resti finali del polinomio associato alla tesi G rispetto a ciascuna componente. Quindi, se ogni resto finale calcolato è nullo, allora si potrà concludere che l'enunciato di partenza è un teorema con l'aggiunta delle condizioni di non degenerazione rappresentate dagli iniziali dei polinomi delle ipotesi in forma triangolare. Se un resto non è nullo allora l'enunciato generale sarà falso, ma risultano veri gli enunciati con le condizioni aggiuntive corrispondenti alle componenti in cui il resto risulta nullo.
Per un algoritmo che indaga la riducibilità di polinomi e li fattorizza nel caso quadratico si rimanda a [5].

3.4 Semplificazione introdotta da Chou

Nell'implementazione del metodo si è utilizzata una semplificazione introdotta da Chou. La costruzione dell'insieme caratteristico è stata sostituita dalla costruzione di un insieme di polinomi, detto catena ascendente in senso debole, $F_1 .. F_t$ tale che:

1. $F_1 .. F_t$ sono in forma triangolare rispetto alle variabili $x_1 .. x_t$
2. il resto finale delle pseudo divisioni successive di ogni iniziale I_j dei polinomi F_j rispetto ai polinomi $F_1 .. F_t$ non è nullo.

Schematicamente il metodo di Wu-Chou può essere così riassunto:



3.5 Esempio

Illustriamo il metodo di Wu, attraverso il seguente esempio: dato un parallelogramma ABCD, sia E l'intersezione delle diagonali AC e BD, dimostriamo che E è il punto medio delle diagonali.

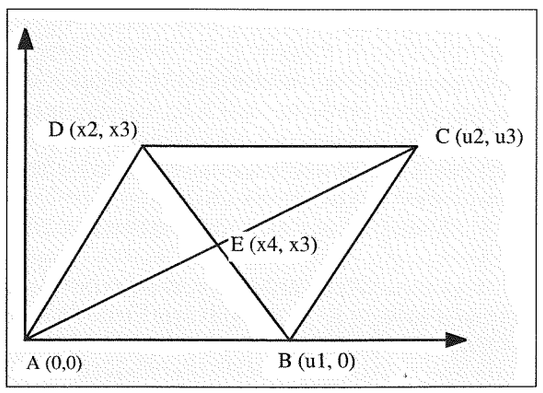


Fig. 3. Teorema delle diagonali

3.5.1 La forma algebrica del problema

Fissato un sistema di riferimento cartesiano, i punti A, B e C della figura possono essere scelti arbitrariamente, quindi le rispettive coordinate potranno ad esempio essere A (0, 0), B(u_1, 0), C (u_2, u_3), dove u_1, u_2, u_3 sono dei parametri.

Fissati i punti A, B e C, il punto D dovrà invece appartenere alla retta passante per A e parallela a BC ed alla retta passante per C e parallela ad AB: le sue coordinate sono quindi una coppia di variabili dipendenti cioè D = (x_2, x_1). Analogamente per il punto E, intersezione delle due diagonali: E = (x_4, x_3).

Le equazioni algebriche corrispondenti alle ipotesi sono le seguenti:

H_1: $u_1 x_1 - u_1 u_3 = 0$ — AB è parallelo a CD

H_2: $u_3 x_2 - (u_2 - u_1) x_1 = 0$ — AD è parallelo a BC

H_3: $x_1 x_4 - (x_2 - u_1) x_3 - u_1 x_1 = 0$ — E appartiene a BD

H_4: $u_3 x_4 - u_2 x_3 = 0$ — E appartiene ad AC

L'equazione corrispondente alla tesi è:

G: $2 u_2 x_4 + 2 u_3 x_3 - u_3^2 - u_2^2 = 0$ — segmento AE = segmento CE

3.5.2 Triangolazione delle ipotesi

Il passo successivo del metodo di Wu è rappresentato dalla triangolarizzazione dei polinomi che rappresentano le ipotesi.

Assegnato quindi come ordinamento totale fra le variabili dipendenti l'usuale ordine $x_1 < .. < x_t$, si trasformano le equazioni polinomiali delle ipotesi nelle equazioni seguenti:

F_1: $x_1 u_1 - u_3 u_1 = 0$

F_2: $x_2 u_3 + (u_1 - u_2) x_1 = 0$

F_3: $x_3 (x_2 u_3 - x_1 u_2 - u_3 u_1) + x_1 u_3 u_1 = 0$

F_4: $u_3 x_4 - u_2 x_3 = 0$

Per un algoritmo che triangolarizzi sistemi di equazioni algebriche si rimanda a [6].

3.5.3 Le pseudodivisioni

Espresse le ipotesi in forma triangolare si dovrà verificare che la tesi è pseudodivisa dalle ipotesi così trasformate. Nel nostro caso si ottiene che il resto finale delle pseudodivisioni successive del polinomio della tesi G rispetto ai polinomi delle ipotesi in forma triangolare F_1 F_2 F_3 F_4 è nullo. Si può quindi concludere che l'enunciato è un teorema sotto le condizioni di non degenerazione:

I_1 : $u_1 \neq 0$

I_2 : $u_3 \neq 0$

I_3 : $x_2 u_3 - x_1 u_2 - u_3 u_1 \neq 0$

I_4 : $x_1 \neq 0$

il cui significato geometrico consiste nella richiesta che i punti A B C non siano allineati e che l'intersezione delle diagonali sia propria.

4. CONCLUSIONI

In questa rassegna si è cercato di illustrare sommariamente i principali metodi di dimostrazione automatica di teoremi geometrici. I metodi algebrici di cui abbiamo parlato si richiamano o alla teoria delle basi di Gröbner e agli algoritmi per costruire tali basi, o al metodo di Wu, fondato sulle basi di Ritt e integrato con le osservazioni di Chou.

Presso il Dipartimento di Scienze dell'Informazione dell'Università degli Studi di Milano sono stati realizzati due sistemi di dimostrazione automatica che implementano i due metodi citati. Tali sistemi sono stati realizzati con il sistema di manipolazione simbolica muMATH, per macchine PC IBM compatibili, ed hanno consentito di constatare l'effettiva utilizzabilità del metodo di Wu-Chou per la dimostrazione di teoremi non banali anche su macchine di prestazioni modeste.

I due sistemi hanno inoltre permesso di verificare la maggior efficienza del metodo di Wu-Chou rispetto al metodo che usa le basi di Gröbner, come viene anche evidenziato in [14]. Per esempio, il teorema di Pascal sulla circonferenza è stato dimostrato con il nostro sistema basato sul metodo di Wu-Chou in circa 45 minuti, laddove a Kutzler e Stifter sono occorse quasi 4 ore per dimostrarlo su un IBM 4341, usando le basi di Gröbner. E' da notare tuttavia che Kapur, introducendo esplicitamente le condizioni di non degenerazione, lo ha dimostrato con le basi di Gröbner in meno di 11 minuti su un elaboratore Symbolics 3640. Questi confronti di prestazioni hanno un valore limitato, dato che si riferiscono a programmi ed elaboratori diversi, ma sono comunque indicativi.

Qualche prima sperimentazione che abbiamo realizzato su Apple Macintosh II con il nuovo sistema Mathematica, che dispone di programmi di libreria per il calcolo delle basi di Gröbner, ha dato risultati poco incoraggianti.

5. RINGRAZIAMENTI

Lo studio di questo argomento e la realizzazione dei sistemi hanno tratto spunto da alcune tesi di laurea svolte o in corso presso il Dipartimento. Gli autori desiderano ringraziare in particolare Alberto Bertoni che ha avviato la ricerca e l'ha sostenuta con numerosi suggerimenti e Giulio Falco che ha contribuito alla realizzazione del sistema che usa le basi di Gröbner e alla sperimentazione con Mathematica.

Questo lavoro è stato in parte finanziato dal Ministero della Pubblica Istruzione con fondi 40% "Calcolo algebrico e simbolico".

6. BIBLIOGRAFIA

[1] J.R.Anderson, C.F.Boyle, G.Yost THE GEOMETRY TUTOR, in Proceedings IJCAI- 85, Los Angeles (1985) 1-7

[2] B.Buchberger, R.Loos ALGEBRAIC SIMPLIFICATION, in B.Buchberger, G.E.Collins, R.Loos (eds) Computer Algebra, 11-43, Springer-Verlag, Wien - New York, 1982

[3] C.Chang, R.C.Lee SYMBOLIC LOGIC AND MECHANICAL THEOREM PROVING, Academic Press, New York 1973

[4] S.C.Chou PROVING ELEMENTARY GEOMETRY THEOREMS USING WU'S ALGORITHM, Contemporary Mathematics, **29** (1984) 243-286

[5] S.C.Chou PROVING AND DISCOVERING THEOREMS IN ELEMENTARY GEOMETRIES USING WU'S METHOD, PHD Thesis, Department of Mathematics, University of Texas, Austin 1985

[6] S.C.Chou MECHANICAL GEOMETRY THEOREM PROVING, Reidel Publ.Co. 1988

[7] S.C.Chou, W.F.Schelter PROVING GEOMETRY THEOREMS WITH REWRITE RULES, Journal of Automated Reasoning, **2** (1986) 253-273

[8] H.Coelho, L.M.Pereira AUTOMATED REASONING IN GEOMETRY THEOREM PROVING WITH PROLOG, Journal of Automated Reasoning, **2** (1986) 329-390

[9] M.Davis THE PREHISTORY AND EARLY HISTORY OF AUTOMATED DEDUCTION, in Siekmann, Wrightson (eds) The Automation of Reasoning I, Springer Verlag 1983

[10] G.Gallo, Tesi di Dottorato in Matematica, Consorzio Università di Catania, Messina e Palermo, ottobre 1989

[11] H.Gelernter REALIZATION OF A GEOMETRY-THEOREM PROVING MACHINE in E.A.Feigenbaum, J.Feldman (eds) Computers and Thought, 134-152, McGraw-Hill, New York 1963

[12] P.C.Gilmore AN EXAMINATION OF THE GEOMETRY THEOREM MACHINE, Artificial Intelligence, **1** (1970) 171-187

[13] I.Goldstein ELEMENTARY GEOMETRY THEOREM PROVING, AI Memo 280, MIT AI Laboratory, Cambridge, MA., aprile 1973

[14] D.Kapur, J.L.Mundy (eds) GEOMETRY REASONING, special volume, Artificial Intelligence, **37** (1988)

[15] D.Kapur USING GRÖBNER BASES TO REASON ABOUT GEOMETRY PROBLEMS, Journal of Symbolic Computation, **2** (1986) 399-408

[16] D.Kapur GEOMETRY THEOREM PROVING USING HILBERT'S NULLSTELLENSATZ in Proceedings SYMSAC-86, Waterloo (1986) 202-208

[17] D.Kapur A REFUTATIONAL APPROACH TO GEOMETRY THEOREM PROVING, Artificial Intelligence, **37** (1988) 61-93

[18] B.Kutzler, S.Stifter ON THE APPLICATION OF BUCHBERGER'S ALGORITHM TO AUTOMATED GEOMETRY THEOREM PROVING, Journal of Symbolic Computation, **2** (1986) 389-397

[19] D.B.Lenat ON AUTOMATED SCIENTIFIC THEORY FORMATION: A CASE STUDY USING THE AM PROGRAM, in J.Hayes, D.Michie, L.I.Mikulich (eds) Machine Intelligence, 9 (1979) 251-283

[20] D.W. Loveland AUTOMATIC THEOREM PROVING: A LOGICAL BASIS, North Holland 1977

[21] D.W.Loveland AUTOMATED THEOREM PROVING: A QUARTER CENTURY REVIEW, Contemporary Mathematics, **29** (1984) 1-45

[22] A.J.Nevins PLANE GEOMETRY THEOREM PROVING USING FORWARD CHAINING, Artificial Intelligence, **6** (1975), 1-23

[23] R.F.Ritt DIFFERENTIAL ALGEBRA , AMS Colloquium Publications, New York (1950)

[24] Wu Wen-Tsun ON THE DECISION PROBLEM AND THE MECHANIZATION OF THEOREM PROVING IN ELEMENTARY GEOMETRY, CONTEMPORARY MATHEMATICS, **29** (1984) 213-234, ristampato da Scientia Sinica, **21** (2) (1978) Beijing

[25] Wu Wen-Tsun BASIC PRINCIPLES OF MECHANICAL THEOREM PROVING IN ELEMENTARY GEOMETRIES, Journal of Automated Reasoning, **2** (1986) 221-252, ristampato dal Journal of Systems Science and Mathematical Sciences, **4** (3) (1984) 207-235, Beijing

HEURISTICS TO LOCATE THE BEST DOCUMENT SET IN INFORMATION RETRIEVAL SYSTEMS (*)

Dario Lucarella
Dipartimento di Scienze dell'Informazione
Università degli Studi di Milano

RIASSUNTO

Il presente lavoro desrive l'uso di strategie di "best match" in sistemi per la ricerca d'informazioni. In risposta a una data richiesta, la ricerca di "best match" richiede l'identificazione di quei documenti nella raccolta, che sono più simili alla richiesta, dove la similarità è misurata tramite una opportuna funzione di vicinanza.
Dopo un breve cenno ai limiti degli attuali sistemi di ricerca da cui si prende spunto per descrivere un'approccio alternativo, l'enfasi è posta sulle euristiche utilizzate per localizzare efficentemente l'insieme di documenti più vicino alla richiesta.
La descrizione del problema inizia con un riferimento ad una procedura di ricerca dei migliori documenti, a partire da una manipolazione degli indici di accesso invertiti. Successivamente, viene introdotto un miglioramento nell' algoritmo che si basa sul calcolo anticipato di un limite superiore relativo alla vicinanza, in modo da evitare il calcolo esatto della vicinanza per ogni caso, ottimizzando pertanto sia il numero di documenti da valutare che il numero di liste invertite da ispezionare.
La parte finale dell'articolo analizza in dettaglio l'algoritmo e riporta i resultati finali ottenuti.

ABSTRACT

This paper discusses the use of best match search strategies in information retrieval systems. In response to a given query, best match searching requires the identification of those documents in the collection which are most similar to the query, with similarity being measured by an appropriate closeness function.
After a brief remark on the limitations of conventional retrieval systems which suggest this alternative approach, the emphasis is on heuristics to efficiently locate the closest documents set.
The problem is introduced with reference to a straightforward search procedure that returns the best documents manipulating inverted index entries. Then, an improved algorithm is presented which computes in advance an upperbound on closeness avoiding the exact computation of closeness in many instances and thus optimizing both the number of documents to be evaluated and the number of inverted lists to be inspected. Finally, the algorithm is analyzed and experimental results are reported.

(*) This paper has been presented at the Eigth annual IEEE Conference on Computers and Communication, Scottsdale, Arizona, May 22 - 24, 1989

1. INTRODUCTION

The provision of responsive and informal retrieval systems becomes a more and more desirable goal with the increasing possibility for cheap access to computers and the wide circulation of large document collections on optical disks [1]. Conversely, most of document retrieval systems currently in use are based upon Boolean models and exact match searching although the considerable disadvantages of this model have been frequently pointed out [2], [3], [4].
End users asking for information have often a vague and incomplete notion of their information needs so that it becomes difficult to formulate their queries in terms of a rigid and unnatural Boolean notation. Furthermore, it is not easy to get the right level of focus for the queries without any good knowledge of the document base content. The use of general terms with poor filtering capabilities leads to retrieve a lot of items whereas the use of specific terms may be too selective yielding no documents in response.
A central problem with Boolean searching is that documents are retrieved when the set of associated index terms matches exactly the combination of search terms specified in the query. A document either qualifies or does not qualify and the document collection is simply partitioned into two sets: documents that satisfy the query and document that do not satisfy. So, the effect of a Boolean query can be questioned: in response to an or-query, a document containing one query term is as good as an item containing all of the query terms; in response to an and-query, a document containing all but one of the query terms is as bad as an item containing none of the query terms. In addition, all of the retrieved items are supposed to be of equivalent importance and the order in which they are returned to the user does not reflect any presumed usefulness. Even if many factors influence in a complex way the user judgment of relevance, the system should be able to make an assessment as to whether or not a document is relevant to the user needs and rank returned documents in order of their estimated usefulness.
Last but not least, there is no means in order to take into account the different contribution of each term to the content characterization. Conversely, research in this area has proved that the ability of assigning importance weights to both document and query terms may significantly improve the retrieval effectiveness. In order to overcome the limitations of conventional Boolean systems, alternative retrieval models have been studied [2]. In this context, a retrieval strategy based upon best match searching could help to remove many of the reported problems [5]. Best match searching involves the identification of those documents in the collection which exhibit a higher degree of similarity with the submitted query. There is no need for the user to specify Boolean interconnections between terms and it is possible to take into account different term contribution to document and query characterization. Returned documents can be ranked in order of decreasing query-document similarity with the size of the retrieved document set under control.

A critical point with models based on similarity evaluation is the high number of query-document comparisons that must be carried out in order to retrieve relevant documents. Particularly, this aspect is amplified if you think of large document collections on slow access optical storage. Perhaps, this is the main reason that has prevented the implementation of systems based upon this strategy and that has motivated many researchers in the last few years to devlop heuristics to efficiently locate the closest document set. The objective of the present paper is to give an additional contribution in this direction.

2. BEST MATCH SEARCHING

Let D be the document collection consisting of n documents and T the system dictionary consisting of m index terms used for content identification. Define $I(D_i) \subseteq T$ an indexing function which, given the text excerpt of a particular document D_i returns a subset of weighted index terms characterizing the content:

$$I(D_i) = \{(t_{ij}, w_{tij}) \mid 1 \leq j \leq m; t_j \in T\}$$

where t_{ij} represents the assignment of term t_j to the document D_i and w_{tij}, is a real number in the interval [0,1] which reflects the importance of the term t_j for qualifying the document D_i. The same indexing function can be applied to the text of the query Q in order to get a comparable query representative:

$$I(Q) = \{(q_j, w_{qj}) \mid 1 \leq j \leq m; q_j \in T\}$$

In this setting, we assume a document is about a term, hence its associated concept, if that term occurs in the document representative. So the set of documents about a particular term t_j is given by:

$$D_{tj} = \{D_i \mid t_j \in I(D_i)\}$$

In the following, this will be referred to as the inverted list associated to the term t_j giving the set of documents indexed by such a term. Given a query Q, the set P_Q of documents which possibly satisfy the query is given by:

$$P_Q = \bigcup \{(D_{qj} \mid q_j \in I(Q)\}$$

The set P_Q represents the union of the inverted lists

associated to each of the query terms, i.e. the set of documents which share at least one term in common with the query. The set R^r_Q of the r documents which best satisfy the query is given by:

$$R^r_Q = \{D_i \mid S(D_i, Q) \geq h; D_i \in P_Q\}$$

where S is a similarity function which returns a real number such as a high value implies a high degree of resemblance and h is a threshold value. A ranked output can be obtained arranging the retrieved items in decreasing order of query-document similarity as measured by the S-values.
Now, let us define in further detail the weighting functions used for documents and queries respectively and the similarity function used to select best matching documents. Given each document, the indexing function must produce a document representative in the form mentioned above where each document D_i is represented by a set of pairs (t_{ij}, w_{tij}).
The weighting function used in our experiments, is based on the occurrence frequency of each term in the document excerpt. A measure that has been tested with good results is the augmented normalized term frequency [6], [7]. The weight w_{tij} which reflects the presumed importance of term t_j for qualifying the content of document D_i is defined as w_{tij} $(0.5 + 0.5F_{ij}/Fmax_i)$ where1 Fij represents the occurrence frequency of the term t_j in the document D_i normalized by $Fmax_i$, the maximum occurrence frequency among the terms associated to D_i. The effect of such a normalization being that longer documents do not produce higher term weights than shorter ones. The same indexing and weighting process is performed on the natural language text of the query producing a query representative consisting of a set of pairs (q_j, w_{qj}) with w_{qj} denoting the degree of importance of the term q_j. In this case, the weight attached to each query term is determined by its IDF (Inverse Document Frequency) value which can be proved under appropriate assumptions to be equivalent to the term relevance weight introduced in probabilistic retrieval models. Such a weight has been shown to give consistently good results with a range of different document test collections [7]. For each term j, it is computed as $IDF_j = \log n/n_j$ where n is the number of documents in the collection and n_j is the number of documents in which the term j appears, that is the cardinality of the set $|D_{tj}|$ defined above. Essentially query terms are assigned weights inversely proportional to their frequency of occurrence in the collection; thus, infrequently occurring terms are assigned larger weights than are terms which occur in many documents. Such weights express the value of a term for searching as they indicate how useful it is to differentiate the document from the collection.
During the matching process, the effect of this approach is to combine the local occurrence of a term within a document, which reflects its importance, with the total occurrence of the same term within the collection, which reflects its discrimination power.
A retrieval operation involves the identification of documents most similar to the submitted query. A range of matching functions is available in order to measure the query-document closeness. They are normally computed as a function of the number and the weights of attributes in common between the document and query descriptors [8]. Thus the similarity of a document D_i with respect to the query Q consisting of (q_1 ,..., q_k) terms is given by:

$$S(D_i, Q) = \frac{1}{F} \sum_{j=1}^{k} w_{qj}\, w_{tij}$$

Several modifications are available depending on F. It is a normalization factor taking into accounts respectively the query and document descriptor lengths in order to avoid that documents with more terms reach higher similarity values and have a better chance of being retrieved than documents with fewer terms.
A general discussion on similarity functions, out of the objectives of the present paper, can be found in the literature [8] while further details concerning the suitability of the cosine normalization factor used in our experiments can be found in [9].

