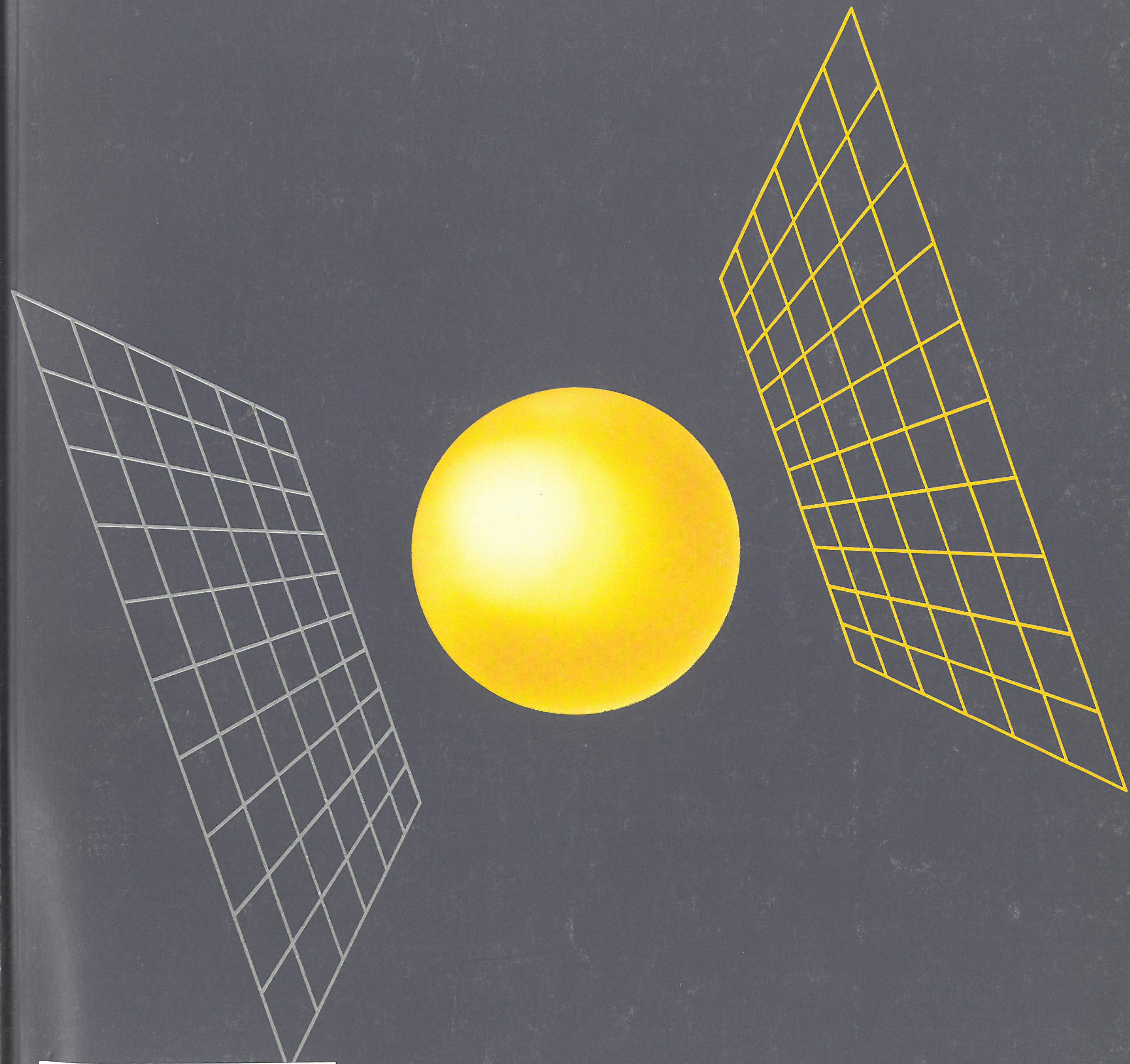
Spedizione in abbonamento postale a tariffa intera/Milano

Quaderni di informatica 26

Rassegna di tecnologia ed applicazioni degli elaboratori elettronici – Anno XII – Numero 1 – 1986



CENTRO STUDI INFORMATICA E AUTOMAZIONE
Honeywell
Honeywell Information Systems Italia

FPDM-71

Honeywell
Honeywell Information Systems Italia

Quaderni di informatica
Rassegna di tecnologia ed applicazioni degli elaboratori elettronici
Anno XII - Numero 1 - 1986

Sommario

Direttore Responsabile
F. Filippazzi

Comitato di Redazione
A. Cicu, C. Falcetti, P. Lupo,
M. Nobile, G. Occhini
F. Sala

Redazione
Honeywell Information Systems Italia
Centro Studi Informatica e Automazione
Via G.M. Vida, 11 - 20127 Milano

Stampa
tipolito Maggioni s.a.s.

Autorizzazione del Tribunale di Milano
n. 93 del 20 Marzo 1974

Elaborazione a prova di intruso: come neutralizzare i "pirati informatici"

W.E. BOEBERT
Honeywell
Secure Computing Technology Center
Minneapolis (USA)

R.Y. KAIN
University of Minnesota
Minneapolis (USA)

W.D. YOUNG
University of Texas
Dallas (USA)

1. Introduzione

Gran parte dei sistemi di elaborazione sono sicuri solo nel senso che non sono stati ancora manomessi o nel senso che i metodi per manometterli sono noti solo ad un piccolo e selezionato gruppo di persone.

La Honeywell è da molto tempo impegnata nello sforzo di accrescere la sicurezza dei sistemi di elaborazione; il MULTICS e lo SCOMP sono due sistemi che hanno fatto testo in questo senso. Il "Secure Ada Target", attualmente in fase di sviluppo al "Secure Computer Technology Center" della Honeywell, rappresenta un ulteriore passo verso un sistema veramente sicuro: ossia, sicuro secondo una ben precisa definizione e la cui sicurezza possa essere dimostrata con rigore matematico.

La macchina è chiamata Ada Target a indicare il suo primo utilizzo come ambiente per l'esecuzione di programmi scritti nel linguaggio ADA, che è stato sviluppato sotto l'egida della Honeywell ed è stato adottato dal Dipartimento della Difesa USA come linguaggio di programmazione standard ad alto livello.

In questo articolo presenteremo una panoramica dei problemi generali di sicurezza dei computer, descriveremo gli aspetti tecnici di cui il Secure Ada Target rappresenta una soluzione e spiegheremo in che senso la nostra macchina è da considerarsi sicura.

2. La minaccia

I rischi per la sicurezza dell'informazione in un sistema di elaborazione si possono grosso modo suddividere in tre classi: ci possono essere incidenti occasionali che l'utilizzatore è in grado di affrontare; oppure manomissioni dolose, o infine la concezione stessa del progetto può essere fondamentalmente insicura. Si può stabilire un'analogia con la sicurezza di un aereo: esso può essere insicuro a causa di costruzione o manutenzione difettosa; può aver subìto un sabotaggio, oppure il progetto stesso può essere intrinsecamente insicuro, ad esempio il suo baricentro potrebbe non trovarsi nel punto giusto.

Sia negli aerei che nei computer, le prime due classi di difetti presentano una casistica molto vasta e dettagliata, ma non sono sostanzialmente legate a questioni di principio: vi si può perciò porre rimedio applicando diligentemente le tecniche di controllo qualità. Ma in ambedue i casi la diligenza nelle lavorazioni e nei controlli non potrà mai sopperire a errori di progettazione.

I difetti intrinseci nella progettazione degli aerei appartengono concettualmente a diverse categorie generali, come la debolezza strutturale o l'instabilità aerodinamica. Nel caso dei computer, la categoria più frequente di difetti di progetto rende il sistema vulnerabile da parte del cosidetto "Cavallo di Troia".

Per comprendere come si manifesta l'attacco del Cavallo di Troia, occorre prima osservare che ogni conoscenza relativa ad un sistema di elaborazione viene acquisita in modo indiretto; manca l'osservazione diretta di una qualsiasi realtà fisica. La relazione tra l'uomo e la macchina è stata definita come "allucinazione amplificata dal computer", poichè le reazioni della mente umana sono innescate da eventi interni alla macchina che non sono essi stessi direttamente osservabili.