3. A BASIC SEARCH PROCEDURE

The straightforward means of carrying out a best match search is to match the query against each of the documents in the collection, compute the similarity, sort the similarities into descending order and take out the r highest-ranking documents. Obviously, it requires O(n) computations which is impractical for large real collections.
Hence there is a need for organizing the document set D so that, given the query, the search for the closest set can be answered efficiently without the need to inspect

all of the documents. The objective is to eliminate from further consideration as soon as possible those documents which fail to comply with the submitted query.

One approach consists in preprocessing the collection partitioning it in clusters, each cluster consisting of similar documents and a representative of the included documents. The first stage of retrieval process consists in finding those clusters whose representatives are most significantly correlated with the given query and then the query is matched against each document contained in the identified cluster. The other approach followed here is based on the availability of an inverted file giving for each query term the set of documents to which that term has been attached as content attribute. Besides the difficulties to develop effective clustering methods, it has been proved by Voorhees that an inverted file organization is more efficient [10].
The availability of the inverted file definitely reduces the number of documents that must be considered as potential candidates for best matching. Most of the common similarity functions that have been used in retrieval systems, including the cosine correlation used in this experiment, involve the calculation of terms in common between the query and the document.
Hence $S(D_i, Q) \neq 0$ iff the query and the document vectors have at least one common term. It means that we have to evaluate only the set $D_i \in P_Q$ of documents which appear at least once in the postings corresponding to the query terms while all other documents can be discarded.
Several researchers have described ranking algorithms based upon the inverted file organization. A review of these search algorithms has been provided by Perry and Willet [11]. Murtagh, starting from a first method developed by Smeaton and VanRijsbergen [12], has described an algorithm which evaluates only a subset of possible documents [13]. Unfortunately, this class of procedures although optimized, requires a large number of accesses to the document file.
A first algorithm that reaches the goal of eliminating the accesses to the document file was proposed by Noreault and is discussed in [11]. The basic idea is to process the query lists, but when a document is encountered for the first time in the list, no attempt is made to match the corresponding document descriptor thus avoiding the access to the document file. Instead a counter is allocated to such a document and set to one. When a document appears once again in a subsequent list, its counter is incremented by one. The end result is to have in each counter the number of matching terms between the document and the query. If the terms have attached weights, as in our setting, the counters will be incremented by the product of such weights. The result being that the counters will contain the inner product between the document and query descriptors. An evident advantage of this procedure is that we have no access to the document file if a suitable similarity function is used. Obviously, in the inverted file entries, we have to store not only the references to the documents indexed by the term but also the corresponding weights.
Considering this algorithm is the basis of further improvements developed in this paper, the pseudo-code is reported and discussed.

```
Procedure Search;
  for each QueryTerm q_j do
    Read Inverted List; {composed of pairs (D_i, wt_ij)}
    for each Document D_i in the list do
      if NewDocument then
        AllocateCounter C (D_i);
          C (D_i) : = 0;
      endif;
      C (D_i) : = C (D_i) + (w_qj * w_tij);
    endloop;
  endloop;
  Sort C (D_i) in decreasing order;
  Present the top r documents;
end Search.
```

The presented procedure computes the document-query similarity for each $D_i \in P_Q$ that is for each document which has a non-zero value for the S-function. Such a set may be considerably large. Perry and Willet, experimenting with several document collections, have shown in [11] there is a considerable number of documents having a very small similarity with to query.
That implies the useless calculation of many low-value similarities for documents that will not reach the closest set. The result being that a large fraction of the document collection must be taken into account. Starting from these considerations further improvements have been achieved by Harper who describes an algorithm using inverted files in which similarities are computed only for documents which are expected to enter the closest set [14]. Conversely Buckley and Lewit propose an algorithm for finding the closest set computing partial similarities for all the involved documents but

taking into account only useful inverted lists [15]. This last approach has been followed also by Loose for the identification of nearest neighbors in a probabilistic retrieval model [16].
This paper considers an approach in which partial similarities are maintained only for documents which are presumed to reach the closest set and the search procedure is stopped as soon as a minimum number of inverted query lists has been examined enough to guarantee the retrieval of best documents.

4. A HEURISTIC PROCEDURE

The considerations developed before suggest that further improvements can be achieved if we minimize both the number of documents to be evaluated and the number of inverted lists to be inspected. Particularly, with reference to the second point, if we find out, before completion, that we have already determined the closest set then the algorithm can be stopped and the remaining lists need not be examined.
As a first step, the query terms are sorted in order of decreasing weight. As the term weight is determined by its IDF value, the effect is to have those terms which apply to few documents at the top. Consequently, we process the shorter inverted lists first leaving at the bottom the lists which include a larger number of documents. Then we start to process the query lists in this order. Let us assume we have processed l query lists out of k and we have a current closest set R including the documents $D_1,...D_r$ in decreasing order of similarity.
We have to process now the $(l + 1)^{th}$ query list. For each document D_i referenced in the list, an upperbound for the similarity value can be computed assuming D_i should happen to match all of the remaining query terms. If the computed upperbound for D_i is less than the similarity value associated to D_r it means the document D_i will never reach the relevance set so that it can be removed from further consideration.
Moreover, we can terminate the algorithm if the best document out of the relevance set R is not expected to give a similarity better than the last document in the set R, that is D_r.
A modification of the previous algorithm to include these improvements is reported. The location "BestDoc0ut" is maintained and updated properly when documents are entered in the document table; it will contain the document with the highest score among the ones loaded into the document table.
As in the basic algorithm, the procedure " Compute $C(D_i)$ " allocates the counter when the document is a new one and then computes the partial similarity as $C(D_i) := C(D_i) + (w_{qj} * w_{tij})$. In the reported code, it is assumed that the "RelSet" is maintained in decreasing order of partial similarity and that the procedure "Enter D_i into the RelSet" inserts the item properly removing the last one so that D_r will always represent the current last document in the relevant set R.

```
Procedure Search;
  sort QueryTerms in decreasing order of weight;
  repeat {for each query term q_j}
   Read InvertedList; { composed of pairs (D_i, w_tij)}
   for each Document D_i in the list do
    if RelSetNotFull then
       Compute C (D_i);
       Enter D_i into the RelSet;
    else
       Compute U (D_i);
       if U(D_i) ≤ C (D_r) then
          DoNotAllocate/Remove Di
       else
          Compute C (D_i);
          if C (D_i) > C (D_r) then D_i in RelSet endif;
       endif;
    endif;
   endloop;
    Compute U (BestDocOut);
  until LastQuery Term or U (BestDocOut) ≤ C(Dr)
  Present the RelSet sorted in decreasing order;
end Search.
```

Let us look now at the evaluation of an upperbound $U(D_i)$ for the total query-document similarity, given a query Q consisting of $(q_j,...q_k)$ terms and given the document D_i.
After having processed the first l query lists, let s_i be the current score for the document D_i. Then, the total similarity can be bounded according to the following relation:

$$S(Q, D_i) \le s_i + \frac{1}{F} \sum_{j=l+1}^{k} w_{tij} * w_{qj}$$

The summation $\sum_{j=l+1}^{k} w_{tij} * w_{qj}$ corresponds to the maximum remaining similarity, assuming the worst case that all the remaining un-inspected terms (k - *l*) are in common between the document and the query.
Since the weighting scheme in effect computes the

document term weight as the intra-document normalized frequency, the document term weight w_{tij} is bounded by 1.0. It follows that:

$$S(Q, D_i) \le s_i + \frac{1}{F} \sum_{j=l+1}^{k} w_{tij} * w_{qj} \le \frac{1}{F} \sum_{j=l+1}^{k} w_{qj}$$

Hence an upperbound for $S(Q, D_i)$ can be defined as:

$$U(D_i) = s_i + \frac{1}{F} \sum_{j=l+1}^{k} w_{qi}$$

The result being that we can compute $U(D_i)$ taking into account only the weights of the remaining query terms which are already known.
The effects of the optimization can be magnified if we accept that only the top r' documents (r' <r) are guaranteed to be the best ones while the remaining (r - r') are simply good matches. It implies changing, in the previous algorithm, the stopping condition to U(BestDoc Out)≤ C(Dr'). Clearly, as C(Dr')>C(Dr), the stopping condition works better dropping an higher number of query lists. It has been experimented by Lucarella in [4] that recall values of the retrieval system do not degrade significantly if, given the cardinality of the returned set, only a subset is strongly ranked.

5. ANALYSIS OF THE ALGORITHM

The running time of the reported algorithm is influenced by the number of the inverted lists (which depends on the number of query terms) and by the number of entries in such lists. So the worst condition means scanning all of the lists when the stopping comparison is not satisfied and the computing cycle is not terminated before completion.
We define N_q the average number of terms assigned to each of the queries, N_d the average number of terms assigned to each of the documents and M the average number of documents associated to a term (that is the average number of postings in the inverted lists). In the worst case the expected running time of the algorithm is of order:

$$O(N_q * M)$$

Considering we had defined n the total number of documents in the collection and m the number of distinct index terms (i.e. the total number of inverted file lists), it follows that $N_d * n$ gives the total number of postings. Hence, the average number of postings in the lists is $M = N_d * n/m$, the total number of postings divided by the number of inverted file lists. Then the previous expression becomes:

$$O(N_q * N_d * n/m)$$

If we assume roughly $N_q = N_d = N$, we have

$$O(n * N^2/m)$$

that shows the dependence on N, i.e. the indexing exhaustivity. A general discussion about the influence of indexing exhaustivity on the efficiency of inverted file algorithms can be found in [17] and partially in [13].
In order to make experiments and test the efficiency of the algorithm, a document collection in the area of Applied Mathematics and Computer Science has been used [9]. The collection was available in machine readable form at the Enel, the Italian Electricity Board, and relevant characteristics are reported in the first part of the table.

Number of documents	1500
Number of terms	2137
Number of queries	50
Average terms per document	14.8
Average terms per query	7.2
Average docs. referenced per query	354.5
Average docs. processed per query	78.3
Average dropped lists per query	0.27

Table 1. Experimental Results

It is interesting to compare, on average, the total number of documents which would have been processed without any optimization, with the number of documents really maintained in the table as a combined effect of the two introduced optimizations. Furthermore, it is interesting to take note of the mean number of dropped query lists considering that such documents are not examined at all.
A set of fifty queries in natural language has been used. The set has been indexed with query terms weighted by

the IDF function. Each query was processed twice, first with the basic algorithm and afterward with the improved version. The returned relevant set consisted of ten good documents (r = 10) while only five documents (r' = 5) were guaranteed to be the best ones. Results are reported on average in the second part of the table. The entry "documents referenced per query" gives the total number of distinct documents indexed by the query terms. The entry "documents processed per query" gives the actual number of documents placed into the table for which partial similarities are evaluated. The last row gives the percentage of inverted lists which are dropped as a consequence of the stopping condition. A value of 0.27 means that, on average, 27% of the inverted lists are not processed. Note that the dropped lists are the longest ones since the query terms are sorted in order of decreasing weight and the weighting scheme in effect assigns the lowest weights to terms which occur in many documents.
The conclusion was of a meaningful decrease in the number of documents to be processed as well as in the number of query lists to be examined. A negligible effect of the introduced optimization is a possible decrease in the ordering precision among the retrieved documents since they are ranked in order of their partial similarity instead of the total similarity to the query.

6. CONCLUSION

In the previous pages, we have discussed the use of best match search techniques in information retrieval systems. Consequent enhancements in terms of system capabilities and user interface have been underlined and an effective and easy implementation of this search strategy has been shown.
An automatic indexing procedure with a weighting scheme to reflect term importance has been applied to both document excerpts and natural language queries. Documents have been stored using an inverted file scheme and an efficient search strategy has been presented. The proposed algorithm returns the relevant document set computing an upperbound on closeness thus eliminating the need for an exact computation of closeness in many instances. The result is an improvement in both the number of documents to be evaluated and the number of inverted index entries to be inspected.
As already remarked, some of the reported improvements assume a particular meaning in connection with large document databases and slow access optical storage. We can expect an increasing importance of such retrieval strategies in the future with the wide circulation of optical disks and the consequent availability to end users of large machine readable document collections.

7. REFERENCES

[1] E.A.Fox, INFORMATION RETRIEVAL: RESEARCH INTO NEW CAPABILITIES, in: S. Lambert, S. Ropiequet, eds., CD-ROM: the new papyrus (Microsoft Press, Redmond, 1986) 143-174.

[2] G. Salton, A NOTE ON INFORMATION RETRIEVAL MODELS AND THEORIES, Proc. RIAO 85 Recherche d'Information Assistèe par Ordinateur, Grenoble, France, 18-20 March (1985) 1-27.

[3] P. Willet, RANKED OUTPUT SEARCHING IN TEXTUAL AND STRUCTURAL DATABASES, Proc. Ninth Online Information Meeting, London, UK, 3-5 December (1985) 343-353.

[4] D. Lucarella, A SEARCH STRATEGY FOR LARGE DOCUMENT BASES, Electronic Publishing vol. 1, n. 2 (1988) 105-116.

[5] D. Lucarella, BEST MATCH SEARCHING IN DOCUMENT DATABASES, Proc. Fourth Int. Conf. on Text Processing Systems Boston, MA, October 20-22, (1987) 102-110.

[6] C. Buckley, IMPLEMENTATION OF SMART INFORMATION RETRIEVAL SYSTEM, Techn. Report 85-686, Dept. Comp. Science, Cornell Univ., (1985).

[7] G. Salton, ANOTHER LOOK AT AUTOMATIC TEXT RETRIEVAL SYSTEMS, Communications ACM 29 (1986) 648-656.

[8] G. Salton and M.J. McGill, INTRODUCTION TO MODERN INFORMATION RETRIEVAL, McGraw-Hill, New York, (1983).

[9] D. Lucarella, A DOCUMENT RETRIEVAL SYSTEM BASED ON NEAREST NEIGHBOUR SEARCHING, Journal of Information Science 14 (1988) 25-33.

[10] E. Voorhees, THE EFFICIENCY OF INVERTED INDEX AND CLUSTER SEARCH, Proc. ACM Conference on Research and Development in Information Retrieval, Pisa, Italy, 8-10 September (1986) 164- 174.

[11] S.A. Perry and P. Willet, A REVIEW OF THE USE OF INVERTED FILES FOR BEST MATCH SEARCHING IN INFORMATION RETRIEVAL SYSTEMS, Journal of Information Science 6 (1983) 59-66.

[12] A.F. Smeaton and C.J. vanRijsbergen, THE NEAREST NEIGHBOUR PROBLEM IN INFORMATION RETRIEVAL, Proc. Fourth International Conference on Information Storage and Retrieval, Oakland, CA, May 31-June 2 (1981) 83-87.

[13] F. Murtagh, A VERY FAST EXACT NEAREST NEIGHBOUR ALGORITHM FOR USE IN INFORMATION RETRIEVAL, Inf. Tech.: Research and Development 1 (1982) 275-283.

[14] D.J. Harper, RELEVANCE FEEDBACK IN DOCUMENT RETRIEVAL SYSTEMS: AN EVALUATION OF PROBABILISTIC STATEGIES, PhD. Thesis, the University of Cambridge, 1980).

[15] C. Buckley and A.F. Lewit, OPTIMIZATION OF INVERTED VECTOR SEARCHES, Proc. Eight International ACM Conference on Research and Development in Information Retrieval, Montreal, Quèbec, 5-7 June (1985) 97-110.

[16] R. Loose, PROBABILISTIC RETRIEVAL AND COORDINATION LEVEL MATCHING, Journal of ASIS 38(4) (1987) 239-244.

[17] A.F. Harding and P. Willett, INDEXING EXHAUSTIVITY AND THE COMPUTATION OF SIMILARITY MATRICES, Journal of ASIS 31 (1980) 298-300.

NOTE DI SOFTWARE 44/45

A KNOWLEDGE-BASED APPROACH TO AN OPERATING SYSTEM FOR A LARGE COMPUTER SYSTEM (*)

J. Takamura, K. Nobusawa, Y. Ebino, A. Date, E. Yoshikawa
NEC Corporation
Fochu, Tokyo, Japan

RIASSUNTO

Per utilizzare al meglio e per operare con un sistema complesso sono richiesti vari tipi di riconoscenza. "Composer" è un sistema basato sulla conoscenza con le conoscenze necessarie richieste per le operazioni e l'utilizzazione come parte di un sistema operativo mainframe e come supporto nell'analisi delle prestazioni, nello sviluppo di programmi e per la diagnosi di guasto. Un approccio Kuonledge-based è stato introdotto per perseguire gli scopi di un sistema operativo, che sono le convenienze per l'utente e l'efficenza per migliorare le prestazioni del sistema. La prima versione di "Composer", che include alcune di queste caratteristiche, è già stata realizzata ed è attualmente in uso.
In questo articolo vengono descritti a titolo di esempi molti metodi di problemi solving quale la diagnosi, la generazione di soluzioni che soddisfano vincoli, manutenzione in tempo reale della consistenza di una base della conoscenza , la comprensione di descrizioni di un linguaggio di controllo e la gestione di inconsistenza tra imput specificati dall'utente. L'attuale implementazione di "Composer" viene eseguita su mainframe NEC, a parte il sistema operativo ACOS - 6 /MVX. I resultati di "Composer" possono essere valutati dal punto di vista sia qualitativo, sia quantitativo. Uno dei resultati qualitativi è che il tempo per la soluzione di problemi calcolato manualmente è stato significativamente ridotto. Per esempio i compiti di analisi di prestazione di solito richiedono circa due giorni per la raccolta e l'analisi di una quantità di dati, una "Composer" fornisce la soluzione in pochi minuti. Qualitivamente, la sua maggiore forza consiste nelle facilitazioni per i livelli di operatività e di utilizzazione.

(*) This paper has been presented at the Eigth annual IEEE Conference on Computers and Communication, Scottsdale, Arizona, May 22 - 24, 1989

ABSTRACT

Various kinds of knowledge are required in order to best utilize and operate a complex computer system. "Composer" is a knowledge-based system with the necessary knowledge required for such operation and utilization, and as part of a mainframe operating system, assists in tasks of operation and utilization such as system performance analysis, troubleshooting and program development. A knowledge-based approach has been introduced to pursue the goals of an operating system, which are convenience for the user and efficient operation to improve performance of the computer system. The first version of "Composer" which includes some of these features is already released and in actual use. In this paper, several methods for solving problems of operation and utilization such as diagnosis, generation of solutions satisfying constraints with different strength, real-time consistency maintenance of a knowledge base, understanding of control/programming language descriptions, and the detection and management of inconsistency among user-specified input, are described as examples. Current implementation of "Composer" runs on NEC mainframe computers as part of the operating system ACOS-6/ MVX. The "Composer's" result can be evaluated from quantitative and qualitative viewpoints. One of the quantitative results is that the time for solving problems previously done by hand has been significantly reduced. For example, performance analysis tasks usually took about two days in collecting and analyzing a large amount of data, but "Composer" provides the solution in a few minutes. Qualitatively, its major strength is that it facilitates high-level operation and utilization.

1. INTRODUCTION

A computer system becomes more difficult to operate and use as it becomes more complex and its functionality increases. We have been developing a knowledge-based system called Composer (an expert system for Computer operation and user assistance) which helps with the utilization and operation of the computer system.

An operating system is an interface between a user and the computer hardware in order to provide an environment for the user to run application programs. Therefore, the goals of an operating system are convenience for the user and efficient operation to improve performance of the computer system.