Si consideri la differenza tra una macchina per scrivere meccanica ed un sistema di word processor elettronico, due macchine che accettano entrambe la digitazione su tasti e la convertono in un testo visibile. Nella macchina per scrivere c'è una connessione meccanica diretta ed osservabile tra una determinata battuta su tasto e la comparsa della lettera corrispondente. Nel word processor manca una relazione osservabile di questo tipo. Un mediatore, od agente, sotto forma di un programma di computer, effettua la trasformazione; le azioni di questo agente non possono essere osservate direttamente dall'utilizzatore della macchina, e l'affidabilità di questo agente può essere determinata solo dall'esperienza. Ogni battuta sulla tastiera finora ha fatto visualizzare la lettera corrispondente; l'esperienza può forse convincere l'utente che la macchina sta eseguendo correttamente la funzione desiderata.

Non è tuttavia affatto certo che il programma del word processor non esegua lavori estranei contrari agli interessi dell'utente, ad esempio produzione di copie clandestine, inquinamento del testo memorizzato,

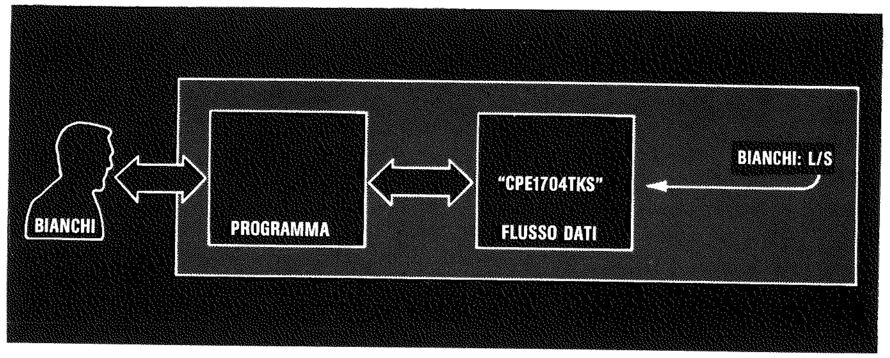


Fig. 1 - La "lista di controllo accessi" (access-control list: acl) è il mezzo usato dalla maggior parte dei sistemi di elaborazione per proteggere informazioni critiche o segrete. Ad ogni flusso dati è associato una *acl*, che abbina un utente o gruppo di utenti a modi di accesso autorizzati. In questo esempio, Bianchi, nel tentativo di proteggere la stringa di caratteri "CPE 1704 TKS" di alta criticità, ha predisposto l'acl nel flusso per consentire accessi di lettura e scrittura (L/S) a programmi operanti per suo conto e nessun accesso a programmi riservati ad altri utenti.

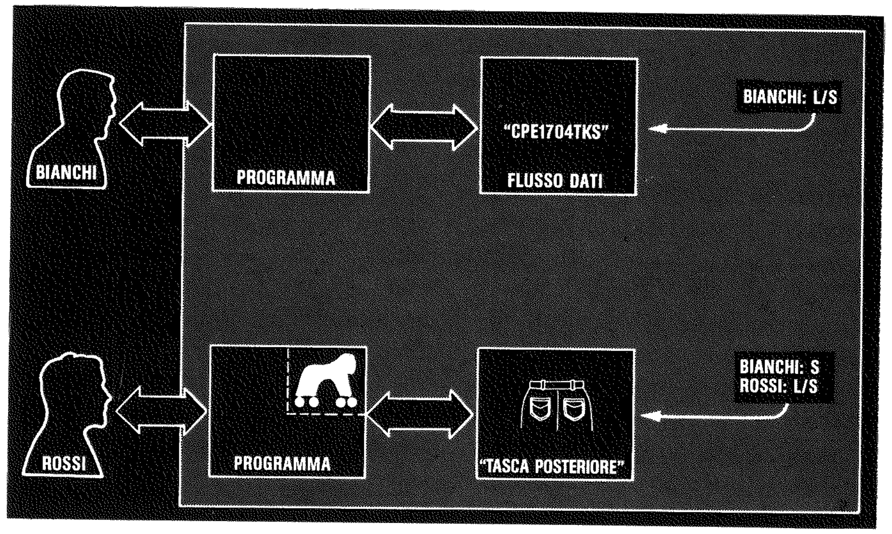


Fig. 2 - La figura rappresenta un tipo possibile di attacco ai flussi protetti dal acl. Un *utente pirata* (Rossi) prepara una incursione sui dati di Bianchi installando un programma "Cavallo di Troia" e un flusso privato. Il programma "Cavallo di Troia" svolge due funzioni. La prima è innocente e visibile all'utente che richiama il programma. La seconda funzione è ostile e invisibile. Il flusso privato serve come "tasca posteriore" (back-pocket) durante l'attacco. Rossi predispone l'acl del suo flusso per consentire accessi di lettura e scrittura a programmi operanti per suo conto, ma solo accessi di scrittura a programmi riservati a Bianchi.

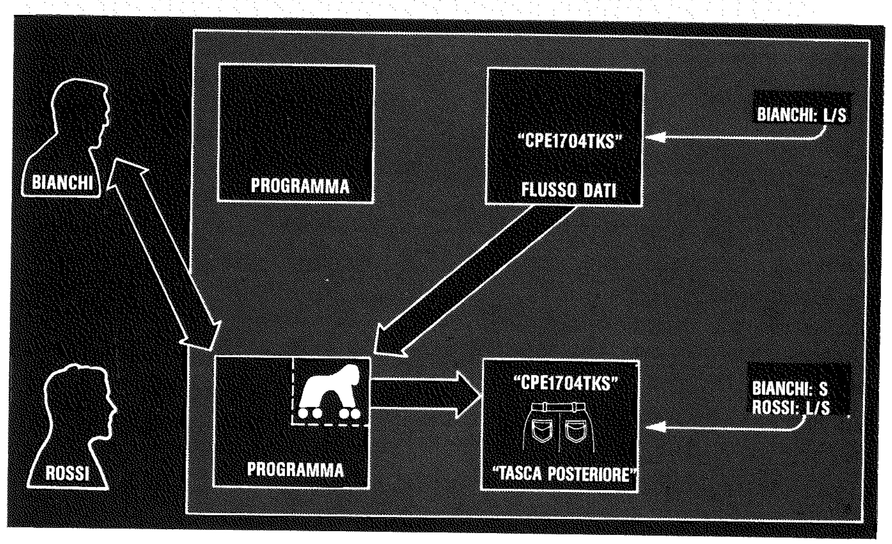


Fig. 3 - Il programma "Cavallo di Troia" è concepito per scoprire se viene eseguito dall'utente Bianchi. Quando Bianchi richiama inconsapevolmente il programma, questo acquisisce diritti di accesso per lettura e scrittura sui flussi di Bianchi e per sola scrittura sul flusso "back-pocket" di Rossi. Il "Cavallo di Troia" incorporato nel programma legge furtivamente i flussi di Bianchi e scrive i loro contenuti nel "back-pocket" del flusso di Rossi. A Rossi rimane ora soltanto di leggere il suo flusso in un secondo tempo per conoscere il valore della stringa segreta di caratteri (CPE 1704 TKS).

oppure la scansione degli input per scoprire se un documento critico o riservato è in preparazione, per poi distruggerlo una volta completato.

Questo è il primo elemento dell'attacco del Cavallo di Troia: agenti sotto forma di programmi di computer, le cui attività sono visibili solo indirettamente, se non invisibili del tutto.

Il secondo elemento è costituito da una organizzazione del computer, in cui un certo numero di tali agenti operano simultaneamente sui dati. Questa organizzazione, chiamata in vari modi, condivisione del tempo (time-sharing), condivisione delle risorse (resource-sharing), o approccio multi-utente (multiuser approach), era usata in origine solo per i più grossi sistemi.

Il rapido declino dei costi di hardware ha ormai permesso di applicarla ad hardware di basso costo, fino a circa 5.000 dollari. La diffusione dei vantaggi dell'elaborazione multi-utente (come: costo inferiore di calcolo, accesso flessibile, facilità con cui i dati possono essere utilizzati in comune) ha significato anche una maggior esposizione agli attacchi del Cavallo di Troia ed un aumento

della loro minaccia.

Già i progettisti dei primi sistemi multi-utente riconobbero la necessità di controllare la condivisione dei dati, se non altro per limitare i danni che programmi difettosi potevano provocare. Lo strumento standard per tale controllo è stato la lista di controllo degli accessi (access control list: acl). Questo strumento svolge essenzialmente una doppia funzione. Chiede anzitutto agli utenti di identificarsi con il nome (e generalmente con una parola d'ordine segreta), prima di concedere loro di richiamare un programma; perciò il sistema sa in ogni momento per conto di chi sta eseguendo un programma.