In order to achieve the goals of an operating system, knowledge concerning operation and utilization is gathered and stored into the knowledge base. This knowledge may come from the various expert users, such as system administrators, operators, systems engineers, systems programmers, and end-users who normally use the computer system. The working system, which enables high-level utilization for the user who does not have such knowledge, assists the normal operational staff, automating as much as possible. There are already several expert systems relating to computer systems: e.g. one that assists systems engineers in configuring computers[1], a system that assists in or automates an operator's task[2,3], and a system that assists supervisors in computer network troubleshooting[4]. Our goal, however, is not to develop an expert system that replaces the knowledge of a particular expert. Instead, our goal is to realize an intelligent operating system.

This paper is an overall description of the system based on the first implementation using several subdomains selected from customer requests. We also describe several examples of problems concerning operation and utilization, and AI techniques used to solve those problems.

2. ASSISTING OPERATION AND UTILIZATION

There are a number of problems involved in operating a computer system. First, operation begins with system design and system generation. Second, there are various problems concerned with scheduling programs and troubleshooting while the system is operating. Finally, there are system evaluation requirements in order to operate the system efficiently and improve performance. Utilization work also includes program development, debugging, and execution. Communication with the computer using languages, etc. is another important issue.

Most of these problems are currently solved by humans. These problems must be solved in person through specific commands and computer languages. Therefore, long-term experience is necessary, along with various knowledge such as general knowledge concerning computers and knowledge of computer languages and commands. The amount of knowledge required, plus its complexity makes the entire process very difficult.

Part of the reason that humans need to give so many orders and select options is because a conventional operating system is designed to be general-purpose, and it does not have knowledge which is not general (e.g. site-dependent operational policy). So, in many cases, what is not common must be specified from outside the system. Another reason for so much human intervention is because operation and utilization problems are generally not well-structured, and the solutions can not be described algorithmically. Namely, ways of operation and utilization depend on each site or user, and also vary according to the situation. Ill-structured problems like these are generally better handled by Artificial Intelligence.

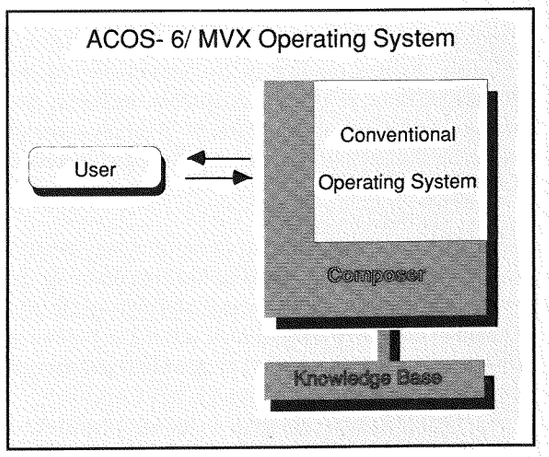


Fig.1

Following from this basic idea Composer has been developed with a knowledge base for operation and utilization knowledge. In addition, it contains the knowledge to solve high-level problems of operation and utilization that normally would require a human user, as shown in Figure 1. Owing to Composer, the user's load of operation and utilization is reduced and high-level operation and utilization is facilitated.

3. AN OVERVIEW OF *Composer*

Composer runs on NEC mainframe computers, S1000, S2000 etc., as a part of the ACOS-6/MVX operating system. ACOS-6/MVX is an operating system for general-purpose mainframe computers, and it supervises multiple dimensions such as TSS, batch processing, transaction processing, and controls a large scale computer system connected with various devices through a network.

As Composer is tightly connected to other operating system components, basically it runs on the host computer. Namely, it needs to have free and fast access to a large number of data on the system tables, log data, trace data, console message data, etc.. In case of need, Composer runs in the privileged mode and has access to these data and acquires necessary information into the knowledge base. In parallel with the development of Composer, necessary data have been reviewed throughout the operating system, and the operating system has been modified to collect important but lacking data.

Composer is mostly written in LISP dialect called UTILISP. Access to data in the kernel part of the operating system, or when calling other operating system modules, control is processed through routines written in system description languages and linked to the UTILISP program.

As the first implementation of Composer, we selected the following five subdomains in the domain of operation and utilization, and have been developing the system which can solve a significant set of the following problems. The first three relate to operation and the rest relate to utilization.

(1) System performance analysis
System performance such as response time or turnaround time is analyzed by manual activation or by automatic monitoring. If a performance problem is detected, the cause of the problem is diagnosed and a solution to achieve the performance goal is generated. There are two modes in performance analysis: long-term and real-time. Long-term analysis checks monthly or daily data using a long-term viewpoint. The real-time analysis monitors the performance in real-time, and it begins performance analysis automatically using real-time data.

(2) *Network troubleshooting*
The cause of the fault occurring in a communication network such as hardware error and installation error is diagnosed and advice is given for appropriate action to recover from the fault. Namely, information such as log data, configuration data and trace data is given to diagnose the cause of fault occurred at the host, communication network, terminals and so forth.

(3) *Database operation assistance*
Database design is assisted in many ways, e.g., calculation of storage capacity, generation of schema structure figure and so forth. Also, while the database is in operation, performance problems of database are detected by using storage information and execution data. The cause is diagnosed and appropriate action is recommended.

(4) *Job description assistance*
When the user describes batch jobs using JCL, errors and inadequate descriptions are detected using an understanding of JCL descriptions. Messages are given to explain how and why they need to be modified. In addition to the JCL knowledge, the knowledge base stores general knowledge concerning the computer system and know - how to make better JCL descriptions. The automatic generation of JCL code is also planned.

(5) *Program debugging assistance*
Program debugging is assisted by advising how to detect bugs more efficiently. Knowledge required for debugging FORTRAN77 program is currently stored in the knowledge base, and when an erroneous result or error message is produced, the debugging task is made easier and more efficient by advising what to do and where to look for the bug.

Each subsystem can be run by TSS or center console commands. Composer's inference mechanism was developed using expert system shell called ESPA, which we developed for internal use. Several additional problem solving mechanisms described below were written in UTILISP.

4. SOLVING PROBLEMS OF OPERATION AND UTILIZATION

Problems of operation and utilization can be categorized into several types (e.g. diagnosis). As already described, we have been developing the system, choosing several subdomains for the first implementation. In some cases, problem type may differ if subdomains are different, but there are inference methods and knowledge common to various subdomains. In order to realize an intelligent operating system, basic problem solving mechanisms for the domain of operation and utilization must be assorted. As examples of such problem solving mechanisms, diagnosis, generation of solutions, consistency maintenance of the knowledge base, understanding of computer language descriptions and detection of inconsistency among user-specified input are described below, citing examples of network troubleshooting, system performance analysis and job description assistance.

4.1. Diagnosis

In cases such as problems caused by performance or in breakdowns on the network, it becomes necessary to diagnose the cause of the problem. In performance analysis, for example, the scope of the problem is so huge, and the number of separate modules can be so complicated that a mathematical modeling of the system becomes very difficult. For that reason, instead of solving a performance problem through mathematical techniques such as queueing theory or simulations, a normal expert will rely mostly on rules of thumb and other knowledge in order to do the diagnosis. Following from this basic idea, we developed a rule-based inference mechanism for the diagnosis[5].

In the domain of computer operation and utilization, the above features of the diagnosis are common to other subdomains, for example, network troubleshooting. Therefore, we are now developing a more powerful diagnosis mechanism common to the domain of computer operation and utilization as a part of the development of the network troubleshooting subsystem. The inference model for the diagnosis used in the prototype subsystem consists of 3-level inference mechanisms, i.e., inference based on empirical knowledge, analytic inference using a diagnosis tree and inference based on deep knowledge[6].

(1) *Inference based on empirical knowledge (level1)*
This inference directly infers an intermediate or final conclusion from the current situation by empirical knowledge without logical thought. Although this inference is faster than next two types of inference, it does not work for inexperienced problems such as problems related to unique system configuration. Therefore, it is used with other types of inferences especially with the next analytic inference. Rules are the simplest knowledge representation for this type of inference.

(2) *Analytic inference (level2)*
This inference enumerates all the possible hypotheses at that point, analyzes them logically using current data, evaluates them relatively and selects the best hypothesis. This process is repeated until the final conclusion is obtained. This type of inference is regarded as a classification problem solving[7]. In our system, a diagnosis tree is used to represent knowledge for this inference. Figure 2 shows an example of the diagnosis tree. It can be also regarded as a search space for the solution.

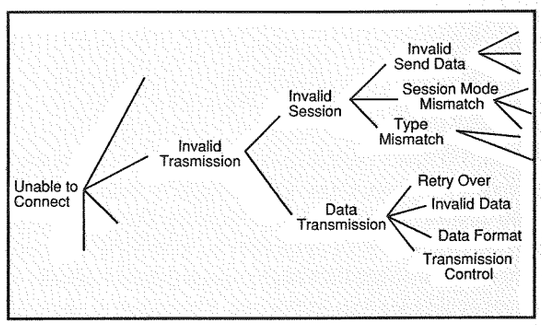


Fig. 2

(3) *Inference based on deep knowledge (level3)*
Level1 and level2 inference cannot solve unexpected problems. In order to solve inexperienced problems, the

inference based on deep knowledge such as knowledge of structure and function of the diagnosed object is required. Our definition of the inference based on deep knowledge is broad and widely includes various types of inference except level1 and level2 inference. We are now studying inference using knowledge of network protocol and system configuration as example of this inference[8].

These three levels of inference are not used separately, i.e., during the diagnosis process the inference shifts from one level to another. This flexible inference method realizes powerful and high-speed diagnosis.

4.2. Generation of solutions satisfying constraints with different strength

When generating solutions to solve a problem, it will normally be important to take into account the differing constraints imposed by each particular site.
For example,

"Reduce to X the number of sessions that can be simultaneously timeshared."

"Increase the number of CPUs by Y."

When generating these sub-solutions, it may be necessary to consider the following sorts of constraints that may be imposed by a particular site:

"Make sure the number of timeshared sessions is greater than 50."

"The number of expansion CPUs is 0."

These kinds of constraints have many levels of relative importance, from those that absolutely must be satisfied to those which are simply requests. Our system will follow the levels of importance, satisfying the constraints by their priority from the absolute constraints to those which are less important. Reference[9] shows a similar approach that generates solutions by successively invalidating constraints. There, the process continues generating solutions until a contradiction occurs, in which case the low-priority constraints are invalidated in order to preserve those of higher priority. Our system is slightly different in that we continue generating solutions until uncovering a performance goal that cannot be satisfied. At that point, constraints are invalidated from weakest to strongest until the goal can be satisfied. To describe this method more specifically:

Assume that S is the solution to a particular performance problem. Since S may involve a number of subsolutions, we can describe S in the following way:

$$S = (S_1, S_2, ..., S_n) \quad (1)$$

To meet a particular performance goal G, solution S will produce an evaluation E that is a function of the machine status M and solution S. The evaluation of this solution must always be greater than or equal to the performance goal. In other words:

$$G \leq E(M, S) \quad (2)$$

For each solution, there are a group of constrains C that may bear on the effect of the solution. These constraints are ordered chronologically based on the order in which they were applied to S.

$$C = (C_1, C_2, ..., C_k) \quad (3)$$

Each C_j has associated with it a priority P_j $(0 \leq P_j \leq 1)$. The system first tries to generate a solution that satisfies all the constraints. If the goal cannot be satisfied, it looks for the constraint with the smallest priority, invalidates it, and backtracks to regenerate the solution. For example, assume that the following constraints were applied when the solution which consists of subsolutions was generated.

$$C = (C_3, C_4, C_2, C_6, C_1) \quad (4)$$
$$0.4\ \ 1.0\ \ 0.6\ \ 0.2\ \ 0.5 \ldots\ldots \text{ priority}$$

The system invalidates C_6 which has the smallest priority if the solution does not satisfy the goal. C_6 is a constraint that means "Detailed trace data should be logged, if possible.". The subsolution that "Stop logging the detailed trace." is newly incorporated into the solution by the invalidation of C_6. This process is repeated until the solution satisfies the goal. When the performance goal is high, and there are many levels of broad constraints that are specified, finding a solution

that will satisfy all the constraints may be difficult. In this case, this system will look for a compromise, the way humans do, and generate a solution that is based as much as possible on the actual intentions of the site.

4.3 Real-time consistency maintenance of the knowledge base

An operational computer system is constantly receiving data and programs from the outside world, generating system events, etc., and the state of the system changes through time. In order to monitor and solve problems in real time, the system's knowledge base needs to update itself and change without contradictions. In other words, the system is non-monotonic, and some data that may have been true at one time might become false as new information is received[10].

To explain, using one example of real-time performance analysis[11], Figure 3 shows a diagram of hundreds of pieces of performance data that were passed to the system at a particular time. For example, information about a system load is changing constantly as the computer operates through time, and inferences that were made using certain system load results will become obsolete and incorrect after a short interval of time.

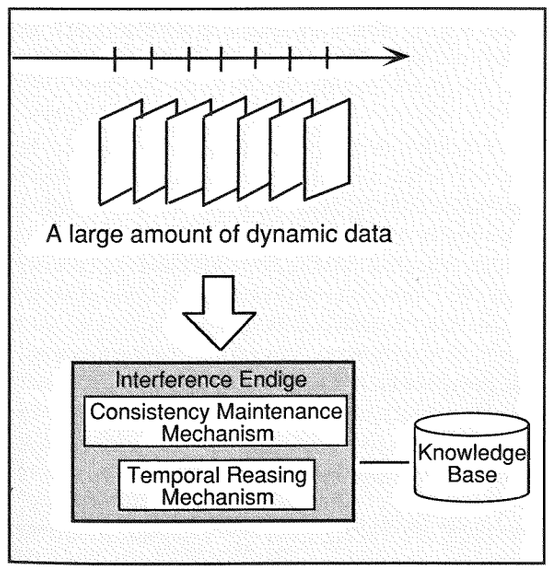


Fig. 3

There are several examples, such as TMS [10], that change themselves in order to maintain consistency, but because of the huge amount of data involved, and the tremendous speed at which that data must be processed, using the traditional TMS system is difficult. For this reason, our system uses a constraint propagation method whereby speed can be maintained and large amounts of data can be processed[12].

Generally speaking, there are two things that must be processed in order to maintain the consistency of the knowledge base. First, when new information is entered that causes a contradiction with old knowledge, the cause of the contradiction must be uncovered and deleted in order to allow the new information to reflect correctly on the knowledge base. In real-time performance analysis, the cause of the contradiction is clear, i.e., the new information. Therefore, the data that caused the change is first updated in the knowledge base, and then all other data that is affected by the changes is propagated through a constraint network.

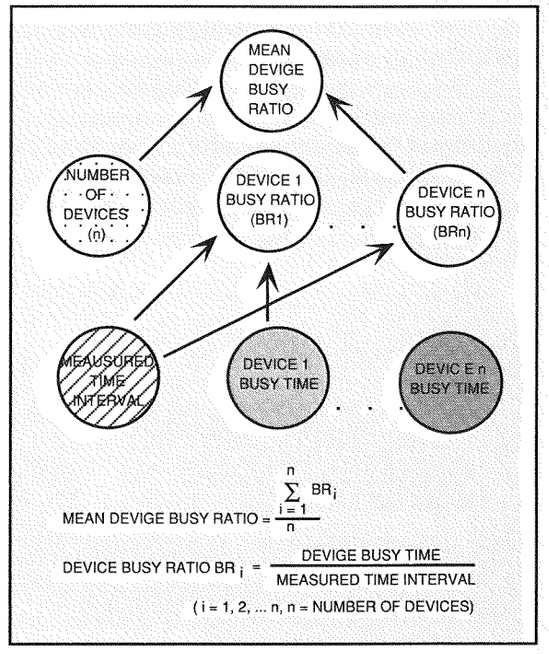


Fig. 4

Figure 4 shows an actual example of this. A device that has a certain busy ratio undergoes a change. That

change is propagated through the constraint network through every part that is dependent on that device, including places that keep track of the average busy ratios, etc. In order to keep track of high-speed changes in the knowledge base, our system first looks through the volume of data and sifts out the non-trivial change. This change is propagated throughout all other places in the knowledge base dependent on it. Finally, in finishing the inference process, inference using time[13,14] as well as deductive reasoning like long-term analysis are also necessary[15]. A set of performance data collected by a resource monitor is provided to the system at regular intervals. If the inference takes longer than the interval, the latest set of data is processed.

4.4. Understanding of computer language descriptions

When some kind of computer language such as programming language or control language like JCL (Job Control Language) is used as means of interface between a user and a computer system, the system must first of all understand the meaning of the language descriptions. Generally speaking, the content described by the user is not always correct. Therefore, it is also necessary to understand descriptions with errors, i.e., errors should be detected and advice given about the correct description. This kind of understanding of a computer language is described below.

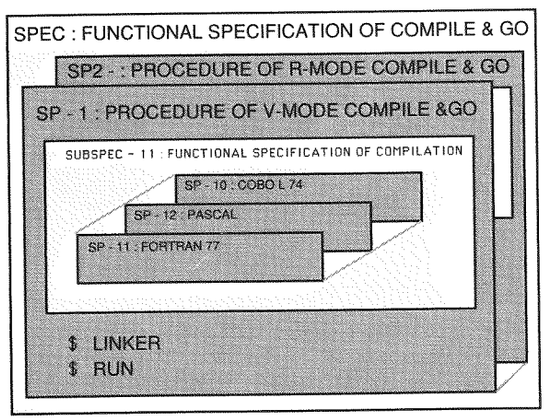


Fig. 5

Generally, in order to perform some processing on a computer, there exists a procedure for the processing. This kind of procedural knowledge is represented using something we call SP (Stereotyped Procedures) which are similar to scripts. Usually, there are several different procedures or algorithms to describe a particular functional specification. In other words, one functional specification corresponds to multiple SPs. In this sense, SP is more close to plans than scripts[16,17].

In the case of ACOS-6/MVX JCL, JCL descriptions consist of a series of sequential descriptions as in an ordinary program. There are several possible descriptions to realize one functional job specification. A job consists of processing units called activities, and an activity again can be regarded as a subfunctional which corresponds to multiple SPs.

For example, Figure 5 is a functional specification of "compile & go". There are two SPs to realize this specification, namely, "compile & go" for a V (Virtual memory) mode job (SP-1) and that for a R (non-virtual memory) mode job (SP-2). The descriptions of "compile & go" for a V-mode job consist of a fixed procedure such as a language control statement, a binding control statement, a linking control statement and an execution control statement. SUBSPEC-11 in SP-1 expresses a subfunction of compilation, i.e., it corresponds to an SP which depends on a computer language, for example, compilation of FORTRAN77. As shown by this example, letting subspecifications be written in one SP enables multiple procedures that are partially different to be integrated into one SP.

Understanding of JCL descriptions is performed by matching actual JCL descriptions with an SP which satisfied the preconditions for activation. The matching is first performed by SPs with correct procedures, and if all the matching fails, it infers the cause of the error using SPs with common error procedures. The SP matcher has powerful pattern-marching functions such as variables, wild cards, string handling functions etc. JCL descriptions are first parsed by the JCL parser before the analysis. Matching of SPs is activated by rules. Rules are also used to represent JCL knowledge other than procedural knowledge, and various types of errors are checked. Frames are used to represent non-procedural knowledge such as knowledge of options which are allowed for a certain JCL, or for structural knowledge. The size of a relatively long JCL description is several hundred steps and some knowledge also involves uncertainty such as "This kind of description is usually wrong".

4.5. Detection of inconsistency among user-specified input

Some elements of user-specified input such as commands or computer language descriptions are dependent on others, and even when descriptions are syntatically correct, they may be semantically inconsistent. It is necessary to detect inconsistency among user-specified input and to infer the intention of the user in order to give advice for correcting it. Taking JCL as an example, there are complicated dependencies among JCL options. For instance, in the case of FORTRAN77 options as shown in Figure 6, when the DlAG option is specified, the IAP option must also be specified. Similarly, if the IAP option is specified, the OPT option must also be specified. Unless all these relationships are satisfied, the user-specified options are inconsistent. As an example of bi-directional dependency, in Figure 6, IAP and ROUND must not be specified at the same time or they are inconsistent. Generally speaking, there are several methods which detect and delete contradictions [18, 19, 20].