In secondo luogo, ad ogni insieme unitario di dati, come un flusso, viene associato un elenco di coppie di attributi: *nome* e *accesso*. L'insieme di tali coppie costituisce l'acl di quel insieme di dati.

La parte "nome" di un acl è il nome di un utente e la parte "accesso" definisce i modi di accesso ai dati che i programmi permessi eseguono per quell'utente. Si assuma ad esempio che i modi possibili siano "lettura" (cioè il programma può esaminare i dati nel flusso), "scrittura" (cioè il programma può modificare i dati), "lettura/scrittura" (cioè il programma può svolgere le due funzioni), e "nessuno", (cioè il programma non può nè esaminare nè modificare i dati). Prima di permettere al programma di accedere ai dati, il sistema ricerca nell'acl associato a quei dati il nome dell'utente per il quale lavora il programma. Se e quando viene trovato il nome dell'utente, il sistema consente quei modi d'accesso specificati dalla parte "accesso" dell'acl. Se invece il nome dell'utente non viene trovato, ogni accesso è negato.

Il meccanismo descritto è apparso sicuro all'intuito di molti, ma purtroppo il campo della sicurezza dei computer abbonda di risultati sorprendenti che contraddicono l'intuizione dei progettisti di sistemi; nel caso in discussione, l'analisi dell'attacco del Cavallo di Troia mostra che l'approccio acl è fondamentalmente e fatalmente errato.

Si consideri anzitutto la situazione illustrata nella figura 1. L'utente Bianchi interagisce tramite programma con un flusso dati contenente la stringa dati altamente critica o segreta "CPE1704 TKS".

Bianchi ha applicato scrupolosamente il metodo acl per garantire che solo i suoi programmi possano ottenere l'accesso al flusso critico.

L'attacco del Cavallo di Troia inizia quando un utente pirata, che chiameremo Rossi, ottiene in modo legittimo l'accesso al sistema ed inserisce un programma contenente il Cavallo di Troia e un flusso privato da usare nell'attacco come "back-pocket" (tasca posteriore). Rossi predispone l'acl di tale flusso privato per consentire ai programmi operanti per suo conto un accesso illimitato al flusso stesso ed ai programmi di Bianchi la possibilità di scrivere sul flusso. Rossi si guarda bene dall'informare Bianchi della cosa. Questa situazione è illustrata nella figura 2.

Rossi induce ora Bianchi a richiamare il Cavallo d' Troia. Dispone di tanti modi diversi per raggiungere questo scopo. Può ad esempio inserire il Cavallo di Troia in un programma di utilità frequentemente usato oppure in un video-gioco divertente.

Quando il Cavallo di Troia scopre che il programma in cui è inserito viene eseguito per conto di Bianchi, copia la stringa di caratteri critica dal flusso di Bianchi su quello di Rossi. Questa trascrizione, illustrata nella figura 3, avviene in assoluto rispetto dei vincoli imposti dal metodo acl; avviene cioè nonostante che il metodo sia applicato in modo assolutamente regolare ed usato correttamente. A Rossi basterà ora accedere semplicemente al suo flusso in un tempo successivo per conoscere il valore della stringa.

Questo esempio dovrebbe chiarire il principio e l'efficacia degli attacchi del Cavallo di Troia: essi sono emblematici dell'uso improprio di diritti correttamente concessi, da parte di un pirata le cui azioni non sono direttamente osservabili.

Una forma particolarmente insidiosa del Cavallo di Troia è il cosidetto programma "virus", che installa una copia di se stesso in altri programmi, propagandosi in tal modo in tutto il sistema e perfino nell'intera rete di computer. Basterà richiamare il virus una sola volta per infettare gli altri programmi che sono normalmente usati dall'utente preso di mira; così, ogni richiamo provoca ulteriori richiami e nuove occasioni di creare scompiglio.

3. Principi fondamentali di sicurezza dei computer

Alla fine degli anni 60 si cominciò a lavorare allo sviluppo di principi sistemistici che sventassero gli attacchi del Cavallo di Troia. Questo lavoro, svolto in gran parte presso la System Development Corporation, la Case Western Reserve University, la Mitre Corporation e la Air Force Electronic Systems Division, condusse al concetto di "reference monitor", alla definizione della "normativa obbligatoria di sicurezza" (mandatory security policy) e alla formulazione del modello di Sicurezza "Bell e la Padula". Descriveremo questi concetti uno per uno; quindi ritorneremo al nostro esempio per dimostrare in che modo essi possono sventare gli attacchi del Cavallo di Troia.

Il reference monitor, chiamato talvolta "security kernel" (nucleo di sicurezza), è un sottosistema che ha il compito di verificare la legittimità del tentativo di un programma di accedere ai dati. In base a questa definizione, il reference monitor dovrebbe avere le seguenti caratteristiche: deve essere inviolabile, costruito in modo da non potere essere aggirato e sufficientemente piccolo e semplice da poter essere provato in modo esauriente. In tempi più recenti è diventato praticamente possibile rafforzare il terzo requisito, nel senso di poter provare a priori che il reference monitor funzioni correttamente. In ogni caso, questo requisito riflette le convinzioni sulla criticità della sicurezza; il fatto che un sistema non si sia dimostrato vulnerabile, non può considerarsi prova sufficiente che esso sia sicuro.

Il secondo elemento, la normativa obbligatoria di sicurezza, consiste nella formulazione di restrizioni sul

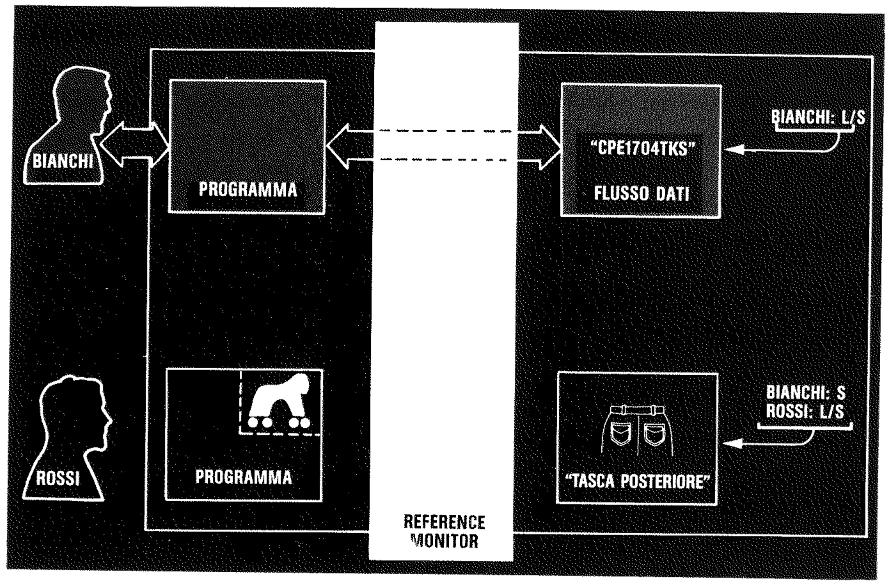


Fig. 4 - Il computer sicuro si avvale di un sottosistema chiamato "reference monitor" che ha la funzione di verificare la legittimità di ogni tentativo di accedere alle informazioni. Il reference monitor prende decisioni di accesso basandosi sui livelli di sicurezza. Ogni soggetto, come un programma di elaborazione, e ogni oggetto, come un flusso, ha un assegnato livello di sicurezza.
In questa figura i livelli di sicurezza sono due: critico (grigio) e pubblico (bianco). Il reference monitor mette in atto due regole: un programma non può leggere un flusso con un più alto livello di sicurezza (Proprietà di Sicurezza Semplice), nè può scrivere su un flusso con livello inferiore di sicurezza (Proprietà*). In questo esempio il programma operante per conto di Bianchi è autorizzato ad accedere al flusso contenente la stringa di caratteri critica, in quanto il programma ed il flusso hanno lo stesso livello di sicurezza.