However in the case of TMS[18] for example, there are several important differences in the way of solving the problem, and applying TMS without modifications is difficult[21]. For this reason, we use a reasoning method called DCMS (Data Consistency Maintenance System) which detects inconsistency among data and infers the solution that minimizes the influence of the deletion.

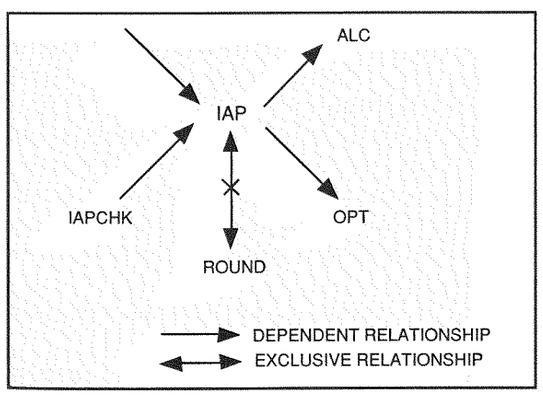


Fig. 6

All the existing nodes processed in DCMS (i.e. specified JCL description data) carry a status of either Correct or Incorrect which is the result of the current interpretation against nodes based on dependent relationships. There is another status called Missing which is interpreted as necessary but omitted data. In the case of a Missing node, if all the nodes which supported the Missing node become Incorrect or deleted, it must also be deleted.

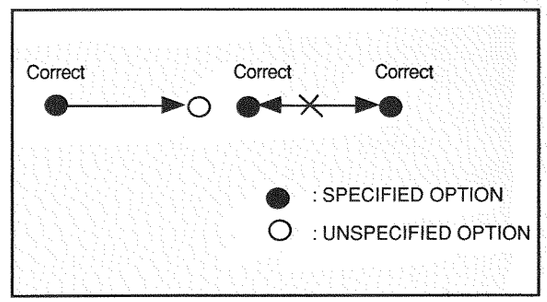


Fig. 7

(1) *Detection of inconsistency*
DCMS interprets all the nodes that the user specified based on the knowledge of dependent relationships. In the beginning, all the specified nodes are interpreted as Correct nodes. DCMS detects the two types of inconsistency shown in Figure 7. i.e., inconsistency is detected if a necessary node is missing or a node which must not exist is specified by the user.

(2) *Deletion of inconsistency*
There are several ways to delete inconsistency depending on which node to deny. The strategy to select a node to deny (i.e. interpret as Incorrect node) is based on the idea that the solution which minimizes the influence is selected.

Namely, both Incorrect nodes and Missing nodes are defined as Error nodes. The strategy is to select a solution which minimizes the number of Error nodes. Generally, there are several inconsistencies among data, so the number of Error nodes is estimated from a global viewpoint, i.e., the number of the Error nodes is estimated supposing all the inconsistencies are deleted. Because, if local inconsistency is deleted separately, global inconsistency may arise as a result of the local solutions.

For example, there are two inconsistencies in Figure 8, i.e., IAP and ROUND must not be specified at the same time, and OPT must be specified if IAP is specified. There are, for instance, two ways to delete the inconsistencies.

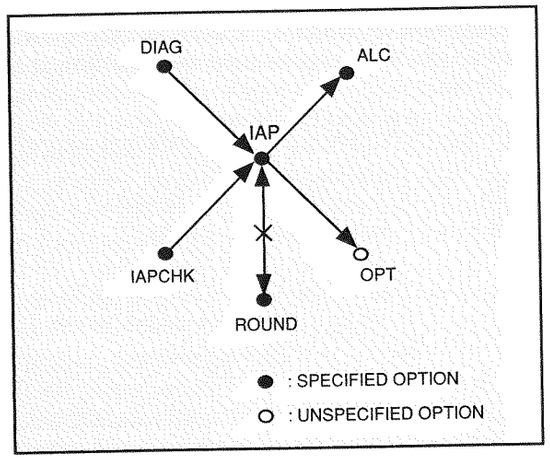


Fig.8

Interpretation 1: ROUND is Incorrect

Incorrect : ROUND
Missing : OPT

Total number of *Error* nodes
= 1 (*Incorrect*) + 1 (*Missing*)
= 2

Interpretation 2: IAP is Incorrect

Incorrect: IAP, DIAG, IAPCHK

Total number of Error nodes
= 3 (*Incorrect*) + 0 (*Missing*)
= 3

In this case, Interpretation 1 is estimated as a better solution than Interpretation 2 from the viewpoint of the total number of Error nodes, and considered to be closer to the user's intention.

5. CONCLUSIONS

This paper has presented a knowledge-based system that solves problems of operation and utilization that have currently been dependent on human intervention. Also, several examples of problem solving mechanisms basic to the domain of the operation and utilization have been described.

Several Composer's subdomains have actually been developed and released to customers, while subdomain such as database operation assistance is still at pre-prototype level, and we continue to actively develop these subdomains. In order to solve a variety of problems of operation and utilization, more basic problem-solving mechanisms are required. Currently, Composer has more than 3000 knowledge components, such as rules, and consists of over one hundred thousand lines of code.

The Composer's result can be evaluated from quantitative and qualitative viewpoints. Quantitatively, the time for solving problems previously done by hand, has been significantly reduced. For example, performance analysis tasks usually took about 2 days collecting and analyzing a large amount of data, but Composer provides the solution in a few minutes. In a large computer, a large number of programs run simultaneously together with many continuous events. It is difficult for a human to monitor all the information without omission. A computer's strong points are its ability to process a large amount of data instantly and without omitting anything, so we take advantage of that. As for quality, it may currently be difficult to exceed the quality of a good human expert, but there are other advantages due to inherent properties of the computer. For instance, sometimes the required knowledge for solving a problem is not owned by one expert, but may be spread over several experts in different fields (like performance analysis). In this case, much more powerful problem solving can be realized by integrating such knowledge into a single knowledge base. In addition, it can always handle the problems calmly and quickly compared with human experts. Moreover, as the number of computer systems increases, the number of skilled experts for operation and utilization will not be sufficient. Composer can solve this problem by providing a means of distributing the expertise.
After all, its strength is that is facilitates high-level operation and utilization.

As future work, efficient knowledge acquisition must be studied. As the knowledge for operation and utilization is spread over many experts, it is hard to acquire knowledge from them. Currently, customization of Composer by a user is done only through limited knowledge acquisition such as menu input of constraints. Low-level tools such as knowledge editors are available but high-level knowledge acquisition is required for the effficient customization. Composer has made only the first step toward an intelligent operating system, and more basic problem-solving mechanisms for operation and utilization need to be developed.

ACKNOWLEDGEMENTS

The development of Composer has been a team effort. The author acknowledges the contribution of many team members and people who provided the knowledge. Thanks are due also to reviewers who made many useful suggestions for improving this paper.

6. REFERENCES

[1] McDermott, J., R1: A RULE-BASED CONFIGURER OF COMPUTER SYSTEMS, ARTIFICIAL INTELLIGENCE, vol. 19, no. 1 (1982) pp. 39 - 88.

[2] Griesmer, J.H. et al., YES/MVS: A Continuous Real Time Expert System, Proceedings of AAAI-84, Austin, TX (1984) pp. 130-136.

[3] Milliken, K.R. et al., YES/MVS AND THE AUTOMATION-OF OPERATIONS FOR LARGE COMPUTER COMPLEXES, IBM systems journal,vol. 25, no. 2 (1989) 159-180.

[4] Mathonet, R., Cotthem, H.V. and Vanryckeghem, L., DANTES: AN EXPERT SYSTEM FOR REAL-TIME NETWORK TROUBLESHOOTING, proceedings oijcai-87, Milano, Italy (1987) pp. 527-530.

[5] Takamura, J., Tanaka, S. and Imai, F., AN EXPERT SYSTEM FOR COMPUTER SYSTEM PERFORMANCE ANALYSIS, proceedings of 35th national conference of information processing society of Japan, Sapporo, Japan (1987) in Japanese.

[6] Takamura, J., Hashimoto, Y. and Oguro, M., A GENERIC DIAGNOSIS METHOD FOR COMPUTER OPERATION AND UTILIZATION USING THREE-LEVEL INFERENCE MECHANISMS, Proceedings of 37th National Conference of Inforïation Processing Society of Japan, Kyoto, Japan (1988) in Japanese.

[7] Clancey, W.J., Classification PROBLEM SOLVING, Proceedings of AAAI-84, Austin, TX (1984) pp. 49-55.

[8] Kiriha, Y. et al., AN APPROACH TO LOGICAL FAILURE ANALYSIS FOR NETWORK DIAGNOSIS EXPERT SYSTEM, Proceedings of 37th National Conference of Information Processing Society of Japan, Kyoto, Japan (1988) in Japanese.

[9] Descotte, Y. et al, MAKING COMPROMISES AMONG ANTAGONIST CONSTRAINTS IN A PLANNER, Artificial Intelligence, Vol. 27, No. 2 (1985) pp. 183-217.

[10] Doyle, J., Truth MainteNANCE SYSTEMS FOR PROBLEM SOLVING, MIT AI Lab Technical Report AI-TR-419 (1978).

[11] Yamada, S. and Takamura, J., Consistency Maintenance against CONTINUOUS REAL-TIME DATA IN AN EXPERT SYSTEM FOR COMPUTER SYSTEM PERFORMANCE ANALYSIS, Proceedings of 35th National Conference of Information Processing Society of Japan, Sapporo, Japan (1987) in Japanese.

[12] Sussman, G.J. et al., CONSTRAINTS: A LANGUAGE FOR EXPRESSING ALMOST-HIERARCHICAL DESCRIPTIONS, Artificial Intelligence, Vol. 14, No. 1 (1980) pp. 1-39.

[13] Williams, B.C., Doing Time: PUTTING QUALITATIVE REASONING ON FIRMER GROUND, Proceedings OF AAAI-86, Philadelphia, PA (1986) pp. 105-112.

[14] Allen, J., MAINTAINING KNOWLEDGE ABOUT TEMPORAL INTERVALS, Communications of the ACM, Vol. 26, No. 11, (1983) pp. 832-843.

[15] Yamada, S., Tanaka, S. and Takamura, I., REAL-TIME ANALYSIS AND TEMPORAL REASONING MECHANISM IN AN EXPERT SYSTEM FOR

SYSTEM PERFORMANCE ANALYSIS, Proceedings of 37th National Conference of Information Processing Society of Japan, Kyoto, Japan (1988) in Japanese.

[16] Schank, R.C. et al., INSIDE COMPUTER UNDERSTANDING: FIVE PROGRAMS PLUS MINIATURES, LAWRENCE ERLBAUM ASSOCIATES (1981).

[17] Schank, R.C., DYNAMIC MEMORY: A THEORY OF REMINDING AND LEARNING IN COMPUTERS AND PEOPLE, Cambridge University Press (1982).

[18] Doyle, J., A TRUTH MAINTENANCE SYSTEM, Artificial Intelligence, Vol. 12, No.3 (1979) pp. 231-272.

[19] de Kleer, J., AN ASSUMPTION-BASED TMS, Artificial Intelligence, Vol. 28, No. 2 (1986) pp. 127-162.

[20] de KIeer, J. et al., BACK TO BACKTRACKING CONTROLLING THE ATMS, Proceedings of AAAI-86, Philadelphia, PA (1986) pp. 910-917.

[21] Baba, N., Osanai, M. and Takamura, J., An Expert SYSTEM FOR COMPUTER JOB DEVELOPMENT ASSISTANCE, Proceedings of 35th National Conference of Information Processing Society of Japan, Sapporo, Japan (1987) in Japanese.

COMBINING THREE CONCEPTUAL MODELS: ABSTRACT DATA TYPES, LOGIC PROGRAMMING, AND DATABASES (*)

Scott N. Woodfield, Jeannette M. Weston, David W. Embley
Brigham Young University
Provo, Utah

RIASSUNTO

Abstract data types (ADTs), programmazione logica e database sono tre dei modelli concettuali dominanti nelle scienze degli elaboratori. Sfortunatamente molti sviluppatori di software usano solamente uno di questi paradigmi. Usare un solo modello concettuale isola gli sviluppatori da ulteriori crescite intellettuali. Comprendere le differenze e le affinità dei tre modelli concettuali migliora l'abilità nella risoluzione di problemi in modo rapido ed efficiente. Altri lavori hanno già mostrato le differenze e le affinità della programmazione logica e dei sistemi di database. In questo articolo viene esteso il lavoro per mostrare come una forma ristretta di sistemi ADT è eugualmente equivalente. L'articolo descrive i vantaggi di comprensione dei tre modelli, descrive le differenze tra i tre sistemi e fornisce esempi di come la loro conoscenza è utile.

ABSTRACT

Abstract data types (ADTs), logic programming, and databases are three of the dominant conceptual models used in computer science. Unfortunately, most software developers use only one of the paradigms. Using only one conceptual model usually isolates developers from further intellectual growth and development. Understanding the differences and similarities of the three conceptual models improves one's ability to solve problems quickly and efficiently. Other work has already shown the differences and similarities of logic programming and database systems. In this paper we extend the work to show how a restricted form of ADT systems is also equivalent. The paper describes the advantages of understanding the three conceptual models, defines the ADT system, shows that it is equivalent to the other models, describes differences between the three systems, and gives examples of how knowledge of the three systems is useful.

(*) This paper has been presented at the Eigth annual IEEE Conference on Computers and Communication, Scottsdale, Arizona, May 22 - 24, 1989

1. INTRODUCTION

As the problems being addressed by computer science grow in complexity, a need for a more abstract level of problem description has become apparent. The attempt to define and describe these high-level, abstract systems in a formal manner is known as conceptual modeling.

Contributions to conceptual modeling have been made separately by computer scientists working in the areas of database design, logic programming, and software engineering. In database design, semantic data models have been developed [4]. In logic programming, knowledge representations using Horn clauses have been investigated [2]. In software engineering, abstract data type models have been developed [3].

Even though the models from each of these areas have evolved separately and with a different emphasis on the representation and manipulation of information, each of these systems addresses semantically similar

problems. Thus, one's ability to solve problems can be enhanced by understanding to what extent these systems are equivalent and how they differ.

Work has already been done showing the equivalence of some of the models. There are well-known proofs for the expressive equivalence of relational algebra and relational calculus [5]. There is also a proof for the expressive equivalence of relational algebra and Horn clause-based predicate assertions [1].

In this paper we describe the results of extending this work further. We first explain in Section 2 the advantages for understanding the various points of view. Since the ADT system we are using is our own, we define it in Section 3. We then show in Section 4 how logic programming, database systems, and ADT systems are equivalent and how they differ. In Section 5 we show, using examples, how this knowledge can be used to improve software development. We summarize our results in Section 6.

2. MOTIVATION

A vital aspect of education is understanding varied points of view. Unfortunately, in computer science many people seem to cling fanatically to one conceptual model. The database person rarely knows what is happening in software engineering. The software engineer, who is used to programming using ADTs, thinks Prolog in useless. The Prolog programmer thinks all problems can best be solved by using logic. Actually, all fanatics do themselves a disservice.

Understanding how the different conceptual models are similar can greatly simplify software development and maintenance. A software developer can create a design using a favorite conceptual model and still communicate with others looking at the problem from another point of view. For instance, a person versed in Prolog can think in Prolog but communicate with customers in an imperative programming style. Sometimes it is easier to think of the solution using one model but, for efficiency reasons, implement the solution using another model. Thus a Prolog programmer can solve a problem in Prolog, but optimize it by translating the solution to a database system. Another advantage of multiple conceptual models is the ability to choose the most expressive model to solve a particular problem. For instance, a developer who normally programs using ADTs may find it advantageous to create a prototype using Prolog. Without knowledge of these various models and their similarities the developer is unknowingly constrained by the weaknesses of a single conceptual model and its corresponding implementation.

Not only is understanding the similarities between the conceptual models important, but also understanding the differences. It is often possible to write statements in one system that require lengthy and difficult-to-understand explanations in other systems. For instance, assume a designer wanted to create a genealogy system such that a user could ask for all the known ancestors of a given relative. The solution of this problem using database systems is difficult since recursively related queries are not allowed. Logic-based systems do allow such queries and thus should be the initial system of choice.

In the future, knowledge of multiple conceptual models will be the norm. Use of multiple conceptual models will allow software developers to mix their paradigms within the same problem solution. We anticipate that many software solutions will contain aspects of expert systems and logic programming, databases, and abstract data types. Designers of new languages and software development methodologies will incorporate the advantages of the different points of view. Thus Prolog may be improved so as to be modular. Abstract data type systems may incorporate declarative statements that improve expressibility. Databases will be changed to allow arbitrarily defined operations and data types. We already have the different points of view. It is time we used them to our advantage.

3. ADT SYSTEM DEFINITION

To describe the similarities and differences of the three conceptual models first requires a definition of the three models. We use the formal models of logic programming systems and database systems presented in the literature [2, 5]. To extend the work, we will show the equivalence of a restricted ADT system and database systems. For the purposes of showing the equivalence of ADT systems with database systems we present a fairly detailed definition of an ADT system.

3.1. ADT Definition

An ADT is a triple (ADT name, domain definition, operations definition). An ADT name names the ADT and uniquely distinguishes it from other ADTs in the system. The domain definition defines the set of possible values for instances of the ADT. The operations definition defines a set of operations each of which creates, deletes, or manipulates these instances in some way.

Domain definitions define sets by enumeration, aggregation, power sets or unions. An enumerated domain lists the values of its domain within braces. An example of an enumerated domain might be a list of the four time zones in the continental United States:

{Eastern, Central, Mountain, Pacific}

An aggregate domain defines a set of (unordered) n-tuples by listing n previously-defined ADTs, each of which is optionally labeled with an identifier unique within the ADT domain definition. For example, an airline route from one city to another may be represented as follows.

Flight: Integer
FromA: Airport
ToA: Airport
Duration: Hours_Minutes

A power-set domain defines a set as the power set of a previously defined ADT and is denoted by a 2 superscripted with an ADT name. For example, the set of all possible sets of routes would be 2Route. A union domain defines a set as the union of domains of previously defined ADTs and is denoted as a sequence of "|" -separated ADT names. The following is an example of the domain definition for the "Gender" ADT.

male | female

A set defined in the domain definition of an ADT constitutes the intent (the set of all possible values) of the ADT. The set of values associated with all instances of an ADT constitutes the extent (the set of all existing values) of the ADT.

The operators in the operations definition are standard function definitions. Functions are value-returning procedures including constants. There are no non-value-returning procedures. Input/Output is handled separately, as a special request. The set of operations for every ADT usually includes an operation that creates an instance of the ADT, an operation that deletes an instance of the ADT, and an operation that compares two instances of the ADT for equality.

For clarity and ease of expression, we add some syntactic sugar to an ADT definition. We label each part of the ADT definition, format the ADT definition vertically, and place "End" at the end of the ADT definition. An example is shown in Figure 1.

```
ADT Name: Domestic_Time_Zone
Domain:
  {Eastern, Central, Mountain, Pacific}
Operations:
  Create (String) Domestic_Time_Zone
  Delete (Domestic_Time_Zone)
  Equal(Domestic_Time_Zone, Domestic_Time_Zone)
  → Boolean
Earlier_Time(Domestic_Time_Zone,Domestic_Time_Zone)
  → Boolean
Next Earlier Time(Domestic_Time_Zone) Domestic_ Time
  _Zone
Next_Later_Time(Domestic_Time_Zone) Domestic_Time
  _Zone
End
```

Fig. 1 Sample ADT Definition

We assume the existence of several predefined ADTs. These include Integer, Real, Boolean, String, and Set. For these ADTs, standard literals are used to invoke the Create operator.

3.2. An ADT Program

Our ADT system is functional, and thus an ADT program is a single operation invocation. Syntactically, an operation invocation names the operation and supplies parameters. To resolve ambiguities that may arise, the operation may be prefixed by the ADT name. For example, Figure 2 shows a simple ADT program based on the ADT in Figure 1.

```
Next_Earlier_Time(Next_Earlier_Time(
   Domestic_ Time_Zone.Create("Eastern")))
```

Fig. 2 Sample ADT Program.

Since a program may be quite complex and since it is sometimes necessary to use the same operation invoca-

tion in more than one place, a shorthand for writing an operation invocation is provided in the ADT system. Any operation invocation may be named by giving a name and assigning it the operation invocation. Thus, for example, we may name the ADT program in Figure 2 by writing

```
Two_Earlier_ZonesNext_Earlier_Time(Next_Earlier_Time(
  Domestic_ Time_Zone.Create("Eastern")))
```

Then, wherever the operation invocation appears, it can be replaced by its name. For instance, Next_Earlier_Time (Next_Earlier_Time(Domestic_Time_Zone.Create ("Eastern"))) can be replaced be Two_Earlier_Zones. The name of an operation invocation persists between invocations of ADT programs and between invocations of the ADT system. A program name can be explicitly removed by assigning it the empty program, denoted by the absence of a right-hand-side of the assignment.

For our ADT system we define some simple Input/Output mechanisms. Output is done with a Display command that has as its one argument an ADT program. For example, Display(Two_Earlier_Zones) displays the results of the ADT program in Figure 2. Input operations are similar to Create statements. They are declared exactly like Create statements except that the word "Read" replaces the word "Create". The Read statement can be used wherever a Create statement may exist. The only difference is Create statements contain the set of values (in parentheses) needed to create a new instance of the associated ADT. Read statements expect the needed values to come from a source external to the program.

4. SYSTEM EQUIVALENCE AND DIFFERENCE

Prolog, relational database management systems, and our system of ADTs have many similarities, but are not equivalent. To establish equivalence, system subsets are necessary. In the equivalence subset we see how the three views of information processing are the same. In the equivalence subset complement for each system, we see how the three views differ.

4.1. Equivalences

We restrict relational database management systems to include only relational algebra/calculus plus the update operators Insert, Delete, and Change [5]. As is well known, relational algebra, tuple relational calculus, and domain relational calculus all have equivalent expressive power. We augment this expressive power with the standard update operators.

To restrict our ADT system to have this same expressive power, we first explain how to implement relational algebra with update as a set of ADTs. This ensures that the ADT system is as expressive as the restricted RDBMS. To obtain equivalence, we then impose restrictions that forbid the use of features that would increase the expressive power beyond the power of the restricted RDBMS.

To implement relational algebra with update, we define the Relation ADT shown in Figure 3. The intent of the domain of the Relation ADT includes all possible relations, and the extent includes all currently existing relations in the relational database system (both temporary and persistent), as needed. The set of operations of the Relation ADT includes the Create operator which constructs single-tuple/single-attribute constant relations, the two Select operators which together provide for the two types of select with a single comparison, natural join, project, union, difference, and renaming and is thus relationally complete [5]. The set also includes the update operators and the Delete operator, which is used to remove temporary relations that are automatically and implicitly removed in RDBMS systems.

```
ADT Name: Relation
Domain:
 Relation1 | Relation2 | ... | Relationn
Operations:
 Create (Attribute, Element) → Relation
 Delete (Relation) →
 Select_Value (Attribute, Operator, Element, Relation) →
 Relation
 Select_Attribute(Attribute, Operator, Attribute, Relation)
 → Relation
 Project(Attribute_Set, Relation) → Relation
 Join(Relation, Relation) → Relation
 Rename(Attribute, Attribute, Relation) → Relation
 Union (Relation, Relation) → Relation
 Difference (Relation, Relation) → Relation
 Insert (Tuple, Relation) → Relation
 Delete (Tuple, Relation) → Relation
 Change (Attribute, Old_Element, New_Element, Relation)
 → Relation
End
```

Fig. 3 The Relation ADT.

Since the Relation ADT (in Figure 3) includes references to several (as yet) undefined ADTs, we must also describe them. Before doing so, however, we observe that we must be able to dynamically construct and alter ADTs during the execution of an ADT program. For example, when an ADT program executes the Relation.Create operation a new Relation, Relation $_{n+1}$, must be created and added to the union of ADTs in the domain of the Relation ADT. We thus define an ADT ADT that can dynamically create and alter existing ADTs.

The ADT ADT is shown in Figure 4. Its domain definition is an aggregation consisting of the three parts required for an ADT. Its operations allow us to create and delete ADTs, to extract any part of an ADT, and to modify a union ADT by adding an ADT to its domain definition. This last operation is especially important for the Relation ADT since it allows us to create constant and temporary relations for our relational database system.

```
ADTN Name: ADT
Domain:
 ADT_Name
 ADT_Domain_Definition
 ADT_Operations_Definition
Operations:
 Create(ADT_Identifier,ADT_Domain, ADT_Operations)→
 ADT
 Delete(ADT)
 Equal(ADT, ADT) →Boolean
 Get_Name(ADT) → ADT_Name
 Get_Domain(ADT)→ADT_Domain_Definition
 Get_Operations(ADT)→ADT_Operations_Definition
 Add_ADT_to_Union_Domain(ADT, ADT)  ADT
End
```

Fig. 4 The ADT ADT

We now explain sufficiently for this paper how each of the undefined ADTs mentioned in Figures 3 and 4 are defined. (Full definitions are contained in [6].)

Each individual relation ADT, Relation$_1$, is defined to have as its domain a power set of an aggregate ADT. Figure 5 shows a sample relation and its corresponding construction in our ADT system. In Figure 5, we use mnemonic names for relations and tuples. Thus, if Relation$_1$ in Figure 3 were the Airport_Timezone_Relation in Figure 5, Relation1, would be replaced by Airport_Timezone_Relation. The domain definition for the Airport_Timezone_Relation is a power set defined over the aggregate ADT Airport_Timezone_Tuple. The intent of the Airport_Timezone_Relation is the set of all possible sets of tuples formed as the cross product of Airport strings and domestic time zones. It thus allows for any possible Airport_Timezone relation. The particular extent created by the ADT program in Figure 5 matches the relation shown in Figure 5.

Relation in Tabular Form

Airport_Timezone (Airport Timezone)

Airport	Timezone
JFK	Eastern
ORD	Central
SLC	Mountain
LAX	Pacific

Relation in ADT Form

```
ADT Name: Airport_Timezone_Relation
Domain:
 2Airport_Timezone_Tuple
Operations:
 Initialize → Airport_Timezone_Relation
End

 ADT Name: Airport_Timezone_Tuple
Domain:
 Airport: String
 Timezone: Domestic_Time_Zone
Operations:
 Create(String,Domestic_Time_Zone)→Airport_Timezone_
 Tuple
End

Insert(Airport_Timezone_Tuple.Create("JFK", "Eastern",
Insert(Airport_Timezone_Tuple.Create("ORD","Central",
Insert(Airport_Timezone_Tuple.Create("SLC","Mountain",
Insert(Airport_Timezone_Tuple.Create("LAX","Pacific",
  Airport_Timezone_Relation.Initialize))))))))
```

Fig. 5 A Sample Relation Implemented as an ADT

Each of the remaining ADTs mentioned in Figures 3 and 4 can also be defined. For example, similar to the Relation ADT, which has as its domain a union of all relations of interest, we also have a Tuple ADT that has as its domain a union of all tuples of interest. Thus, we are able to pass an appropriate tuple and relation to the

Insert or Delete operator to do an update. Additional definitions and explanations for mentioned ADTs can be found in [6].
Because we can implement relational algebra with updates, we are able to establish that our ADT system is as powerful as relational algebra with updates. To ensure that it does not have too much power, we restrict our ADT system to just the implementation and no more. Specific details of this restriction as well as a proof that the restricted ADT system is equivalent to relational algebra with updates is contained in [6].

To make Prolog part of this equivalence class, we must also restrict it. Chandra and Harel have given a proof of the equivalence of queries on relational databases and logic programs consisting of restricted sets of Horn clauses [1]. To do so, they provided a way of adding the negation operator to logic programs such that logic programs were equivalent to first-order, fixpoint query systems. We adopt the same Horn clause system and negation operator and thus, because of their proof, obtain the equivalence we desire.

In our restricted Prolog system, a relation is created by stating basic facts using the same predicate name. Our Airport_Timezone relation in Figure 5, for example, is shown in Prolog in Figure 6. Prolog queries are written as Prolog programs, but are restricted to sets of Horn clauses whose referencing structure constitutes a rooted acyclic graph.

```
% Attribute names: Airport Timezone
airport_timezone(jfk, eastern).
airport_timezone(ord, central).
airport_timezone(slc, mountain).
airport_timezone(lax, pacific).
```

Fig. 6 The Airport_Timezone Relation in Prolog

4.2. Strengths, Weaknesses, and Differences

Determining the aspects of logic programming, database systems, and ADT systems that are equivalent also provides an understanding of the differences, both weaknesses and strengths, between the three conceptual models.

Recursive relations are a definite strength of logic programming. An attribute of a relation can be defined in terms of the relation itself. This is not possible in database systems and is not as easily represented in ADT systems. Another strength of logic programming is the ability to determine why a particular answer was given. Also, in logic-based systems it is easy to add a constraint to a relation.

While these attributes are useful, logic programming could be improved by including some of the capabilities found in the other paradigms. It would be useful if the rules could be organized and abstracted using some of the abstractions found in ADT systems. Also, it would be better if operations that cnanged the state of the system were made first class concepts in Prolog rather than considering them as distasteful "side effects". It would also be useful if, when desired, the elements and variables of a program could be typed such as in Turbo Prolog.

As with logic programming, database systems have some strengths and weaknesses. Because the relational database conceptual model is essentially the intersection of the three conceptual models, databases are usually efficient and the easiest model to use. When a problem can be solved using any of the paradigms, it is probably best solved using database technology. Also, the conceptualization and expression of ideas using tables is usually easier than using logic and ADT systems. The use of universal and existential type quantifiers often makes the statement of queries compact and easy to understand.

Database systems can be improved. It would be nice if recursive relations such as those found in Prolog could be expressed. It would also be beneficial if the attributes of relations could be defined by ADTs rather than as basic built-in types. This and the generalization/specialization capabilities found in ADT systems could simplify the structuring and organization of a relational data model. With the operations associated with ADTs it would also be nice if imperative control capabilities were included in the system.

Working with the three paradigms has shown us that ADT systems are powerful. To show that ADT systems were equivalent to database systems required many restrictions on ADT systems. The ability to create many operations besides create, delete, and change is helpful.

The operational abstractions make it much easier to understand a program. The other forms of abstraction, aggregation and generalization, are also useful. Often it is much better to think in terms of what can be done to something rather than how "a" is related to "b". The use of types to constrain the entire system is also helpful.

Though ADTs are nice there is still room for improvement. The ability to use implication could significantly reduce software costs in many situations. It would also be nice if generalized iterators similar to the universal and existential quantifiers were made available for this paradigm.

5. EXAMPLES

The following examples show the use and advantages of the different conceptual models. The examples use the information shown in the "Airport_Timezone" relation of Figure 5 and the "Flight-Information" relation of Figure 7.

Route (Flight	FromA	ToA	Duration)
59	ORD	BOS	2:24
78	SLC	BOI	0:46
290	SLC	ORD	6:20

Fig. 7 Flight_Information Relation

5.1. Problem Solutions Expressed in Different Systems

The problems that can be solved using the equivalent subsets of the three conceptual models are essentially information storage and retrieval problems. The following examples show how the same information retrieval question can be answered using the different models. The question is "Which flights end in SLC?" The following is an example of the query in relational algebra.

$$\text{slcflights} = \pi_{\text{Flight}} + \sigma_{\text{ToA}} = \text{SLC}^{\text{routes}}$$

In Prolog the query would appear as follows.

```
slcflights (Flight) :-
   routes (Flight, ToA, ToA, Duration),
   ToA = 'SLC'.
slcflights (Flight)?
```

In our ADT system the question would take the following form.
slcflights Project (Flight, Select_Value (ToA, Equal,"SLC", routes))

5.2. Mixed Use of Different Systems in the Same Problem

While the previous examples show that for simple information storage and retrieval the model of choice can be used, under other circumstances it is desirable to mix the paradigms. The following examples again show that the same query can be answered using either relational algebra or the ADT system. These two queries in Figure 8 answer the following question: "List flights beginning and ending in different time zones that do not begin or end in the Eastern time zone".

The following example shows that by mixing the two paradigms the query is much smaller and easier to understand. At least it is easier for those who understand both Prolog and the ADT system.

Mixed ADTs and Prolog:

```
X_Flights  Project (Flight,Difference (diffzones, Union (
   Select_Value (FromZone, Equal, "Eastern", diffzones),
   Select_Value (ToZone, Equal, "Eastern", diffzones))))

diffzones (Flight, FromZone, ToZone) :-
   routes (Flight,FromA,ToA,Duration),
airport_timezone (FromA,FromZone),
   airport_timezone (ToA, ToZone),
   not(FromZone = ToZone).
   diffzones (Flight, FromZone, ToZone)?
```

5.3. Use of Additional Features of Other Systems

The following examples show how useful it would be to use some of the features of one system in another system. In some sense these examples demonstrate, as does the previous section, that the combination of the three conceptual models is useful. Our claim is that it is possible to extend the languages such that the user thinks of the combination of models as one new model rather than as three separate models.

The first example shows how natural and useful it

would be to extend database systems so as to ask why a particular result was obtained from a query. The added feature is taken from logic systems but since relational algebra is equivalent, the extension to database systems should be simple. The natural language version of the query is: Which flights begin or end in SLC and why? The extended database form of the query might appear as follows.

slcflights = π_{Flight} (
($\delta_{fromA\ City}\sigma +_{fromA = SLC}$ Routes) $\cup$
($\delta_{ToA\ City} +_{ToA = SLC}$ Routes))
why (slcflights)?

It would be more difficult to answer this question using standard database languages.
The second example poses the question: Which routes form the shortest round trip? In this case shortest round trip means least amount of flying time. The following is one solution to this query using an extended form of the ADT system. It is assumed that the ADT system allows arbitrary operations to be defined and that declarative statements, such as those found in Prolog, can be used.

```
shortest  Union (
  Join(Routes, Rename (Flight1, Flight,
    Project (Flight1,  Select(Triptime, Equal,
    Select_Least_Value(Triptime, roundtrip), round
      trip))),
  Join(Routes, Rename (Flight2, Flight,
    Project (Flight2, Select (Triptime, Equal,
      Select_Least_Value(Triptime, roundtrip), round
      trip)))))

  roundtrip(Flight1, Flight2, Triptime) :-
    route(Flight1, FromA, ToA, Duration1),
    route(Flight2, ToA, FromA, Duration2),
    Triptime is Duration.Add(Duration1, Duration2).
  roundtrip(Flight1, Flight2, Triptime)?
```

For those not used to the combined paradigms the solution appears to be difficult to understand. Actually it is much shorter and simpler than any corresponding solution expressed in a single conceptual model. The first statement, "shortest", searches all round trips for the ones with the shortest flying times. It then reports the routes associated with these round-trip flights. The use of the two operations "Select_Least_Value" and "Duration.Add" make the statement shorter than it would be in the Prolog or database versions. These abstract operations are not available in the restricted equivalence class but are easily defined elsewhere in the extended ADT system. The declarative statement for "roundtrip" also significantly shortens the program. This one statement searches all pairs of flights and computes the duration for each round trip found.

6. CONCLUDING REMARKS

Understanding the similarities and differences between the three paradigms has produced some interesting insights. While all three models may initially appear to be different they are in fact quite similar. The equivalent subset of the three models provide basic information and storage capabilities. Prolog does allow recursive relations but to a large extent is similar in power to database systems. Unrestricted ADT systems do seem to have a few valuable capabilities not found in the other systems. ADT systems allow arbitrary types or concepts to be represented and provides nicer abstraction mechanisms. ADT systems are especially useful for operations that interact with or represent actions in the real world.

Knowledge of the similarities and differences makes it possible to produce better software. Problems can be solved using the most appropriate conceptual model. In the future we should be able to do even better by combining paradigms. Hopefully we will create Relational Algebra Only:

π_{Fight} ($\sigma_{FromZone}$<>ToZone (
(Routes ∞ $\delta_{FromA\ Airport}$ $\delta_{FromZone\ Timezone}$ Airport_Timezone)
∞
(Routes ∞ $\delta_{ToA\ Airport}$ $\delta_{ToZone\ Timezone}$ Airport_Timezone))
- ($\sigma_{FromZone = Eastern}$ (
(Routes ∞ $\delta_{FromA\ Airport}$ $\delta_{FromZone\ Timezone}$ Airport_Timezone)
∞
(Routes ∞ $\delta_{ToA\ Airport}$ $\delta_{ToZone\ Timezone}$ Airport_Timezone))
- ($\sigma_{ToZone = Eastern}$ (
∞
(Routes ∞ $\delta_{FromA\ Airport}$ $\delta_{FromZone\ Timezone}$ Airport_Timezone)
∞
(Routes ∞ $\delta_{ToA\ Airport}$ $\delta_{ToZone\ Timezone}$ Airport_Timezone))))

ADTs Only:

```
X_Flights Project ( Flight, Difference(
     Select_Attribute(FromZone, Not_Equal, ToZone, Join(
       Join(Routes, Rename(FromA, Airport,
        Rename(FromZone,Timezone,Airport_Timezone))),
       Join(Routes, Rename(ToA, Airport,
        Rename(ToZone, Timezone, Airport_Timezone))))),
     Union(Select_Attribute(FromZone, Equal, "Eastern",
       Join( Join(Routes, Rename(FromA, Airport,
        Rename(FromZone, Timezone, Airport_Timezone))),
       Join(Routes, Rename (ToA, Airport,
         Rename(ToZone, Timezone, Airport_Timezone))))),
Select_Attribute(ToZone, Equal, "Eastern", Join(
     Join(Routes,Rename(FromA, Airport,
       Rename(FromZone, Timezone, Airport_Timezone))),
      Join(Routes, Rename(ToA, Airport,
       Rename(ToZone, Timezone, Airport_Timezone))))))))
```

Fig. 8 Simple Relational Algebra and ADT Only Queries

development methodologies and implementation languages that take advantage of these insights. Such capabilities will help reduce many of the problems associated with software development and evolution.

7. REFERENCES

[1] A. Chandra and D. Harel, HORN CLAUSES AND THE FIXPOINT QUERY HIERARCHY, Proceedings of the ACM Symposium on Principles of Database Systems, pp. 158-163, Los Angeles, California, March 1982.

[2] W.F. Clocksin and C.S. Mellish, PROGRAMMING IN PROLOG, Springer-Verlag, New York, New York, 1984.

[3] J.A. Goguen, J.W. Thatcher, and E.C. Wagner, AN INITIAL ALGEBRA APPROACH TO THE SPECIFICATION, CORRECTNESS, AND IMPLEMENTATION OF ABSTRACT DATA TYPES, in Current Trends in Programming Mlethodology, ed. R.T. Yeh, pp. 80-149, Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1978.

[4] R. Hull and R. King, SEMANTIC DATABASE MODELING: SURVEY, APPLICATIONS, AND RESEARCH ISSUES, ACM Computing Surveys, vol. 19, no. 3, pp. 201-260, September 1987.

[5] D. Maier, THE THEORY OF RELATIONAL DATABASES, Computer Science Press, Rockville, Maryland, 1983.

[6] J.M. Weston, EQUIVALENCE OF AN ABSTRACT DATA TYPE VIEW OF INFORMATION PROCESSING AND A KNOWLEDGE AND DATABASE VIEW OF INFORMATION PROCESSING, Master's Thesis, Brigham Young University, 1988.

AVVENIMENTI

" NEURAL NETWORKS : COMMERCIAL PROSPECTS "

(Londra, 16 - 17 febbraio 1989)

Paolo Stofella
Advanced Computing Systems s.r.l. - Milano

Le attività nel settore delle reti neuronali e dei modelli connessionisti hanno subito, dalla pubblicazione dei lavori del Parallel Distributed Processing Group, un incremento crescente ed una rinata attenzione all'interno della comunità scientifica internazionale, attraverso un interessamento che coinvolge discipline tra loro differenti, quali le scienze computazionali, le scienze neurofisiologiche, la psicologia cognitiva. Le proprietà di taluni modelli sviluppati in questo ambito, in particolare quel concetto di comportamento adattivo che delle reti neuronali è la caratteristica peculiare, si sono rivelate estremamente promettenti ed hanno stimolato l'interessamento di numerosi ricercatori provenienti dalle suddette discipline.

Il settore presenta un grado di maturità tale da cominciare a coagulare interessi che esulano dall'ambito proprio della ricerca; numerosi esperimenti suggeriscono infatti possibilità d'impiego del paradigma connessionista in settori applicativi di importanza commerciale non trascurabile, quali pattern recognition, elaborazioni di segnali, robotica, visione artificiale, comprensione del parlato e del linguaggio naturale, oltre al settore delle memorie associative per il quale questi modelli potrebbero rappresentare una autentica rivoluzione tecnologica.