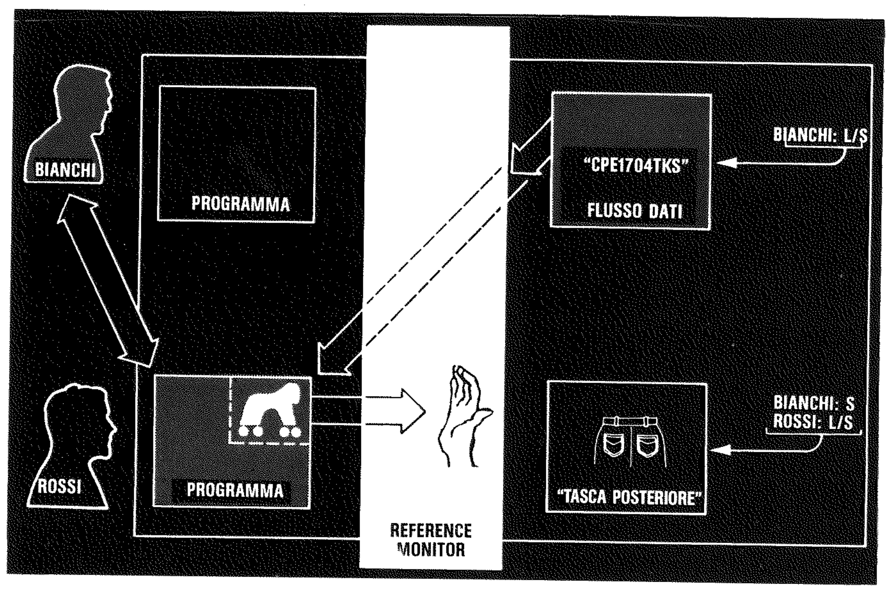


Fig. 5 - Il reference monitor neutralizza il Cavallo di Troia. Quando Bianchi richiama inconsapevolmente il programma con "Cavallo di Troia", esso acquisisce il proprio livello di sicurezza e, poichè opera a livello "critico", ha lo stesso livello di sicurezza del flusso di Bianchi. Essendo uguali i livelli di sicurezza, la Proprietà di Sicurezza Semplice viene tutelata e il reference monitor autorizza il programma a leggere il flusso di Bianchi. Poichè tuttavia il programma opera a livello "critico", ha un livello di sicurezza superiore rispetto al flusso "back-pocket". Poichè la scrittura da un livello superiore ad uno inferiore trasgredirebbe la Proprietà *, il reference monitor la blocca. Si osservi che il reference monitor blocca operazioni che l'acl consentirebbe; infatti blocca la scrittura mentre l'acl per il flusso "back-pocket" la permette.

trasferimento di informazioni che il reference monitor deve far applicare. Il termine "obbligatorio" sottolinea il fatto che tale normativa non è una opzione.

La normativa obbligatoria di sicurezza caratterizza le restrizioni di accesso in termini di azioni da parte dei soggetti.

"Soggetto" è un agente attivo, come un programma in esecuzione, che lavora per conto di un utente. Ad ogni soggetto viene assegnato un livello ben determinato ed immutabile di sicurezza, espresso da un valore appartenente ad un insieme di valori parzialmente ordinato. (Un valore di tale insieme può essere maggiore o minore di, uguale a, o non confrontabile con, gli altri valori dell'insieme. In un insieme completamente ordinato invece, tutti i valori devono essere confrontabili). Il livello di sicurezza di un soggetto riflette il grado di confidenza riposto nello utente per conto del quale opera il soggetto stesso.

La formulazione generale della normativa obbligatoria di sicurezza postula che un soggetto di alto livello non possa trasferire informazioni ad un soggetto di livello più basso o non confrontabile, a meno che tale trasferimento risponda alla volontà di un utente autorizzato. (Per "autorizzato" si intende autorizzato a provocare tali flussi discendenti).

Il trasferimento di informazioni da un livello più alto di sicurezza ad uno più basso è chiamato "downgrading"; il trasferimento al livello minimo è chiamato "declassification". Il terzo ed ultimo elemento, il modello ideato da D. Elliot Bell e da Leonard J. La Padula della Mitre Corporation, traduce gli obiettivi della normativa obbligatoria di sicurezza nelle regole che devono essere seguite dal reference monitor. Prima di esporre tali regole occorre introdurre altri due concetti: "oggetti" ed "accessi". Gli "oggetti" sono enti passivi, portatori di informazioni, come gli archivi. Ad ogni oggetto deve essere assegnato un livello di sicurezza, preso dallo stesso insieme di valori che si possono assegnare ai soggetti. Gli "accessi" sono operazioni simili a quelle descritte nella trattazione dell'acl: lettura, scrittura, ecc.

Il modello di Bell e la Padula descrive la normativa obbligatoria di sicurezza in termini di "Operazioni" (come decisioni di concedere o negare l'accesso) anzichè in termini astratti (come flusso delle informazioni). Nella versione semplicistica che useremo per la nostra esposizione, esso afferma che il reference monitor è sicuro, nel senso definito nella trattazione della normativa obbligatoria, solo se presenta due proprietà, chiamate rispettivamente "Proprietà di Sicurezza Semplice" (Simple Security Property) e "Proprietà*" (*-Property) ovvero "Proprietà Stella", in cui * è un carattere fittizio.

Nessuno è riuscito a dare un nome a questa proprietà durante la stesura del primo report sul modello. L'asterisco era un carattere fittizio nella bozza del report in modo che il text editor potesse rapidamente individuarlo tutte le volte che era usato e sostituirlo non appena che alla proprietà fosse assegnato un nome. Non si è invece mai riusciti a darle un nome e così il report è stato pubblicato con il carattere*.

La Proprietà di Sicurezza Semplice stabilisce che il reference monitor conceda ad un soggetto l'accesso per lettura su un oggetto solo quando il livello di sicurezza del soggetto è maggior del, od uguale al (cioè prevale sul) livello di sicurezza dell'oggetto. La Proprietà * stabilisce che i diritti di scrittura siano concessi solo quando il livello di sicurezza dell'oggetto su cui si scrive è maggiore del, od uguale al livello di sicurezza del soggetto che esegue la scrittura. Nel loro insieme, le due proprietà limitano l'accesso di soggetti ad oggetti, alla lettura in giù od alla scrittura in su (cioè rispettivamente per livelli descrescenti o crescenti di sicurezza). Poichè l'unico modo per ottenere le informazioni memorizzate in un sistema di elaborazione è quello di accedere ad un oggetto intermedio, l'effetto combinato delle due proprietà sarà quello di garantire che le informazioni possano muoversi solo verso l'alto.

Ritorniamo ora al nostro esempio, inserendo un reference monitor e assegnando livelli di sicurezza ai soggetti e agli oggetti; dimostreremo che l'organizzazione di sistema che ne consegue è a prova degli attacchi del Cavallo di Troia.

I livelli di sicurezza vengono normalmente assegnati ai soggetti nella fase di identificazione ovvero presentazione, usando come criteri lo specifico terminale attraverso cui si accede alla macchina e l'utente associato al soggetto. L'identità dell'utente è generalmente confermata dalla sua conoscenza di una parola d'ordine segreta. I metodi qui descritti non proteggono contro la perdita o la sottrazione delle parole d'ordine, allo stesso modo che nessuna serratura protegge contro chiavi rubate.

L'organizzazione del sistema è illustrata in figura 4. Abbiamo due livelli di sicurezza, "critico" e "pubblico", ordinati secondo il concetto che il livello critico è più elevato del pubblico. I soggetti operanti per conto di Bianchi ed il flusso contenente l'importante stringa di caratteri, hanno assegnato il livello di sicurezza critico. Il flusso di Rossi ed i soggetti operanti per suo conto sono invece al livello di sicurezza pubblico.

La figura 5 mostra cosa succede quando Bianchi richiama inconsapevolmente il programma contenente il Cavallo di Troia. Come in precedenza, il programma acquisisce i diritti associati a Bianchi ed opera ad un livello di sicurezza critico. È pertanto in condizione di osservare, in base alla proprietà di Sicurezza Semplice, la stringa di caratteri "CPE1704TKS". Tuttavia, quando il programma tenta di registrare la stringa in un flusso pubblico, violandosi la Proprietà *, il tentativo viene frustrato dal reference monitor. Così si rompe un anello della catena del Cavallo di Troia, "copia e vedi poi". Il lettore può verificare facilmente che le regole frustrano anche altre due possibilità: Rossi che si presenta al sistema e tenta di leggere direttamente la stringa e Rossi che assegna un livello di sicurezza critico al flusso "back-pocket".

Si noti che il sistema illustrato, analogamente a gran parte dei sistemi reali, conserva il metodo acl. La realizzazione della normativa obbligatoria di sicurezza secondo l'interpretazione del modello di Bell e La Padula, prevale però sul metodo acl. Ad esempio il tentativo di trascrizione nella "back-pocket" viene frustrato anche se il metodo acl lo autorizza. Il metodo acl, chiamato anche "normativa discrezionale di accesso", viene conservato principalmente come comodità amministrativa; esso consente ad un utente di limitare l'accesso su un oggetto, ad un sottogruppo dei soggetti che si vedrebbero invece accordare l'accesso dalla normativa obbligatoria di sicurezza.