In questo clima di crescente attenzione si inserisce il secondo workshop europeo dal titolo "Neural Networks: Commercial Prospects", organizzato a Londra nei giorni 16 e 17 febbraio 1989 dalla società IBC Technical Services Ltd. Suddiviso in quattro sezioni, denominate "Marketing Segments", "Keynote", "Perspectives" e "Company Strategies", il convegno era volto a fornire una visione d'insieme sulle attività del settore a livello internazionale, attraverso la descrizione di alcune esperienze implementative e l'esposizione delle attività di grandi organizzazioni e società specializzate in Europa, Stati Uniti e Giappone.

Un'idea delle attività svolte a livello industriale in Europa è stata fornita attraverso gli interventi di Bernard Angeniol di Thomson-CSF, W. Buettner di Siemens, C. Nightingale di British Telecom. Queste organizzazioni hanno attivato unità di ricerca nel campo connessionista; Thomson-CSF mostra particolare interesse per applicazioni di signal processing e pattern recognition per segnali provenienti da radar e sonar, prevalentemente per il mercato militare, ed è forse la società che più ha investito in Europa nelle reti neuronali. Come Siemens, inoltre, ha avviato ricerche per l'implementazione di modelli connessionisti su chip. British Telecom concentra le proprie attività

su applicazioni di riconoscimento del parlato, elaborazione e compressione di immagini, pattern recognition.

Certamente più ricco il panorama degli investimenti statunitensi nel settore. L'intervento di G. Micholski di New Sciences Associates ha evidenziato come, oltre ai programmi finanziati dal governo americano, attraverso DARPA/MIT, National Science Fund, Office of Naval Research, ed ai programmi universitari, spiccano per dimensioni MIT, Stanford, Johns Hopkins, UCSD e Brown University, gli investimenti non mancano all'interno di grosse società private, quali DEC, IBM, General Dynamics, TRW, Hughes, Dupont, AT&T, Synaptics.

Altrettanto ricco il programma delle attività giapponesi, con il coinvolgimento di enti governativi, Ministry of International Trade & Industry (MITI) ed Electrotechnical Laboratory (ETL), ed organizzazioni industriali quali NEC, Fujitsu, Hitachi e Mitsubishi. Dalla relazione di Akio Kobotu di ETL si osserva una particolare attenzione, all'interno di questi progetti, per l'integrazione delle tecnologie connessioniste nell'ambito dei sistemi esperti e per la messa a punto di architetture neuronali multirete. Da segnalare un enorme progetto finanziato dal governo che partirà nel 1991, volto alla realizzazione di potenti neurocomputer, un po' frettolosamente battezzati computer della 6ª generazione.

Sul fronte applicativo l'esperienza di gran lunga più significativa, all'interno del convegno, è stata proposta da uno dei grandi padri del settore, il Prof. Teuvo Kohonen dell'università di Helsinki. "Phonetic Typewriter" è chiamato il prodotto realizzato da Kohonen e collaboratori, uno strumento per l'interpretazione del linguaggio parlato in grado di funzionare in tempo reale nel quale la fase di "phonemic recognition" è realizzata per mezzo di una rete neuronale del tipo "Self Organizing Feature Map" la cui definizione teorica si deve allo stesso Kohonen. Nel sistema sono impiegati algoritmi tradizionali (FFT, normalizzazione) per la fase di preelaborazione e viene utilizzata una Self Organizing Map per il riconoscimento dei fonemi. Una fase di "tuning" ulteriore sulla rete è realizzata con l'applicazione dell'algoritmo di Learning Vector Quantization, anch'esso teorizzato da Kohonen. Lo strumento, composto di hardware specializzato (utilizza un chip di tipo Digital Signal Processor) e di software di simulazione della rete neuronale, permette di ottenere un testo in tempo reale mediante dettatura.

Altre applicazioni presentate, sviluppate nel contesto di progetti di ricerca finalizzata, comprendono un sistema di pianificazione dei prezzi per compagnie aeree (un problema di ottimizzazione le cui caratteristiche in termini di complessità e numero di variabili lo rendono quasi intrattabile con tecniche tradizionali), un sistema per la valutazione dello stato di salute di una persona, legato all'esigenza di determinare i premi assicurativi di una polizza sulla vita, un sistema per la previsione dell'andamento di fattori finanziari.

Da segnalare l'intervento del Prof. I. Aleksander dell'Imperial College of Science Technology and Medicine di Londra, nel quale è descritta una macchina per pattern recognition, denominata WISARD, che utilizza tecniche di tipo connessionista. Secondo Aleksander, il tipo di modello utilizzato in WISARD, pur non discostandosi sensibilmente dai principi della computazione neuronale, potrebbe essere implementato utilizzando semplicemente memorie RAM, leggermente modificate. Malgrado l'impressione sia che questo modello presenti una effettiva applicabilità solo a problemi di pattern recognition, la prospettiva di poter utilizzare una tecnologia veloce e ben conosciuta come quella delle memorie è sicuramente degna di attenzione.

Gli interventi dei relatori di Hecht-Nielsen Neurocomputers, SAIC e Neural Ware, società produttrici di hardware e software per simulazione di reti neuronali, completano un convegno che ha messo in chiara evidenza quanto l'interesse per questo paradigma computazionale sia esteso, ma ha mostrato come questo interesse non abbia ancora prodotto dei risultati applicativi che superino lo stadio di prototipo nel contesto di ricerche finalizzate. Il volume degli sforzi a livello internazionale prospetta comunque una rapida evoluzione di queste tecnologie verso una maturità che permetta la produ-zione di strumenti complessi e affidabili per l'impiego nel mondo reale.

UMANISTI E INFORMATICA: UNA PANORAMICA DELLA CONFERENZA INTERNAZIONALE "THE DYNAMIC TEXT"

(Toronto, 5 - 10 giugno 1989)

Paolo Fezzi

La conferenza internazionale tenutasi a Toronto tra il 5 e il 10 giugno rappresenta un significativo momento di riflessione sullo stato attuale del matrimonio tra informatica e discipline umanistiche: si tratta infatti della prima conferenza in comune delle due principali associazione di informatica umanistica, la ACH (Association for Computers and Humanities), americana, con sede a Pittsburgh, e la ALLC (Assiciation for Literary and Linguistic Computing), europea, con sede a Oxford. Seguiranno a queste altre conferenze unificate, con scadenza annuale, alternativamente in America e in Europa: la prossima si terrà a SIEGEN, in Germania, dal 5 al 9 giugno 1990.

Alla conferenza, organizzata dall'Università di Toronto, erano presenti tutte le principali autorità delle due associazioni: Nancy IDE, del Vassar College, presidente dell'ACH; Ian LANCASHIRE e Willard Mc CARTY, dell'Università di Tronto, organizzatori della conferenza e compilatori del The Humanities Computing Yerabook; Susan HOCHEY, dell'Università di Oxford, fondatrice ed ex presidente dell'ALLC e per finire Antonio ZAMPOLLI, attuale presidente dell'ALLC. Sono intervenute alla conferenza alcune personalità di rilievo in campo sia umanistico che informatico, tra le quali Northrop FREYE, critico letterario canadese di fama mondiale, su posizioni di cauto scetticismo circa l'utilità del computer in campo umanistico; Jean-Claude GARDIN, studioso francese, autore di saggi molto interessanti di epistemologia della critica letteraria e assai favorevole all'impiego del computer per analisi di tipo strutturalistico; infine Ted NELSON, generalmente noto come padre degli ipertesti, che con un breve intervento, del tutto inatteso in quanto la sua presenza non era inizialmente prevista, ci ha dato modo di gettare uno sguardo al suo famoso quanto misterioso progetto XNADU, il cui obbiettivo è la costruzione di un ipertesto " universale ".

Parallelamente alla conferenza si è ottenuta una mostra di software specificamente destinato agli umanisti, significativa come segnale della presa di coscienza del potenziale mercato che studenti e professori di discipline umanistiche costituiscono per le case di software, ma povera di prodotti innovativi.

1. BILANCIO GENERALE

La conferenza, nel suo complesso, mi è sembrata rappresentativa più del livello medio che non delle tendenze più recenti dell'informatica umanistica: del resto i dirigenti delle due associazioni, non più giovanissimi, mi sono sembrati per la maggior parte più recettivi alle ultime novità, e legati a un uso riduttivo del computer,

legato alla tradizione della linguistica computazionale. In effetti, perlopiù il computer è ancora visto come strumento di calcolo per indagini statistiche sui testi (occorrenze, concordanze etc.): siamo ancora sostanzialmente fermi a Padre BUSA, le cui ricerche, allora pionieristiche, risalgono alla fine degli anni 40. Di conseguenza, benchè computers and Humanities siamo nel nord America un binomio riconosciuto e ufficializzato, non ancora istituzionalizzato , è vero, ma senz'altro più usuale e diffuso che da noi (per lo meno a livello scolastico), si tratta tuttavia ancora di Computers and Humanities e manca una reale integrazione tra le due discipline. Dagli umanisti il computer non è cioè visto come uno strumento in grado di avere un significativo impatto sulla propria metodologia e di condurre a una riorganizzazione delle proprie discipline.

E' quindi ancora poco diffusa l'idea del computer come strumento di organizzazione dei testi, a livello sia di lettura che di scrittura, in grado di svincolare il lettore e lo scrittore dalla struttura meramente sequenziale dei testi su carta e di consentire una lettura e una scrittura strutturate di volta in volta in base alle esigenze dello studioso. Sono cioè ancora largamente inesplorate, sul piano pratico delle realizzazioni effettive, le potenzialità del computer come strumento di manipolazione dei testi, per una lettura dei testi secondo più punti di vista e l'integrazione e comparazione di testi diversi, potenzialità che personalmente ritengo quanto mai promettenti.
In questa direzione spingono gli ipertesti e le tecniche di markup descrittivo, opportunamente cogniugate con interfacce multiwindow, ma a riguardo poco si è visto a Toronto.

Forse un impulso notevole in questa direzione lo potrà dare NEXT, il nuovo computer progettato da Steve JOBS, presentato sia alla conferenza che alla fiera come una macchina esplicitamente destinata a un pubblico di umanisti. NEXT unisce infatti un efficace e flessibile sistema multiwindow, particolarmente adatto alla realizzazione di ipertesti, a uno schermo con una luminosità sufficientemente bassa da render possibile una lettura prolungata, (sono infatti dell'opinione che fino a quando gli schermi non saranno progettati in funzione delle esigenze di lettura prolungata, strumenti come gli ipertesti, per quanto utili e interessanti possano essere, non possono esplodere per un uso di massa). Significativo è che l'aquirente riceva gratuitamente, insieme al computerNEXT, un disco ottico contenente l'opera ononima di Shakespeare.

In questa medesima direzione opera anche Ted NELSON, che vede il computer soprattutto come strumento di comunicazione universale, secondo un ottica che molti ritengono utopica: ma per progredire l'utopia è spesso indispensabile.

Sono inoltre rimasto negativamente colpito dalla totale assenza, in questa conferenza, di una riflessione metodologica sulle modalità di organizzazione delle informazioni e sulle modalità dell'interfaccia, problemi cruciali sia per il database tradizionali che per gli ipertesti. A quanto ho potuto vedere, sia alla conferenza che alla fiera del software, prevale un'approccio monumentale e fantastico, in cui l'enafsi è posta più sulle dimensioni dei database che non sulle modalità di organizzazione e di utilizzazione, vale a dire sulla loro reale utilità. L'interfaccia é generalmente trascurata, spesso macchiosa, di accesso lungo e laborioso, con il rischio di avere enormi archivi di dati, che pochi utilizzano, perchè troppo complicati.

2. DATABASE: IL DANTE'S PROJECT

I progetti di database di grandi dimensioni non solo denunciano un'evidente trascuratezza del problema dell'interfaccia, ma portano più in generale il peso di tecnologie ormai superate, le sole disponibili quando sono iniziati, ma difficili da abbandonare e da convertire in qualcos'altro, una volta che il database si è sviluppato raggiungendo una mole ragguardevole, tale da rendere antieconomica qualsiasi ipotesi di cambiamento. Emblematico a riguardo il Dante's Project, ambizioso progetto di un database destinato a conte nere, oltre alla Divina Commedia, circa 90 commentari (attualmente 25), sviluppato nei Dartmouth College, soprattutto ad opera di Robert HOLLANDER, un simpatico personaggio commovente nel suo sincero amore per Dante.

Attualmente si cerca con fatica di migliorarne la macchinosa interfaccia usando Hypercard, senza però sfruttarne le potenzialità di ipertesto, per le obbiettive difficoltà di riadattamento di cui sopra. Ed è un peccato, perchè un progetto del genere sarebbe molto adatto a

uno sviluppo ipertestuale, come conferma il progetto italiano Commedia Divina, più modesto di dimensioni, ma sicuramente molto più elastico nella struttura e più fantasioso e agile nell'interfaccia.

3. IPERTESTI

Benchè il titolo della conferenza alludesse agli ipertesti, poco si è visto a riguardo, e sicurapmente mi aspettavo di più.
Per quanto concerne il software, non ho visto nulla di nuovo rispetto a quello che già conosciamo in Italia. Per quanto concerne le applicazioni, ne ho viste alcune, tutte di Hypercard, che in America è molto diffuso. Il modello più in auge è quello dell'ipertesto enciclopedico, che sfrutta la capacità di integrazione di campi di studio generalmente separati per finalità divulgative: significativi esempi sono Culture, un prodotto americano che contiene una breve storia della cultura occidentale e permette rapidi passaggi dalla storia politica alla storia della letteratura, dalla storia dell'arte alla storia della musica di uno stesso periodo e Hyperbase francese, a cura del Centro Pompidou, un piccolo corpus di una ventina di scritti inerenti alla rivoluzione francese, di cui a luglio ricorreil bicentenario.

Il progetto scientificamente più serio e ambizioso che abbia visto nel campo degli ipertesti è sicuramente il Perseus Project, un archivio destinato, nelle intenzioni, a contenere l'intera letteratura greca antica, sviluppato dall'Università di Harvard in ambiente Hypercard, che tuttavia non mi sembra sfruttaretutte le potenzialità di un ipertesto limitandosi a fornire un'interfaccia per passare rapidamente da un documento all'altro.
In generale Hypercard è utilizzato più che come strumento ipertestuale per strutturare i testi, come interfaccia per realizzare semplici tavole sinottiche.

Sicuramente interessante è satto l'intervento di Ted Nelson, affascinante personalità di utopista pragmatico, l'unica vera strar della conferenza, per la sua capacità di instaurare un rapporto immediato e non accademico con il pubblico. Ci ha raccontato di come stia cercando di realizzare la sua idea di ipertesto come strumento di comunicazione universale, progettando XANADU come un ipertesto in grado di interfacciare tutti i sistemi operativi: il prototipo dovrebbe essere prontonel 1990, su stazione SUN e nel 1992 sarà dispensabile il prodotto accessibile al pubblico.

L'idea di Nelson è di fare di XANADU una storia di interfaccia universale tra i testi elettronici di tutto il mondo, predisponendo un archivio al quale chiunque, tramite XANADU, possa collegarsi, pagando una royalty per ogni link ipertestuale effettuata. In tal modo vorrebbe aggirata la legge copyright, e il computer diverrebbe realmente un formidabile mezzo di dialogo e integrazione tra testi, cioè, in ultima analisi, di diffusione della cultura.
In base a questo suo ideale Ted Nelson ha criticato i sistemi attualmente in commercio per la costruzione di ipertesti, in quanto non fanno che riprodurre lo stato attuale dell'editoria elettronica, caratterizzato da un'insostenibile incompatibilità tra i formati dei programmi di word-processing: gli ipertesti finirebbero così per costituire a loro volta dei mondi chiusi, tra loro incompatibili, e tradirebbero l'originaria e più autentica vocazione degli ipertesti, quella di strumento di comunicazione.

E' difficile fare previsioni sull'esito del progetto di Ted Nelson, che si dice convinto di riuscire a interessare e coinvolgere per ora IBM, APPLE e NEXT. Certamente è un tentativo che comunque non cadrà nel vuoto e produrrà degli effetti. Ed è ineccepibile il suo rilievo sulla contraddizione tra l'attuale incompatibilità dei testi elettronici e l'esigenza di una comunicazione la più ampia possibile.

Come ho già accennato, mi ha sorpreso l'assoluta mancanza di una riflessione metodologica sugli ipertesti, in particolare sui problemi concernenti la loro organizzazione, problemi particolarmente rilevanti nel caso di ipertesti di grandi dimensioni. Probabilmente questo è un indizio della scarsa diffusione degli ipertestinegli ambienti umanistici, che non si sono evidentemente ancora scontrati con questo tipo di problemi.
Naturalmente sono in corso interessanti realizzazioni nel campo degli ipertesti , non direttamente documentate dalla conferenza: mi é in particolare giunta notizia, parlando con i presenti, della realizzazione all'University of Columbia di New York di un grande archivio storico ipertestuale.

Anche il Max Plank Institute di Gottinga, in Germania,

sta facendo qualcosa nel campo delle fonti della storia tedesca.
Di entrambi questi progetti ho richiesto la documentazione.
Posso comunque affermare, da quello che ho visto, che gli ipertesti realizzati dal Dipartimento di Scienze dell'Informazione a Milano avrebbero fatto la loro figura a Toronto, segnalandosi soprattutto per una maggiore cura e fantasia nella progettazione dell'interfaccia (penso in particolare ad Ipermilano, a Iperfuturismo e alla Commedia Divina).

4. TEXT - PROCESSING: IL MARKUP

Si è molto parlato, a Toronto, delle tecniche di markup, un aspetto fondamentale del text-processing. Nel markup rientrano tutti quei contrassegni convenzionali che non fanno parete del contenuto del testo, ma servono a suddividerlo in porzioni riconoscibili e classificabili, per facilitarne la lettura , sia dall' uomo che dalla macchina.
Il markup si può considerare in realtà, a rigore di termini, come qualcosa di antecedente ai testi elettronici e comune a qualsiasi tipo di testo: per esempio, la punteggiatura e la spaziatura sono un tipo di contrassegno, cioè di markup, con la funzione di dividere tra loro le frasi e le parole.

I programmi di word-processing hanno aggiunto a questo tipo di markup tipografico, un loro specifico markup, che potremo definire procedurale, per indicare al programma in questione quale procedure di formattazione attuale in quel punto del testo (per esempio, il Wordstar il contrassegno .pa è letto dal programma come equivalente al comando " pagebreak "), per realizzare automaticamente determinati markup tipografici (per esempio, l'impaginazione, la suddivisione in paragrafi ecc.).

Negli ultimi anni si è avvertita da più parti l'esigenza di un markup descrittivo, vale a dire di una serie di contrassegni indipendenti dal programma di word-processing usato e con una funzione di classificazione: in grado cioè di indicare a quale classe appartiene la porzione di testoche segue. Il markup descrittivo avrebbe i seguenti vantaggi:

- grazie alla sua indipendenza dal software usato, costituirebbe un'interessante soluzione al problema dell'incompatibilità di formato tra i documenti elettronici, e consentirebbe un più facile scambio di testi tra gli studiosi;

- grazie alla sua maggiore generalità può essere utilizzato anche per fini diversi alla semplice formattazione, cioè per la classificazione concettuale del contenuto del testo, secondo le categorie scelte di volta in volta dallo studioso (per esempio per una classificazione grammaticale o stilistica delle varie parti del discorso, o per un'indicazione sintetica del contenuto di paragrafo o di una frase). In tal modo il markup descrittivo può diventare la base per tecniche di data-retrieval a full-text, suddividendo il testo in strutture analoghe ai record e ai campi di un data-base: con il vantaggio, rispetto ai database, di strutturare il testo in modo più flessibile di quanto non facciano i record e i campi, ridige strutture predefinite, e di aggiungersi al testo secondo un processo aperto e teoricamente infinito, anzichè richiedere laboriosi trasferimenti di testo in record e campi, difficili da cambiare, una volta definiti.

In tal modo il markup presegue fini analoghi a quelli degli ipertesti: svincola cioè la lettura e la scrittura dal consueto ordinamento sequenziale e consente da un lato una lettura strutturata e selettiva, secondo diversi punti di vista, di volta in volta scelti dallo studioso, dall'altro una scrittura organizzata secondo una gerarchia di livelli di astrazione successivi (si tratta del cosidetto structure-oriented editing).