4. Prove di sicurezza

Come si è accennato, il reference monitor, deve presentare tre attributi, per potersi definire sicuro: deve essere inviolabile, non deve poter essere aggirato e deve potersi dimostrare corretto. Di questi tre requisiti, l'ultimo è di gran lunga il più difficile da rispettare.

I primi tentativi di dimostrare corretto il reference monitor si affidavano a strumenti automatizzati di prova. La descrizione del reference monitor veniva scritta in una notazione speciale, chiamata "specification language". La descrizione, chiamata "Specifica Formale ad Alto Livello" (Formal Top-Level Specification), era esplorata da un programma che generava statement logici riguardanti lo stato di sicurezza del sistema descritto. L'insieme di regole che presiedevano alla generazione degli statement logici era ritenuto capace di garantire che, se tutti gli statement erano veri, il sistema specificato sarebbe stato sicuro nel senso definito dal modello di Bell e La Padula. Gli statement logici erano poi dimostrati veri, con l'ausilio di un altro programma del computer.

Richard A. De Millo del Georgia Institute of Technology e Richard J. Lipton e Alan J. Perlis della Yale University, hanno aspramente criticato le prove risultanti, sostenendo che sono complesse ed incomprensibili, che non esiste alcuna garanzia che l'universo degli argomenti abbracciato da una prova copra il sistema effettivamente implementato e che il procedimento di prova si fonda su strumenti la cui affidabilità non è maggiore (anzi in molti casi è sensibilmente minore) di quella del sistema in questione.

Noi abbiamo tenuto conto di queste critiche e abbiamo preso la risoluzione, fin dall'inizio del nostro lavoro con il Secure Ada Target, che le nostre prove sarebbero state verificate dalla macchina ma non generate da essa. In particolare ci impegnammo ad offrire argomentazioni tradizionali ed informali che chiarissero il perchè ed in che senso il nostro sistema fosse sicuro, e che fossero comprensibili ai nostri colleghi. Tale prova potrebbe essere sottoposta allo stesso esame critico cui è esposta una dimostrazione matematica tradizionale. Allo scopo di incoraggiare tale esame, abbiamo usato strumenti informatici basati su principi comunemente accettati; i "giudici" possono così ripercorrere i nostri passi con un'insieme diverso di strumenti, riducendo così il rischio che il nostro sistema apparentemente corretto sia stato validato erroneamente da uno strumento difettoso.

Tenteremo ora di tracciare il filo del ragionamento su cui sono basate le nostre prove. Per far questo occorre introdurre tre concetti: livelli di astrazione, proprietà e correlazioni. Li illustreremo ricorrendo ancora all'analogia dell'aeroplano.

Si consideri il progetto costruttivo di un aereo commerciale. L'affermazione più generale che possa essere fatta su tale aereo è che deve essere sicuro e utile. Si tratta di proprietà il cui grado di generalità è il loro livello di astrazione. A questo elevato livello di astrazione, si può far ricorso solo all'intuizione e a principi fondamentali. Sarebbe inumano produrre un aereo insicuro. La proprietà di utilità è invece richiesta per non cadere nell'assurdo, poichè un aereo che non si muove può evidentemente essere costruito in modo da essere sicuro.

Abbiamo dunque a questo livello di astrazione due proprietà (sicurezza ed utilità) ed una descrizione (aereo). Per proseguire dobbiamo precisare meglio i termini del problema, cioè scendere di livello di astrazione e scrivere la descrizione tradizionalmente chiamata specifica di ingegneria, con l'autonomia di volo, il carico pagante e tutte le altre caratteristiche dell'aereo. Quanto più dettagliata la nostra descrizione dell'aereo, tanto più dettagliate dovranno essere le nostre descrizioni di sicurezza ed utilità. Così possiamo stabilire che l'aereo debba poter volare da New York a Los Angeles senza scalo intermedio ed abbia inoltre sufficiente riserva di carburante per raggiungere San Francisco in caso di cattivo tempo. Poi verifichiamo che l'entità descritta dalla specifica di ingegneria presenti le proprietà definite, avvalendoci a questo scopo dell'ispezione, revisione e del ragionamento matematico.

Abbiamo ora due descrizioni a due livelli di astrazione, e dobbiamo tracciare tra loro la mappa delle correlazioni, cioè mostrare che la più dettagliata delle due interpreta correttamente l'altra di tipo più generale. Così esaminiamo la nostra specifica e determiniamo se descrive effettivamente una entità che è comunemente riconosciuta come un aeroplano. Analogamente si devono correlare le varie proprietà. Esaminiamo le nostre definizioni e determiniamo se la possibilità di volare da New York a Los Angeles è un affinamento conveniente di "utile" e se il requisito che l'aereo possa arrivare a San Francisco sia un affinamento conveniente di "sicuro".

Con lo stesso procedimento discendiamo la scala dei livelli di astrazione, arrivando infine al livello più basso: un aereo reale, utilizzato per il trasporto effettivo di persone.

Le nostre prove sulla sicurezza dei computer adottano precisamente questo paradigma. Possiamo invero vedere come esso lavora nella nostra precedente trattazione sulla sicurezza. Al più alto livello di astrazione abbiamo una descrizione generale (computer multi-utente) ed una proprietà generale (sicurezza). Poi abbiamo correlato i vari elementi fino al livello della normativa obbligatoria di sicurezza, dove il computer multi-utente è divenuto un insieme interattivo dei soggetti e dei livelli di sicurezza loro associati, e il requisito di sicurezza una regola per il flusso delle informazioni fra livelli. Abbiamo sostenuto che il sistema come era stato descritto era sicuro per definizione, tracciando quindi la nostra mappa giù fino al livello del modello di Bell e La Padula. A quel livello, abbiamo introdotto nella descrizione i concetti di oggetto e di modo d'accesso e tracciato la nostra mappa sulle restrizioni al flusso delle informazioni, giù fino alla Proprietà di Sicurezza Semplice ed alla Proprietà *. Abbiamo poi enunciato due argomentazioni: la prima, essenzialmente una tautologia a questo livello, che il reference monitor presentasse queste proprietà; la seconda, tutt'altro che una tautologia, che le due proprietà fossero una interpretazione appropriata della regola al livello superiore.

A questo punto l'interpretazione di "aereo" coinvolge migliaia di parti assemblate, e l'interpretazione di "sicuro" e "utile" chiama in causa migliaia di procedure e specifiche. Entrambe le interpretazioni sono complesse al punto da risultare incomprensibili; ciononostante noi voliamo fiduciosi poichè siamo scesi metodicamente con la nostra mappa delle correlazioni giù fino a questo punto, per passi comprensibili.

Seguiremo lo stesso paradigma anche nel prosieguo della nostra trattazione sulla macchina Secure Ada Target, discendendo la scala dei livelli di astrazione e descrivendo le mappe associate. Nel progetto ora configurato esistono altri quattro livelli, denominati (a partire dal più alto) Modello Astratto, Interpretazione, Specifica Formale ad Alto Livello e Specifica di Progetto.

Prima però di considerare i livelli più minuti del nostro progetto, occorre accennare ad uno dei problemi più scottanti dei computer, quello della fidatezza dei soggetti.

Bell e La Padula si resero conto che la Proprietà di Sicurezza Semplice e la Proprietà * interpretavano in modo incompleto la normativa obbligatoria di sicurezza. Queste due proprietà garantiscono che le informazioni non scendano di livello di sicurezza, ma non permettono neanche il movimento autorizzato verso il basso. Jonathan K. Millen della Mitre Corporation ha dimostrato che un sistema concreto deve permettere questi flussi; non è generalmente accettabile un sistema di elaborazione costruito in modo che le informazioni si muovano esclusivamente verso l'alto, cioè per livelli di sicurezza ascendenti, pena l'impossibilità di eseguire un lavoro utile. Nel modello è stata perciò incorporata una deroga alle due proprietà, basata appunto sulla fidatezza dei soggetti.

Un soggetto fidato è esentato dalla Proprietà *, cioè è in grado di scrivere selettivamente informazioni verso il basso. Ogni soggetto di questo tipo deve dimostrarsi fidato indipendentemente dagli altri. Si deve dimostrare che i programmi eseguiti dal soggetto fidato sono inviolabili e non si può mai far loro trasgredire la normativa obbligatoria di sicurezza.