Naturalmente l'efficacia del markup descrittivo aumenta se la comunità degli studiosi si accorda su uno standard, con il quale comunicare. La novità a riguardo, di cui si è discusso alla conferenza di Toronto, è la decisione comune delle due associazioni di informatica umanistica, l'ACH e l'ALLC, di adottare come standard di markup descrittivo l'SGML (Standard Generalized Murkup Language). Questo standard ha buone possibilità di imporsi, abnche perchè appoggiato dal Dipartimento della Difesa statunitense e dalla CEE, oltre che da case di software come la McGraw-Hill.

L'ACH e l'ALLC hanno inoltre dato vita, nel novembre del 1987, al TEI (Text Encoding Iniziative), il cui compito è di stabilire delle direttive comuni in materia di markup. La TEI lavora divisa in quattro commis-

sioni: la prima si occupa di Text Documentation, cioè di stabilire un markup in grado di classificare le caratteristiche fondamentali dei documenti d'archivio; la seconda si occupa di Text Representation, cioè di elaborare un markup per la classificazione delle caratteristiche tipografiche dei testi; la terza si dedica alla Text Analysis and Interpretation, cioè alla determina-zione di un markup uniforme per rappresentare le fondamentali categorie interpretative delle discipline umanistiche; la quarta commissione si occupa infine di Syntax and Metalanguage, controllando l'adeguatezza della sintassi di SGML rispetto a tutte le prevedibili applicazioni, definendo inoltre il metalinguaggio in grado di descrivere e di tradurre i principali linguaggi di markup esistenti.

Se la sintassi di SGML, relativamente semplice, è stata già stabilita, notevoli discussioni sono in corso sulla semantica, soprattutto su quali caratteristiche classificare e sul livello di generalità da adottare nel descriverle.

Alcuni studiosi propendono per uno sviluppo di un linguaggio di markup in grado di descrivere esaustivamentee minuziosamente le più diverse caratteristiche di un testo; altri concepiscono lo standard come una sorta di minimo comun denominatore, che lascia a ciascuno studioso la possibilità di concepire un markup ad hoc, rispetto alle sue specifiche esigenze; altri ancora sono ostili all'adozione di uno standard, vedendo in esso un'ostacolo alla propria inventivadi interpreti.
Si è sottolineato il fatto che il successo di SGML è indipendente dalla sua accettazione come standard da parte delle case di software (tra il software attualmente disponibile segnalo Softquad, su Macintosh), e che la sua accettazione sarà piuttosto una inevitabile conseguenza della sua effettiva utilità e della sua adozione da parte della comunità degli studiosi.

In effetti un linguaggio di markup descrittivo è utile anche in assenzadi un software specificamente dedicato: grazie alle consuete funzioni di sostituzione, comuni a qualsiasi editor, SGML può essere sfruttato come linguaggio di trasferimento da un formato all'altro.

Ritengo tuttavia che il markup standard, per essere maggiormente efficace, richieda assolutamente lo sviluppo di un software specifico, non ancora esistente, per:

- una maggiore leggibilità dell'output. Infatti, poichè si può arrivare, nel corso del makup, a un testo con più contrassegni che parole, si può avere come risultato un documento che è si macchine-readable ma è purtroppo human-unreadlabe ... Benché alcuni studiosi siano dell'opinione che le esigenze di text-processing richiedano un prezzo da pagare, in termini di una minore leggibilità dell'output, ritengo al contrario che la leggibilità dell'output, ritengo al contrario la chiarezza dell'output sia un'esigenza irriducibile, se si vuole realmente usare il computer come strumento di lettura di testi. E' dunque necessario un software che nasconda le marcature sullo schermo, consentendone l'eventuale visualizzazione a comando, come già fanno comunemente molti word-processor nei confronti dei propri comandi di formattazione;

- il potenziamento della lettura parallela delle porzioni di testo con stessa marcatura. Particolarmente utili risulterebbero sistemi multiwindow, in grado di consentirela visualizzazione simultanea di porzioni di testo concettualmente connesse. Sistemi del genere sarebbero qualcosa di simile a ipertesti dinamici, i cui link verebbero effettuati di volta in volta, a seconda della categoria scelta.

L'esigenza di un software specializzato per le esigenze specifiche degli studiosi umanistici è del retso avvertita da più parti: software orientato in questo senso già esiste, ma o è troppo complicato da usare (è il caso di TUSTEP, un programma tedesco, sviluppato da Whilelm OTT dell'Università di Tubinga, molto elastico e specificamente finalizzato al text-processing di tipo umanistico, ma richiedente una competenza pari a quella di un programmatore), o è ancora troppo generico e rigidamente legato a un suo propri specifico formato (è il caso dei sistemi comunemente in commercio, come Nota Bene, Word Perfect, Word Cruncher e simili).

5. CONTENT ANALYSIS

La content analysis è un analisi di tipo quantitativo, statistico, compiuta generalmente su un intero corpus di testi (di un autore o di un'epoca o di un genere letterario). E' un approccio ormai consolidato e, si può dire, tradizionale, che risale agli anni 50, e che è tuttora il più seguito nelle ricerche umanistiche con il computer, usato qui semplicemente come strumento di calcolo.

La prima fase consiste nell'enumerazione delle occorrenze di ciascuna forma linguistica. La seconda consiste in una classificazione (tagging) delle forme linguistiche, mediante la loro riconduzione a categorie stabilite dallo studioso,[1]: in questa fase rientra la lemmatizzazione, cioè la riconduzione delle varie voci ad alcune forme convenzionalmente riconosciute come fondamentali, dette lemmi, assunte come rappresentative di classi semantiche i cui membri hanno tutti lo stesso significato (per esempio, l'infinito di un verbo, come " andare ", è convenzionalmente riconosciuto come lemma, rappresentativo di tutte le altri voci del verbo. Se la lemmatizzazione è l'operazione più frequente, sono possibili altre forme ulteriori di classificazione (tagging), come il raggruppamento dei lemmi per sinonimi e per generi (per esempio " pantaloni " e " camicia " nel genere " vestiti "), secondo concettualizzazioni via via sempre più complesse.

L'intervento di un insieme coerente di classi semantiche (lemmi o categorie concettualmente più astratte) costituisce un thesaurus. I membri di un thesaurus possono essere oggetto di diversi calcoli statistici, spesso assai sofisticati.

Al momento di occorrenze di una certa classe semantica si può associare la lista dei riferimenti testuali (libro, capitolo, paragrafo, riga, ecc.), e si ha allora un indice, o la lista dei contesti in cui occorrono, cioè delle altre parole a cui si trovano associate nel testo, e si ha allora un elenco di concordanze.

La contenet analysis non mi sembra suscettibile di ulteriori sviluppi (questi rilievi statistici più di tanto non possono dire) e, come ho già detto, mi pare legata a un uso riduttivo del computers, come semplice strumento di calcolo.

Più che novità nei risultati, c'è quindi da registrare una grande quantità di programmi in grado di supportare questo tipo di analisi; programmi che, originariamente disponibili solo su maiframe, sono stati portati anche su personal computer (come OCP, realizzato ad Oxford, e TUSTEP, in Germania) o sono ora realizzati direttamente su personal, ad opera di università (DISCAN, TACT e altri) e di case di software, come estensione delle funzionalità di word-processing (Nota Bene, Word Cruncher e altri).

6. DISCOURSE ANALYSIS

Se la content analysis attua un inventario dei segni presenti in un corpus, la Discours Analysis si occupa delle relazioni tra i segni: se la contenent analysis opera sul piano del paradigma, la Discourse Analysis opera sul piano del sintagma, cioè della sequenza dei segni.

La Discourse Analysis, a metà strada tra statistica e strutturalismo, tra approccio e empirico e approccio e sistemico, è per ora più interessante per le potenzialità che lascia intuire, che per i resultati effettivamente ottenuti.

L' attenzione alle dinamiche del discorso sposta l'analisi dal testo in se stesso alla relazione tra testo e lettore (la cosidetta reader response), mettendo a fuoco il problema dell'interpretazione. In questa direzione si sta timidamente facendo strada (Jean-Claude GARDIN, Elaine Nardicchio) l'idea, suscettibile di sviluppi molto interessanti, del computer come strumento di esplicitazioni delle categorie interpretative dei lettori, allo scopo di rendere commensurabili tra loro le diverse interpretazioni, e di facilitare quindi il dialogo tra i critici.

Il limite consiste nella riduzione dell'interpretazione a un ristretto numero di categorie, troppo schematiche e semplicistiche per essere realmente rappresentative e nel ridurre la commisurazione nelle interpretazioni a un semplice confronto statistico tra le categorie impiegate dai lettori per classificare un personaggio o una situazione. Tale approccio rischia di produrre risultati banali (come la conclusione di Nardocchio, secondo la quale l'accordo tra le interpretazioni risulta maggiore tra i lettori acculturati, esperti della materia che leggono e che hanno sottoposto il testo ad almeno due letture) e di essere riduttivo rispetto alla creatività dell'interprete e

[1] Il tagging è attuabile o mediante il markup (vedi più avanti) o mediante la raccolta in un diverso file delle categorie associate alle forme linguistiche (cifr. DISCAN).

della complessità dell'interpretazione.

Interessante risulta l'idea dell'antropologo Pierre MARANDA, autore di un programma, DISCAN, di facile uso, che tra le altre cose consente un'analisi markoviana (probabilistica) delle relazioni tra porzioni di testo preventivamente marcate. In tal modo è possibile analizzare il grado associatività di categorie grammaticali, temi narrativi, concetti, immagini, misurando il grado di inerzia semiotica, cioè di stereotipia e, inversamente, di creatività , di un testo o di un intero corpus di testi.
Maranda collega inoltre l'approccio probabilistico alla teoria dei grafi, mirando a un discorso sgtrutturalistico empiricamente fondato su dati statistici.

7. RICERCHE LINGUISTICHE

La dirigenza delle due associazioni di informatica umanistica (primo tra tutti Zampolli) ha una buona partepreso le mosse da ricerche nel campo della linguistica computazionale, che dunque gode ancora di un prestigio notevole.

A giudicare dalla relazione di nicoletta CALZONARI, ricercatrice italiana che sta lavorando a Pisa insieme ad Antonio ZAMPOLLI, nella creazionedi database lessicali l'enfasi si sta gratuitamente spostando dalle analisi quantitativesul numero di occorrenze alla trasformazione di conoscenza, caratterizzati:

- dalla formalizzazione delle relazioni semantiche tra le diverse espressioni linguistiche, mediante reti semantiche;

- dalla riusabilità di queste basi di conoscenza per diversi scopi, grazie a una pluralità rispetto alle teorie linguistiche.

L'intenzione è di usare questi database come basi di conoscenza per eventuali sistemi esperti, capaci di svolgere interrogazioni su di un corpus testuale non più per semplici parole-chiave, ma per "concetti e relazioni semantiche". E' stata a riguardo auspicata dallo stesso Zampolli una più stretta collaborazione tra umanisti e linguistica computazionale e si è sottolineata l'esigenza di trarre tutte le informazioni necessarie per la strutturazione di tali sistemi esperti direttamente dai corpora (senza mediazioni?), al fine di garantire la fondatezza empirica dei sistemi in questione.

Mi è stato difficile distinguere, in questa come in analoghe relazioni , i desiderata dalle effettive realizzazioni e rendermi conto del grado di effettiva competenza di questi ricercatori nel campo dei sistemi esperti. Per quanto concerne le modalità di formalizzazione, ci si concentra soprattutto su relazioni tassonomiche, di similarità e di opposizione, con qualche accenno all'uso di una grammatica dei casi, per collegare la sintassi alla semantica.
E' legittimo aspettarsi, nello sviluppo di questi lavori, problemi di non facile soluzione.

8. INTELLIGENZA ARTIFICIALE

Da un serrato confronto con i sistemi esperti ci si aspetta, da alcuni studiosi soprattutto un chiarimento sullo statuto epistemologico delle scienze umane: daquesto punto di vista le conclusioni negative circa alcuni tentativi di tradurre in sistemi esperti la conoscenza di storici e critici letterari possono essere metodologicamente interessati.

Molto documentato a riguardo uno studio di Nancy IDE sull'applicazioni di tecniche di AI alla comprensione di testi letterari: passati in rassegna i vari metodi di rappresentazione della competenza linguistica, dalle reti semantiche ai frames, dagli script alla dipendenza concettuale al metodo dei plans and goals, si sono puntualmente precisati i limiti di queste metodologie e se ne è concluso che esse sono insufficenti, sia da sole che integrate, a emulare la competenza di un comune lettore umano di testi letterari.

Jean-Claude GARDIN si è occupato di questi problemi su un piano epistemologico, e vede nella plurualità di possibili interpretazioni di un testo, spesso tutte eugualmente legittime, il più grosso problema si pone all'applicazione di tecniche di AI alla comprensione dei testi, tanto più che questo pluralismo interpretativo, ben lontano dall'essere un limite da superare, come sostenevano i positivisti, è una ricchezza del pensiero umano e in particolare un patrimonio delle discipline umanistiche. Inoltre risulta arduo ricondurre a regole esplicite ez univoche molti dei criteri interpretativi , inconsapevoli e di natura contestuale.

Da altri studiosi è stata sottolineata la difficoltà che pone alle tecniche di AI la comprensione della metafora.

9. TRADUZIONE

Svanite le velleità ditraduzione integrale automatica, i tools per la traduzione visti in fiera propongono un approcciotipicamente umanistico, mettendo l'accento sull'interattività della traduzione, e presentando la traduzione automatica come stimolo didattico, che consente di imparare dagli errori della macchina: emblematico a riguardo il programma ALPNET.

10. SCANNERS/ OPTOPUS

Tra gli scanners visti in fiera mi è sembrato interessante un prodotto tedesco, Optopus, in grado di operare, oltre che automaticamente anche interattivamente, con la possibilità di imparare a riconoscere, mediante un apposito training, caratteri non previamente definiti.

11. CD-ROM E IPERMEDIA

Nessuna novità rispetto a quello che possiamo vedere in Italia.

12. STORIA E COMPUTERS

Un discorso a parte meritano le applicazioni del computer allo studio della storia.
Il Max Plank Institute di Gottiga ha in corso già da alcuni anni un monumentale progetto di un archivio di fonti di storia tedesca, che si dovrebbe avvalere anche di tecniche ipertestuali: ne sapremo quando ci giungerà la documentazione che ho richiesto.

All'University of Glasgow, in Scozia, stanno sviluppando un tool integrato per la ricerca storica, di cui ho una documentazione abbastanza vaga , che non era presente in fiera , perche non ancora finito.

All'University of Columbia di New York stanno lavorando a un archivio storico ipertestuale: ho chiesto di inviarmi la documentazione.

Ho potuto vedere un esempio di cosidetta simulazione storica: metodologicamente ingenua, criticabile da molti punti di vista (oltretutto non si tratta affatto di simulazione, in senzo stretto), ma interessante l'idea di dare allo studente l'impressione di entrare nella storia e di prendervi parte.

A un seminario ho potuto sentire l'interessante opinione di insegnanti di storia non specialisti, i quali vedono il computer come uno strumento per rendere più attivo l'apprendimento della storia, avvicinando lo studente a un uso diretto delle fonti. Grazie alla sua capacità di manipolazione di testi, l'elaboratore elettronico consente la visione in parallelo di fonti, generalmente considerate isolatamente, ma soprattutto il lavoro di immissione di documenti in formato elettronico permette allo studente una maggiore consapevolezza metodologica delle problematiche di critica delle fonti.
Infine l'Association for History and Computing ha iniziato quest'anno a Oxford la pubblicazione di un periodico dal titolo History and Computing, che si ripromette di documentare tutti i principali progetti in corso in Europa e nel Nord America.

13. " COMPUTERS AND HUMANITIES " E' UNA DISCIPLINA?

In un seminario ho potuto assistere a un animato e interessante dibattito sullo statuto epistemologico e accademico dell'informatica umanistica. Poiché nel Nord America le applicazioni dell'informatica alle scienze umane sono più diffuse che tra noi , si avverte ormai l'esigenza di riconoscere a "Computer and Humanities " un ruolo ufficiale in ambito universitario e si pone la questione " Computers and Humanities " è una disciplina autonoma? e se non lo è, come si può potenziarne lo sviluppo all'interno delle discipline curriculari?

Chi ritiene che l'informatica umanistica non costituisca una disciplina argomenta chein primo luogo essa costituisce disciplina meramente ausiliaria, di supporto alle ricerche umanistiche; in secondo luogo ha uno statuto ambiguo, essendo la combinazione di discipline etereogenee nei metodi e nelle tradizioni .

E' stato obbiettato, con ragione, che anche la fisiologia è una disciplina ausiliaria , che combina in sè metodologie e discipline diverse, senza per questo che le venga negato lo statuto di disciplina. Rimane comunque il

fatto che la materia in cui spazia "Computer and Humanities" è troppo ampia e generica per costituire il campo di un'unica discipliuna e ha probabilmente bisogno di essere ulteriormente specificata.

Rimanendo nell'ambito delle discipline curriculari nelle università nordamericane è possibile assemblare qualcosa di simile a una laurea in Computers and Humanities, configurandola come una specializzazione (majoring) in Computer Science nell'ambito di un corso di laurea umanistico o viceversa. E' stato segnalato però il pericolo che ad occuparsi di Computer and Humanities siano letterati e informatici di seconda cartegoria, che non riuscendo ad imporsi nella loro disciplina, cerchino di rilanciarsi combinando le due.

In ogni caso sono quasi tutti d'accordo nel vedere Computer Science e Humanities come discipline sostanzialmente separate e giustapposte, anzichè come discipline suscettibili di un integrazione, fruttuosa per entrambi.

14. WORKSTATION PER UMANISTI

In un seminario introduttivo, precedente di un giorno l'inizio della conferenza Norman MEYROWITZ (Brown University) e Ronald WEISMAN (University of Maryland) hanno cercato di definire le caratteristiche di una workstation in grado di soddisfare le esigenze degli studiosi umanisti, esponendo concetti non nuovi dfa noi, ma che è stato comunque interessante sentir ribadire in questa sede, e che sono indispensabili per superare quell'uso riduttivo del computer, che è ancora il più diffuso tra gli umanisti.

E' stato chiaramente evidenziato che l'umanista, più che alla definizione di algoritmi e regole, tende, nelle sue ricerche, alla definizione di relazioni tra oggetti: in altre parole lo stile ed il pensiero umanista è object oriented. Inoltre il ricercatore ha in questo campo bisogno di uno strumento in grado di supportare una veloce ed efficiente comparazione tra grandi quantità di testi.

Per soddisfare nella maniera migliore queste esigenze una workstation ideale dovrebbe avere queste caratteristiche:
- una memoria centrale possibilmente di 16 Mb (e comunque non inferiore ai 2 Mb);
- un video ad alta risoluzione;
- un mouse;
- una memoria esterna intorno ai 250 Mb, il che implica un'interfaccia con CD-ROM;
- connessione in rete;
- integrazione con mezzi audio-visivi, in grado di trasformare la workstation in un ipermedia.

Il software dovrebbe essere:
- object-oriented;
- multitasking;
- multiwindow;
- ipertestuale.

Qualcuno potrebbe osservare che queste sono esigenze comuni a tutti gli utenti più avanzati, e non specifiche degli umanistici. Personalmente ritengo che la convergenza tra le esigenze degli studiosi umanisti e quelle degli utenti comuni dimostri che l'attuale trend dell'informatica è orientato verso un approccio più umanistico verso il computer.

NOTE DI SOFTWARE 44/45

"SISTEMI DI SUPPORTO ALLE DECISIONI: METODOLOGIE E STRUMENTI"

(Milano, 25-26 maggio1989)

Giulio Falco

Artificial Intelligence Software S.p.A. - Milano

In una società caratterizzata dalla crescente complessità dei fenomeni e delle interazioni, dove il cambiamento è la norma, la disponbilità di informazione in grande quantità non sempre facilita il processo decisionale. I sistemi informativi esistenti sono basati generalmente su processi transazionali, problemi strutturati e modelli statici. E' uno scenario ben diverso da quello in cui si trova ad operare il decisore nella realtà quotidiana.

In questa ottica, non poteva quindi non destare interesse il seminario dedicato ai Sistemi di Supporto alla Decisione (DSS - Decision Support Systems) promosso dalla FAST e coordinato da Stefania Bandini (Dipartimento di Scienze dell'Informazione - Università di Milano) e da Davide Pandini (Inferentia s.r.l. - Milano).