Quest'ultima fase esige una mappa di collegamento tra i programmi richiamati dal soggetto e la normativa obbligatoria di sicurezza. Il tracciamento di tale mappa comporta solitamente che i flussi discendenti siano identificati e ben compresi e che riflettano fedelmente le intenzioni di un utente autorizzato. I soggetti fidati possono pertanto essere concepiti come estensioni speciali del reference monitor.

Come ogni eccezione ad un complesso di regole, può verificarsi un abuso di soggetti fidati, che potrebbero diventare, come li ha chiamati Roger R. Schell, quando lavorava al DoD Computer Security Center, "il tappeto sotto cui si scopa di tutto". Il Secure Ada Target si è proposto appunto l'obiettivo importante di ridurre drasticamente il numero di soggetti fidati richiesti per un lavoro utile, di limitare i loro privilegi (riducendo quindi la quantità di prove che bisogna produrre a loro riguardo) e di garantire che, se svolgono funzioni critiche per la sicurezza, essi non possano essere aggirati.

5. Principi fondamentali del Secure Ada Target

Passiamo ora a discutere i livelli di astrazione, introducendo in essi abbastanza dettagli da rendere comprensibili le nostre decisioni progettuali. Cominciamo con il cosidetto livello di Modello Astratto, cioè il livello in cui si dà al reference monitor una struttura interna di massima.

A questo livello il sistema totale consiste di tre "spazi di stato", ovvero tre insiemi di informazioni immagazzinate. (Questa caratterizzazione del sistema è stata una estensione del lavoro di John M. Rushby del SRI International). I tre spazi di stato sono chiamati "stato dei valori", "stato delle protezioni" e "astrazioni sottostanti". Lo stato dei valori consiste di tutti gli oggetti che possono eventualmente essere resi visibili ai soggetti; si trova perciò al di fuori del reference monitor. Lo stato delle protezioni e le astrazioni sottostanti sono le informazioni che il reference monitor tiene nel suo interno e utilizza per prendere decisioni sugli accessi. A questo livello lo stato delle protezioni si presenta come una matrice e le astrazioni sottostanti come tabelle.

Lo stato delle protezioni è una matrice le cui righe sono i soggetti, le colonne sono gli oggetti e le entrate specificano i modi di accesso ad ogni oggetto attualmente concesso ad ogni soggetto. Le astrazioni sottostanti a questo livello, sono tabelle che forniscono gli attributi dei soggetti e degli oggetti, come i loro livelli di sicurezza e le loro entrate di acl. Questi attributi servono per decidere quale valore deve apparire in una determinata cella dello stato protezioni.

Introduciamo infine il concetto di operazione richiamata da un soggetto ed eseguita dal reference monitor. Questa operazione consulta le astrazioni sottostanti ed aggiorna in conformità lo stato delle protezioni. Nel loro complesso le nuove definizioni danno un primo abbozzo alla struttura del sistema. Abbiamo infatti un reference monitor con una interfaccia verso i soggetti, le informazioni che deve proteggere e le informazioni che gli servono per prendere decisioni sulle protezioni.

Tracciamo poi la mappa del modello di sicurezza di Bell e La Padula, giù fino a tre proprietà. Esaminiamo qui le due proprietà chiamate Proprietà Obbligatoria di Sicurezza (Mandatory Security Property) e Proprietà Discrezionale di Sicurezza (Discretionary Security Property). La terza proprietà, chiamata Imposizione Tipo (Type Enforcement), è il mezzo per limitare i soggetti fidati, facilitare l'applicazione di politiche di sicurezza complesse basate su aggregazioni e deduzioni e per garantire che sottosistemi critici (come quelli che contrassegnano l'output con l'appropriato livello di sicurezza) non possano essere aggirati. La descrizione del meccanismo di type enforcement non rientra tuttavia nello scopo di un articolo introduttivo come il presente.

La Proprietà Obbligatoria di Sicurezza postula che lo stato delle protezioni sia sicuro, nell'accezione di Bell e La Padula, in ogni circostanza; cioè lo stato delle protezioni sia sempre in accordo con la Proprietà di Sicurezza Semplice e con la Proprietà*.

La Proprietà Discrezionale di Sicurezza garantisce che un soggetto operante per conto di un determinato utente non possa accedere ad un oggetto a meno che il nome di quell'utente non appaia nell'acl dell'oggetto nel momento in cui il soggetto dichiara la sua intenzione di accedere all'oggetto.

Il nostro metodo di prova, a questo e ad ogni altro livello di astrazione, comporta due fasi. Anzitutto, un determinato livello di astrazione deve essere verificato: bisogna dimostrare cioè che la descrizione al livello in questione presenta le proprietà di sicurezza formulate per quel livello. In secondo luogo, le proprietà devono essere correlate verso l'alto: bisogna dimostrare cioè che la dichiarazione di sicurezza al livello in questione è una interpretazione appropriata della dichiarazione di sicurezza al livello immediatamente superiore.

Le verifiche sono basate sull'induzione matematica.

Anzitutto dimostriamo che il valore iniziale dello stato delle protezioni è sicuro. Quindi dimostriamo che ogni operazione descritta ad un dato livello preserva la sicurezza: se essa viene richiamata quando lo stato delle protezioni è sicuro, allora questo sarà sicuro dopo il richiamo. Abbiamo in tal modo dimostrato che lo stato delle protezioni è sicuro attraverso tutte le possibili sequenze di operazioni. Questo uso dell'induzione matematica ci offre il vantaggio di prove teoricamente esaurien-

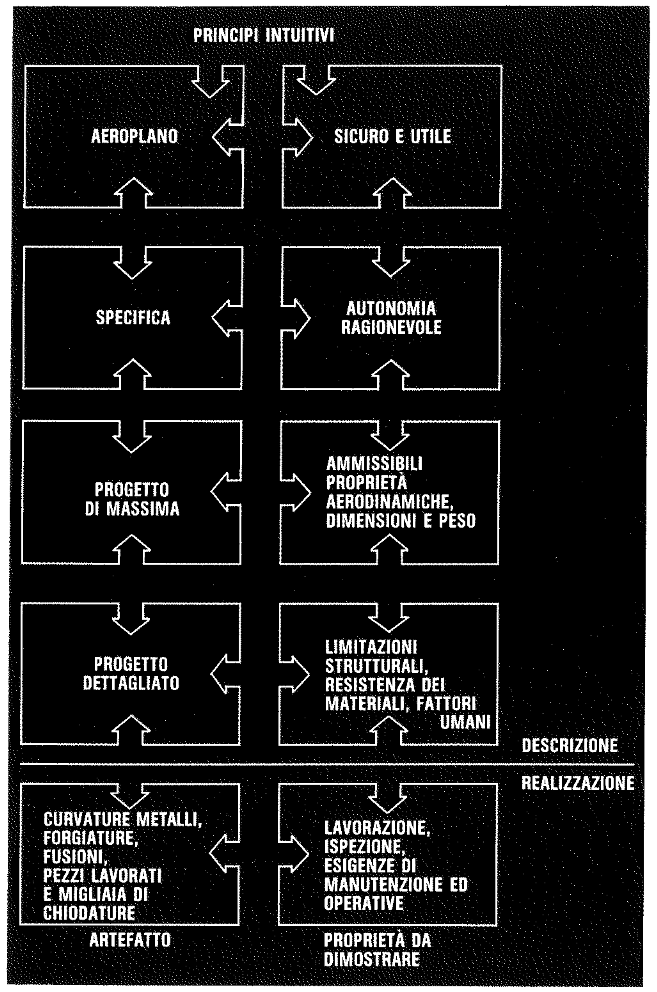


Fig. 6 - Le fasi di progettazione di un computer sicuro possono essere paragonate a quelle di un aereo sicuro. Il progetto finale è il risultato conclusivo di una serie di descrizioni via via più dettagliate e meno ambigue della nozione di aeroplano (colonna di sinistra). Si dimostra che ciascuno di questi artefatti ingegneristici ha un diverso insieme di proprietà (colonna di destra). Esse sono definizioni via via più dettagliate della nozione di sicuro ed utile.
L'aereo, una volta costruito, è complesso al punto dell'incomprensibilità; tuttavia noi voliamo fiduciosi poichè il progetto finale è derivato da principi intuitivi attraverso passi comprensibili.

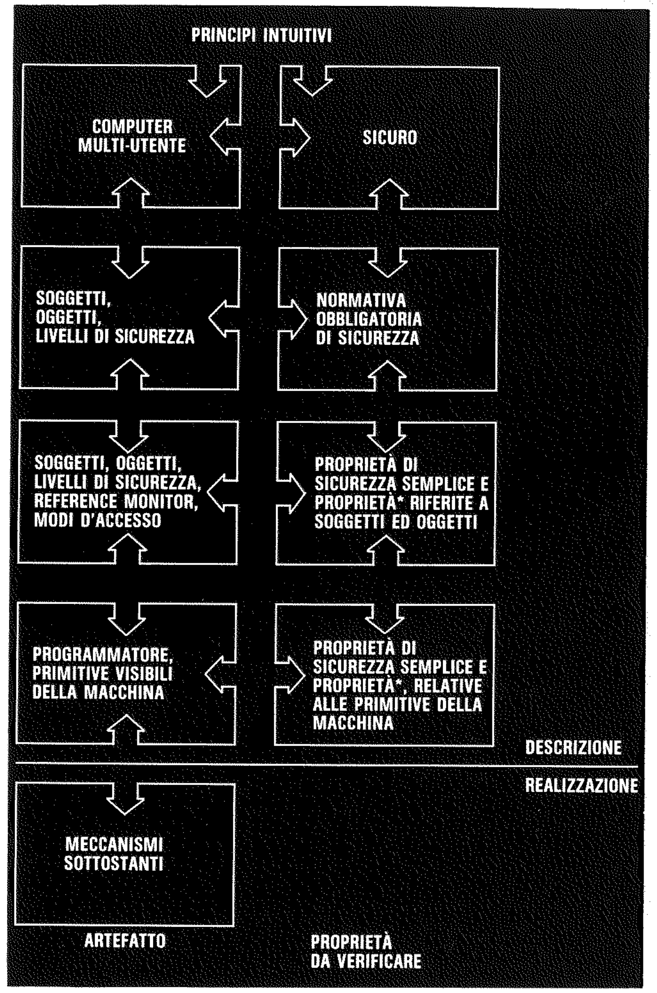


Fig. 7 - La prova di sicurezza del "Secure Ada Target" è strutturata in modo simile a quella utilizzata per progettare gli aerei. La struttura consiste di una serie di rappresentazioni via via più dettagliate della nozione di computer multi-utente (colonna di sinistra), ciascuna delle quali è abbinata ad una definizione altrettanto dettagliata della nozione di sicurezza (colonna di destra). La prova di sicurezza comporta due enunciati per ogni livello. Primo, la descrizione del sistema deve dimostrare di presentare le proprietà di sicurezza formulate per quel livello (frecce orizzontali). Secondo, la proprietà di sicurezza deve dimostrare di interpretare correttamente la proprietà al livello immediatamente superiore (frecce verticali nella colonna di destra).

ti, risparmiando enormemente sul tempo e la fatica richiesti da queste prove.
La seconda parte della prova, il tracciamento della mappa tra le nostre proprietà ed il modello di Bell e La Padula, è banale a questo livello di astrazione, ma diventa più complesso con l'aumento dei dettagli. In generale le mappe sono basate sulla rappresentazione di entità particolari a diversi livelli di astrazione. A questo livello di astrazione, ad esempio, i livelli di sicurezza sono memorizzati in una tabella e sono designati come elementi di tabella; al livello immediatamente inferiore, essi possono essere raccolti in due flussi (flusso soggetti e flusso oggetti) per efficienza di accesso. L'enunciato della Proprietà di Sicurezza Semplice a questo livello postula che: livello [soggetto] $\geq$ livello [oggetto]. Al livello inferiore la medesima clausola può essere espressa così:
flusso livelli soggetto, entrata soggetto [n]$\geq$ flusso livelli oggetto, entrata oggetto [m].
La mappa deve essere accompagnata dall'argomento che l'organizzazione flusso è l'equivalente della tabella.

6. L'approccio ingegneristico al Secure Ada Target

Ai livelli inferiori di astrazione cominciamo ad affrontare la realtà dell'ingegneria. Ovviamente non esiste alcuna macchina pratica od economica con cui consultare una grande matrice per ogni accesso ai

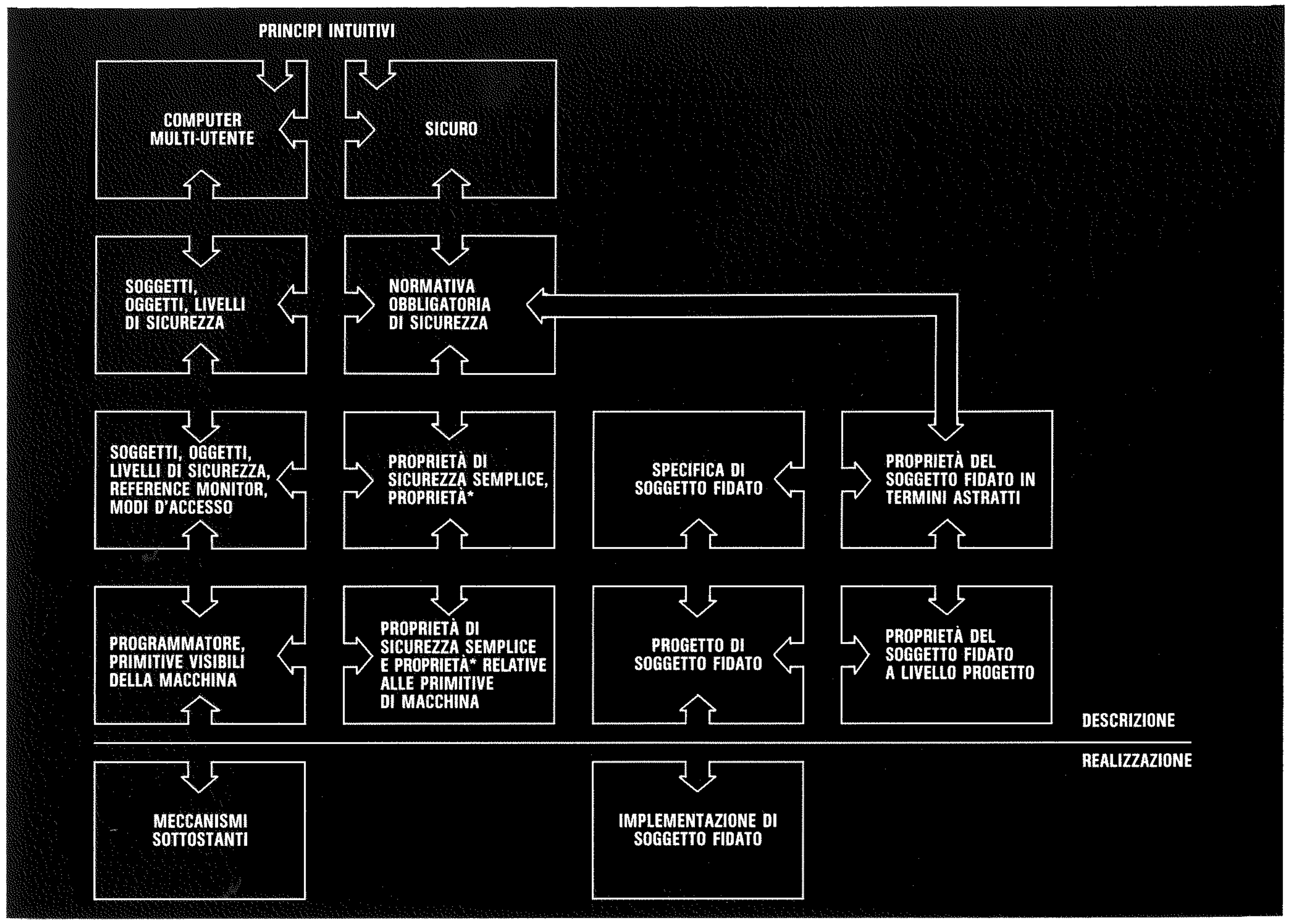


Fig. 8 - I "soggetti fidati" richiedono l'estensione della prova di sicurezza del sistema di elaborazione. La normativa obbligatoria di sicurezza autorizza flussi di informazione per livelli decrescenti di sicurezza se tali flussi rispecchiano fedelmente le intenzioni di utenti autorizzati. La Proprietà di Sicurezza Semplice e la Proprietà Stella (*), interdicono invece qualsiasi flusso verso il basso.

dati; dobbiamo riuscire a realizzare uno stato delle protezioni che sia contemporaneamente pratico e sicuro e ci consenta inoltre di dimostrare qualcosa che fu omesso nella trattazione delle mappe e delle verifiche: cioè che le operazioni dei soggetti sugli oggetti sono davvero limitate dallo stato delle protezioni. Infine dobbiamo stabilire in che modo realizzare le operazioni del reference monitor.