Lo scopo del seminario era quello di fornire un quadro circa gli strumenti e le tecniche in grado di assistere e migliorare la qualità del lavoro di chi quotidianamente si trova nella necessità di operare delle scelte, in uno scenario complesso, caratterizzato da incertezza e da un elevato costo degli errori. Non a caso la maggior parte dei partecipanti, proveniva da settori aziendali legati al management.
Per quanto riguarda gli interventi si è voluto dare un taglio decisamente multidisciplinare e di interattività con i partecipanti.

La relazione introduttiva é stata tenuta da Stefania Bandini, che ha tracciato una panoramica storica, mettendo in luce la stretta connessione che la scienza delle decisioni ha sin dai suoi esordi avuto con le tematiche legate all'Intelligenza Artificiale e ai Sistemi Esperti.
Dalla fine della seconda guerra mondiale la realizzazione di strumenti decisionali, vede coinvolte principalmente Ricerca Operativa e Computer Science, per l'applicazione di modelli matematici (teoria dell'ottimizzazione) per gestire risorse scarse al fine di soddisfare requisiti complessi.
Queste teorie che si adattano perfettamente a problemi ben formalizzati, erano però poco esplicative rispetto al processo di modellazione di problemi reali, e oscure rispetto alla validazione e all'uso pratico di risultati in termini di allocazione concreta di risorse.
La naturale evoluzione ha condotto quindi verso strategie "aperte", ossia in grado di fornire non solo risultati,

ma anche una interpretazione dei risultati, al passaggio dalla computazione numerica a quella simbolica e all'impiego di strumenti in grado di "spiegare se stessi", come i Sistemi Esperti.
Una visione questa in accordo con quanto è emerso dalle frequenti interazioni con i partecipanti le cui esperienze mostrano chiaramente che:
- il manager non é disposto ad applicare risultati, e a utilizzare strumenti che non comprende perfettamente.
- non é disposto ad essere sostituito nel processo decisionale, ma desidera strumenti che lo assistano in situazioni complesse, dinamiche e incerte.

Queste tematiche sono state approfondite nell'intervento di Fabio Schoen (Dipartimento di Scienze dell'Informazione - Università di Milano) il quale ha trattato del rapporto tra Sistemi di Supporto alle Decisioni e Ricerca Operativa.
Ha posto l'enfasi sul passaggio da una Ricerca Operativa "classica" basata su modelli di tipo "valutativo" a quella basata su modelli "generativi": il primo é un modello passivo in cui il decisore descrive lo scenario, ossia assegna dei valori alle variabili che compongono il modello, osserva quindi l'output risultante dall'esecuzione del programma di ottimizzazione.
Il modello generativo é invece un modello attivo, in cui il decisore specifica le relazioni tra le variabili e il livello di qualità desiderato, ossia specifica i vincoli e l'obiettivo. In questo contesto, il modello determina una decisione.
Rispetto all'approccio classico é maggiore la consapevolezza che la soluzione ottimale di un modello, non é necessariamente la soluzione ottimale di un problema. Le linee attuali di ricerca sono quindi di minore enfasi sugli algoritmi, e di maggiore attenzione all'interfaccia uomo-macchina. La difficoltà di comprensione dei sistemi classici di ricerca operativa e la loro scarsa trasparenza viene superata grazie all'integrazione di tecniche quantitative, es. algoritmi di ottimizzazione, con tecniche qualitative, es. Sistemi Esperti.
L'impiego dell'Intelligenza Artificiale fornisce inoltre la possibilità di introdurre delle valutazioni soggettive nella risoluzione di un problema e di ridurre le barriere di comunicazioni fra il manager e i modelli della ricerca operativa.

Il tema dell'integrazione è poi stato analizzato nell'intervento di Alberto Fioravanti con la descrizione di un'applicazione reale sviluppata dalla società Inferentia s.r.l di Milano nell'ambito delle decisioni riguardanti la progettazione di isole energetiche.
Un sistema di supporto alle decisioni é visto come un sistema automatico interattivo, progettato per supportare il decisore nell'uso di dati e modelli per risolvere problemi non strutturati.
Nella definizione di una architettura del sistema descritto è stata rivolta particolare attenzione alla interattività, alla disponobilità e capacità di trattamento di dati, alla disponibilità di modelli e alle capacità di problem solving.
I principali moduli componenti il sistema dovevano essere quindi:

Interfaccia utente, (User-System Interface, USI)
Database (Data Base Management System, DBMS)
Model Base (Model Base Management System, MBMS)
Base di conoscenza (Knowledge Base, KB)

Sarebbe stato improponibile realizzare in tempi ragionevoli un sistema ad hoc, in grado di esibire in modo sufficientemente sofisticato le funzionalità appena descritte.
Si é puntato quindi alla integrazione di strumenti già disponibili e a larga diffusione. Per l'interfaccia utente é stato utilizzato New_Wave pacchetto dotato di notevoli capacità grafiche e rivolto all'integrazione delle applicazioni. Per quanto riguarda la gestione del data base SQL linguaggio di interrogazione per data_base relazionali. Infine Nexpert Object, una shell di sviluppo per Sistemi Esperti. Il sistema possiede inoltre moduli di simulazione e modelli di ricerca operativa selezionabili dall'utente.

Il punto di vista della sociologia, é stato descritto da Stefano Draghi, (Istituto Superiore di Sociologia). I primi modelli utilizzati in ambito sociologico ed economico, si basavano sul concetto di "homo oeconomicus" della teoria utilitaristica classica. I presupposti consistevano essenzialmente nel pensare al soggetto economico come un individuo in grado di scegliere sempre l'alternativa in grado di massimizzare i benefici (utilità). Ciò presupponeva, una società dova la realtà era data oggettivamente, e le regole e i vincoli erano perfettamente conosciuti e immutabili. Le critiche a questo modello hanno portato attraverso successive

evoluzioni al "garbage can model", proposto da Cohen, March e Olsen. Quest'ultimo si propone di descrivere e interpretare i processi decisionali nelle situazioni di ambiguità. Caratteristica questa di organizzazioni dotate di scarsa coesione interna e obiettivi non chiari.
Un termine che descrive assai bene questi ambiti é "anarchie organizzate". Le scelte in questo tipo di situazioni vengono spesso prese per "astrazione", ossia senza alcun riferimento ai problemi, e l'esito del processo decisionale é influenzato oltre che dagli elementi strutturali, da fattori casuali.
Il "garbage can model" tenta di dimostrare come gli aspetti caotici e casuali delle decisioni organizzative prese in condizioni di ambiguità, obbediscano in realtà ad una logica precisa.

L'ultimo intervento prima delle conclusioni é stato di Augusto Morello, consulente di pianificazione d'impresa (Politecnico di Milano).
In una società caratterizzata da molteplici ambiti, e cambiamenti repentini delle "regole del gioco", i modelli manageriali classici, che puntavano sll'organizzazione non sono più sufficienti. E' necessario puntare sulla flessibilità dei piani, e sulla capacità di adattamento dell'impresa a situazioni nuove. Si dovrebbe quindi parlare non tanto di supporto alle decisioni, ma piuttosto supporto alla creatività. Un aiuto che la tecnologia può fornire riguarda soprattutto la gestione della "diversita" che ne é l'ingrediente essenziale.
Occorre produrre una maggiore integrazione fra dati quantitativi e dati qualitativi, saper tradurre i dati in informazioni, per decidere o verificare le decisioni prese. L'obiettivo diventa così formalizzare criteri di coerenza.
Uno svantaggio delle organizzazioni non burocratizzate é però quello di basare tutto sulle persone. Una parte consistente del "patrimonio aziendale" viene perduto quando le persone abbandonano l'azienda.
Occorre garantirsi quindi con delle riserve d'intelligenza (magari artificiale).

Le conclusioni del seminario sono state tracciate da Stefania Bandini, che non ha avuto molta difficoltà a ricollegare gli interventi in un unico filo conduttore.
Da segnalare inoltre gli interventi di Vincenzo Bloise di Informatica Oggi, ed. Jackson che ha tracciato una panoramica sui prodotti esistenti oggi sul mercato, per quanto riguarda tools generali per il supporto decisionale e di Davide Pandini che ha tracciato un utile tassonomia e definizioni di terminologie nell'ambito dei DSS.

"SISTEMI ESPERTI IN BANCA E NELLE ASSICURAZIONI"

(Lugano, 2 - 3 Maggio 1989)

Roberta Gulden
Artificial Intelligence software S.p.A. Milano

Si è tenuto a Lugano, dal 2 al 3 maggio, presso il Palazzo dei Congressi, il Secondo Simposio Internazionale "Sistemi Esperti in Banca e nelle Assicurazioni", organizzato dalla SGAICO (Swiss Group for Artificial Intelligence and Cognitive Science) e dall'IDSIA (Istituto Dalle Molle di Studi sull'Intelligenza Artificiale).

L'iniziativa rientra nell'ambito dello 'Sgaico Technology Transfer', una serie di manifestazioni volte a creare un produttivo collegamento tra gli istituti di ricerca ed il mondo della produzione.

Tema del congresso: la conoscenza aziendale in ambito bancario ed assicurativo: come ottenerla, formalizzarla, utilizzarla e mantenerla. Tale conoscenza costituisce il vero patrimonio di un'azienda e la sua corretta gestione rappresenta un elemento imprescindibile per raggiungere competitività ed efficienza.
I sistemi esperti, per le caratteristiche che evidenziano, si presentano come lo strumento adeguato per trattare, manipolare e, se richiesto, esplicitare tale conoscenza.

Grazie alla partecipazione di 'figure' di riferimento del mondo dell'Intelligenza Artificiale, quali, ad esempio, John Campbell e Roger Shank, e alla presenza di numerose software house distributrici di tools per lo sviluppo di sistemi knoweledge-based e di società specializzate nello sviluppo di applicazioni di Intelligenza Artificiale, si è cercato di fornire all'utente finale un quadro complessivo sullo stato dei sistemi esperti in campo bancario ed assicurativo, sia per quanto riguarda le problematiche affrontate nel campo della ricerca, sia relativamente alla diffusione che tali sistemi hanno avuto e stanno avendo nel settore.

L'iniziativa ha suscitato un notevole interesse, testimoniato dall'alto numero di partecipanti, circa 360, molti dei quali provenienti dall'Italia.

I lavori del primo giorno sono cominciati con una relazione del presidente J. Campbell dell'University College of London che ha introdotto gran parte dei problemi affrontati negli interventi successivi, relativi al filo conduttore del congresso: la conoscenza in azienda. In particolare, rispondendo ad alcune domande di partenza, ha evidenziato la differenza esistente fra dati e conoscenze, l'importanza di una corretta formalizzazione e gestione delle stesse, l'essenzialità del ruolo del knowledge engineer.
La gestione delle conoscenze presenta problematiche differenti da quelle relative al trattamento delle semplici informazioni.
Le conoscenze, infatti, secondo la definizione data dallo stesso Campbell, sono costituite da dati a cui vengono associate informazioni sul contesto. Quindi, a livello di formalizzazione e rappresentazione delle conoscenze, si pone il problema di codificare l'associazione fra insiemi di informazioni. In questa fase occorre fare ricorso a nuovi schemi, definiti nell'area dell'Intelligenza Artificiale, quali frames, reti, etc... che si aggiungono a quelli già forniti dall'informatica tradizionale.

Il problema connesso al trattamento delle conoscenze è stato ampiamente ripreso, anche, se vogliamo, in un'ottica più 'operativa', da Klaus Brott, direttore del Gerling Konzern di Colonia, il quale, nel suo intevento, ha evidenziato come l'informatica tradizionale, nonostante gli importanti momenti di evoluzione che si sono avuti dagli anni '60 ad oggi, si sia dimostrata inadeguata nel gestire le conoscenze aziendali.

Le tradizionali procedure EDP, che costituiscono per le aziende un supporto insostituibile per il trattamento, la gestione e l'archiviazione dei dati e delle informazioni, si sono spesso dimostrate inadeguate nel trattamento delle conoscenze aziendali, che costituiscono il vero patrimonio di ogni società.
Tali procedure, infatti, hanno automatizzato solo le attività procedurali e operative che trattano dati e informazioni ma non quelle dove sono richiesti una capacità decisionale basata sulle conoscenze degli esperti dei vari settori. Conseguentemente tutte le procedure complesse e difficilmente riconducibili ad un algoritmo sono rimaste escluse dal processo informativo.
Lo stadio successivo nel processo di automazione consiste in una sua estensione alle attività che coinvolgono processi decisionali ai diversi livelli gerarchici dell'organizzazione aziendale.
Almeno due fattori hanno reso difficile, fino ad oggi, il trattamento automatico del patrimonio informativo costituito dalla cultura d'azienda: da una parte, il fatto che queste conoscenze sono difficilmente esprimibili in forma algoritmica e pertanto non sono traducibili con i tradizionali linguaggi di programmazione, dall'altra la difficoltà di reperimento delle stesse: non si tratta infatti di nozioni codificate sui libri ma del bagaglio di esperienza acquisito dagli individui che operano nella società; pertanto occorre esplicitare queste conoscenze e formalizzarle utilizzando degli strumenti diversi da quelli tradizionali.
L'apporto che un razionale utilizzo dei sistemi esperti può dare al superamento di questo problema è talmente importante da creare i presupposti per una vera evoluzione del modo di operare delle aziende.
Non solo si andrà incontro ad un accorciamento del ciclo di pianificazione produzione sfruttamento del prodotto, ma si apriranno nuove prospettive (p. es. mercato elettronico). La conoscenza va quindi vista come fattore chiave per la competitività. Klaus ha concluso il suo intervento con una attenta analisi dei fattori chiave che rallentano l'utilizzo di queste tecniche (costi, rischio, difficoltà operative).

Otto Laske del Boston College di Boston nel suo intervento ha sottolineato i problemi connessi alla gestione, al mantenimento e all'aggiornamento delle basi di conoscenza. Alla figura dell'ingegnere della conoscenza, predominante nella fase di sviluppo di un sistema esperto, si affiancherà quella del gestore della conoscenza, che interverrà in una seconda fase. La competitività si misurerà quindi in termini di flessibilità nell'acquisizione e nello sfruttamento delle nuove tecnologie.

Ha concluso gli interventi del mattino Helen Ojha, della Coopers & Lybrand di Boston (USA), che ha affrontato i temi della possibilità pratica di utilizzare queste nuove risorse. Si è proposto un tema alquanto vasto quale è la "tecnologia della conoscenza" (in alternativa o meglio in complemento alla "tecnologia dell'informazione"). Accanto alla descrizione delle limitazioni in termini di tempi operativi e di efficienza della gestione dei sistemi attuali, la Ojhia ha ribadito la necessità di creare continui collegamenti con la ricerca per poter rapidamente usufruire sul terreno pratico della continua evoluzione tecnologica.

La giornata è stata conclusa da una serie di osservazioni da Marco De Marco, dell'Università Cattolica di Milano, relative al valore della conoscenza, da intendere come vero capitale aziendale, da acquisire e far rendere

non meno degli altri investimenti. Questo è particolarmente vero nel settore bancario ed assicurativo per il quale le informazioni rappresentano la 'materia prima' del processo produttivo.

I lavori del secondo giorno hanno avuto inizio con una vivace presentazione di Erwin Franclick, membro della direzione dello Scweitzer Ruck di Zurigo, e di Ernst Lebensaft di Synologic AG di Binningen (Zh). Il tema dell'intervento, la possibilità di utilizzare operativamente la conoscenza, ha dato lo spunto per una serie di considerazioni sulla necessità di organizzare la conoscenza, e sui metodi per gestirla.
Anche in questo intervento è stato posto l'accento sulla necessità di eleborare degli schemi logici e strutturali per passare dalla semplice elaborazione dei dati alla gestione attiva dei concetti. I passi di questo processo, sintetizzati nell'acronimo Drive (Daten Realisirung Integration Vernetzung Evolution) forniscono le basi per un utilizzo cosciente e proficuo del patrimonio conoscitivo delle aziende. Non sono mancati i punti critici, volti ad individuare i punti deboli di questo sistema e soprattutto il limite di convenienza. Procedere liberi da condizionamenti e con la massima flessibilità garantirà la migliore efficienza e flessibilità del sistema. In ogni caso i risultati saranno commisurati agli investimenti, e per questo una attenta sensibilizzazione del "top management" ad investire in alta tecnologia costituirà le premesse fondamentali per il successo.

Nella seconda parte della mattinata si sono svolte in sessioni parallele esami di casi di studio proposti da produttori ed utilizzatori. Tali sessioni hanno affrontato i seguenti argomenti specifici: analisi finanziaria, valutazione del rischio associato alla concessione del credit, gestione della tesoreria, supporto agli investimenti, strumenti e metodologie di sviluppo di sistemi esperti, ingegneria della conoscenza, trasferimento di fondi, etc...

L'ultimo intervento è stato affidato a Roger Shanck, da lunghissimo tempo in prima linea nella realizzazione e nella implementazione di sistemi esperti. Shanlk, che è passato da Yale alla North West University dell'Illinois, dove sta avviando l'"Institute for Learning Sciences", ha riassunto, dalle origini dell'Intelligenza Articiale fino ai giorni nostri, le aspettattive, le difficoltà i successi e le delusioni nei confronti di questa nuova scienza. Riportando il problema della conoscenza ad una corretta rappresentazione e rintracciabilità dell'informazione, ha quindi schematizzato i passi fondamentali del processo conoscitivo che portano dalla acquisizione dell'esperienza alla capacità di ritenerla e di riproporla a richiesta. L'attenzione è quindi stata concentrata sull'apprendimento e/o sull'insegnamento che a loro volta possono essere ricondotti da una serie di regole e quindi nel preciso dominio dei sistemi esperti. In conclusione non è mancata una serie di considerazioni sulle enormi difficoltà che ancora oggi l'Intelligenza Artificiale sta affrontando per entrare con piene redditività in ambiente operativo e l'invito a cimentarsi nella realizzazione dei sistemi esperti.

La conclusione dei lavori è stata sviluppata dall'intervento di Campbell che, dopo essersi chiesto se la formalizzazione della conoscenza può già considerarsi una scienza o ancora una attività artigianale, ha puntato l'attenzione sulla evoluzione del tradizionale ciclo di vita del software al ciclo di vita dei sistemi basati sulla conoscenza. Uno sguardo al futuro ci porta a domandare se ci saranno sostanziali cambiamenti nel modo in cui noi acquisiremo la conoscenza in funzione dei metodi di rappresentazione che avremo a disposizione.

Infine ha avuto luogo una tavola rotonda in cui Franclick, Mickle, Mannesman, Lebensaft e Shanck, moderatore Campbell, hanno esposto il loro punto di vista su tre controversi argomenti:
- L'acquisizione della conoscenza è allo stato attuale una scienza esatta o ancora un lavoro artigianale?
- Se doveste eseguire da capo il vostro ultimo progetto lo rifareste allo stesso modo, alla luce delle nuove possibilità di rappresentazione della conoscenza?
- Come potrà la gente che lavora su sistemi commerciali basati sulla conoscenza considerare la scienza dell'AI?

La ricca serie di sessioni di studio che, nel corso delle due giornate, ha seguito gli interventi prima descritti o si è svolta in parallelo con essi, ha portato a conoscenza dei numerosi partecipanti una serie di esperienze reali, alcune delle quali installate e utilizzate da anni, presso istituti bancari.

NOTE DI SOFTWARE 44/45

L'alto numero di applicazioni realizzate o in fase di sviluppo è testimonianza del fatto che il settore bancario ed assicurativo risulta essere un terreno particolarmente fertile per le applicazioni di sistemi esperti.
Questo, da una parte per le peculiarità del settore, dall'altro per il tipo di attività decisionale che viene formalizzata.
Si tratta infatti di un mercato in forte espansione, caratterizzato da un vivace clima di concorrenzialità; questo crea negli istituti che vi operano la necessità di essere sempre più competitivi, sia nella qualità che nella velocità di erogazione dei servizi offerti.
Inoltre, le conoscenze da trattare sono complesse ma ben circoscritte e identificabili e, in quanto tali, facilmente formalizzabili in una base di conoscenza.

EDITO DA CALEIDOGRAF S.R.L.
VIA L. DA VINCI N.4/6 TEL. 039 - 587541
22058 OSNAGO (CO)

REALIZZATO DALLA COMPUTER AREA S.R.L. SEZ. D.T.P.
VIA A. VOLTA, 27 20058 VILLASANTA (MI)
TEL. 039 - 306081

Spedizione in abbonamento postale, Gruppo IV, 1° semestre.
Pubblicità Inferiore al 70%
N. 36, - 44/45, Giugno - Ottobre 1989
TAXE PERCUE