Tutte queste decisioni comportano la necessità di complicati compromessi ingegneristici. Ogni compromesso deve anzitutto poter essere provato, vale a dire che nessuna funzionalità, per quanto allettante per altri motivi, può essere incorporata se non consente la prova nel senso da noi definito.

Nella progettazione del computer sicuro, come del resto di ogni altro computer, gran parte dei compromessi riguardano la temporizzazione dei vincoli. Si tratta di decidere quando, in una successione di eventi, deve essere stabilita l'associazione, o vincolo, tra due valori particolari. In generale, più tardi si vincola, più il sistema risulterà flessibile, ma anche più costoso. Viceversa, più si anticipa il momento del vincolo, più si risparmia in hardware ma più rigido risulterà il sistema. Questi problemi, come pure il funzionamento dell'hardware del Secure Ada Target, possono essere chiariti prendendo in esame il problema generale dell'estrazione ed immagazzinamento dei dati.

Ad un livello sufficientemente basso di astrazione, una architettura semplice di computer si può immaginare composta da un registro di lavoro, in cui vengono memorizzati i dati temporanei, e da una schiera lineare di celle in cui vengono memorizzati i risultati più permanenti. La selezione di un elemento della schiera avviene dando la sua posizione nella schiera stessa, cioè il suo indirizzo. Le istruzioni di programma sono composte da due parti: un codice di operazione, che definisce l'operazione da eseguire, ed un indirizzo, che designa un elemento dei dati registrato in memoria. Si veda in proposito l'istruzione LDA, rappresentata nella figura 11 (LDA sta per Load Accumulator, cioè caricamento del accumulatore; il registro di lavoro principale è per tradizione

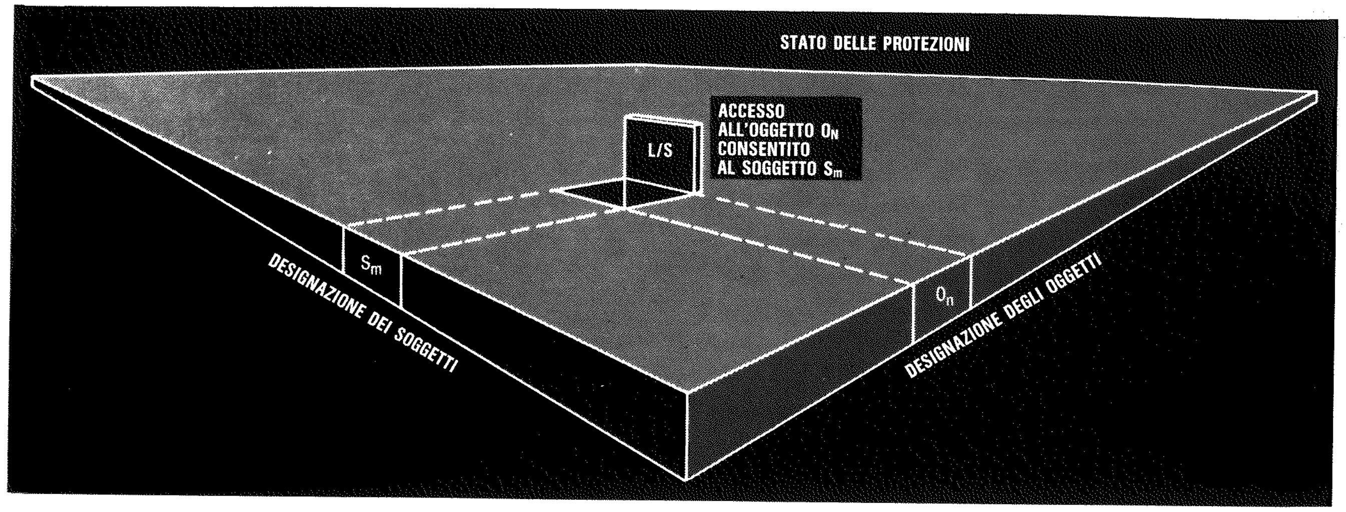


Fig. 9 - Il modello del sistema di elaborazione consiste (ad un livello di astrazione intermedio) di tre insiemi di informazione, ovvero spazi degli stati, chiamati: stato dei valori, stato delle protezioni e astrazioni sottostanti. Lo stato dei valori, che consiste di tutti gli oggetti del sistema che possono essere resi visibili ai soggetti, è al di fuori del reference monitor. Lo stato delle protezioni, qui rappresentato, si presenta come una matrice, in cui le righe sono indicizzate dai soggetti e le colonne dagli oggetti. Alla intersezione tra riga e colonna si trova il modo (od i modi) d'accesso autorizzato attualmente ad un soggetto per un oggetto.

chiamato accumulatore). Si può considerare l'istruzione come l'elemento "atomico" di un soggetto; essa è infatti l'entità minima indivisibile del soggetto considerato negli studi sulla sicurezza. Nel nostro esempio, l'istruzione LDA comanda una lettura di memoria per conto del suo soggetto.

Lo schema della figura 12 è un esempio di temporizzazione dei vincoli molto anticipata. Il dato che interessa viene assegnato all'indirizzo 1069 al momento della stesura del programma e tale assegnazione viene incorporata nel programma memorizzato nella macchina. Se si vuole spostare il nostro dato, occorre trovare nel programma tutti i riferimenti al suo indirizzo e modificarli, processo costoso e suscettibile di errori.

Una volta comprese le limitazioni di queste decisioni di temporizzazione dei vincoli, i progettisti di hardware inventarono la tecnica cosidetta di indirizzamento bi-dimensionale o segmentato. Una tabella intermedia traduce l'indirizzo contenuto in una istruzione in un indirizzo di memoria. L'indirizzo nell'istruzione è suddiviso in due parti: la prima seleziona una entrata della tabella intermedia, che fornisce l'indirizzo di memoria dell'elemento di base, ossia il più basso, nel segmento; la seconda parte dell'indirizzo seleziona una cella entro il segmento. La prima parte dell'indirizzo nell'istruzione è chiamata generalmente numero di segmento e la seconda parte scostamento. La figura 12 mostra questa organizzazione: l'indirizzo 1069 è stato diviso in due parti; il 10 seleziona un valore della tabella intermedia (4835) ed il 69 costituisce lo scostamento rispetto a questo valore di base. Perciò l'indirizzo di memoria della cella designata dall'indirizzo 1069 è 4904.

Se si vincola più tardi un dato ad un indirizzo di memoria, si ottengono diversi vantaggi. Anzichè assegnare l'indirizzo di memoria al momento della stesura del programma, noi possiamo assegnarlo in qualsiasi momento prima dell'esecuzione dell'istruzione, semplicemente introducendo il valore appropriato nella tabella intermedia. Inoltre questa riduce il costo di spostamento di un segmento dati nell'ambito della memoria. Se non cambiamo le posizioni dei dati rispetto all'inizio del segmento, non dobbiamo cambiare gli indirizzi forniti nelle istruzioni.

L'indirizzo del segmento può essere cambiato semplicemente modificando il decimo elemento della tabella intermedia. La maggiore flessibilità che così si ottiene ha un certo costo in termini di complessità e prestazioni dell'hardware, poichè il prelievo di dati comporta ora un prelievo intermedio ed una operazione aritmetica, ma tale costo aggiuntivo è accettabile.

L'indirizzamento segmentato fu sviluppato per rispondere alle esigenze dei primi sistemi multi-utente. Su questi sistemi, eventi non prevedibili come l'identificazione di un nuovo utente, rendevano necessario spostamenti frequenti di programmi e dati. Come abbiamo visto prima, i progettisti di macchine multiutente dovettero anche affrontare il problema di controllare la condivisione dei dati tra programmi diversi, e lo risolsero aggiungendo un campo nella tabella intermedia, in cui registrarono l'elenco dei modi di accesso consentiti. Ogni codice di operazione designa ora, oltre all'operazione stessa, anche il modo d'accesso che essa deve esercitare. Prima di eseguire ogni istruzione, l'hardware controlla se il modo d'accesso chiesto dall'istruzione stessa è tra i modi d'accesso permessi dall'entrata appropriata della tabella intermedia. Se il modo d'accesso chiesto non è consentito, l'operazione non può essere eseguita.

Il meccanismo descritto ha parecchie virtù che si conciliano con i requisiti che deve avere il reference

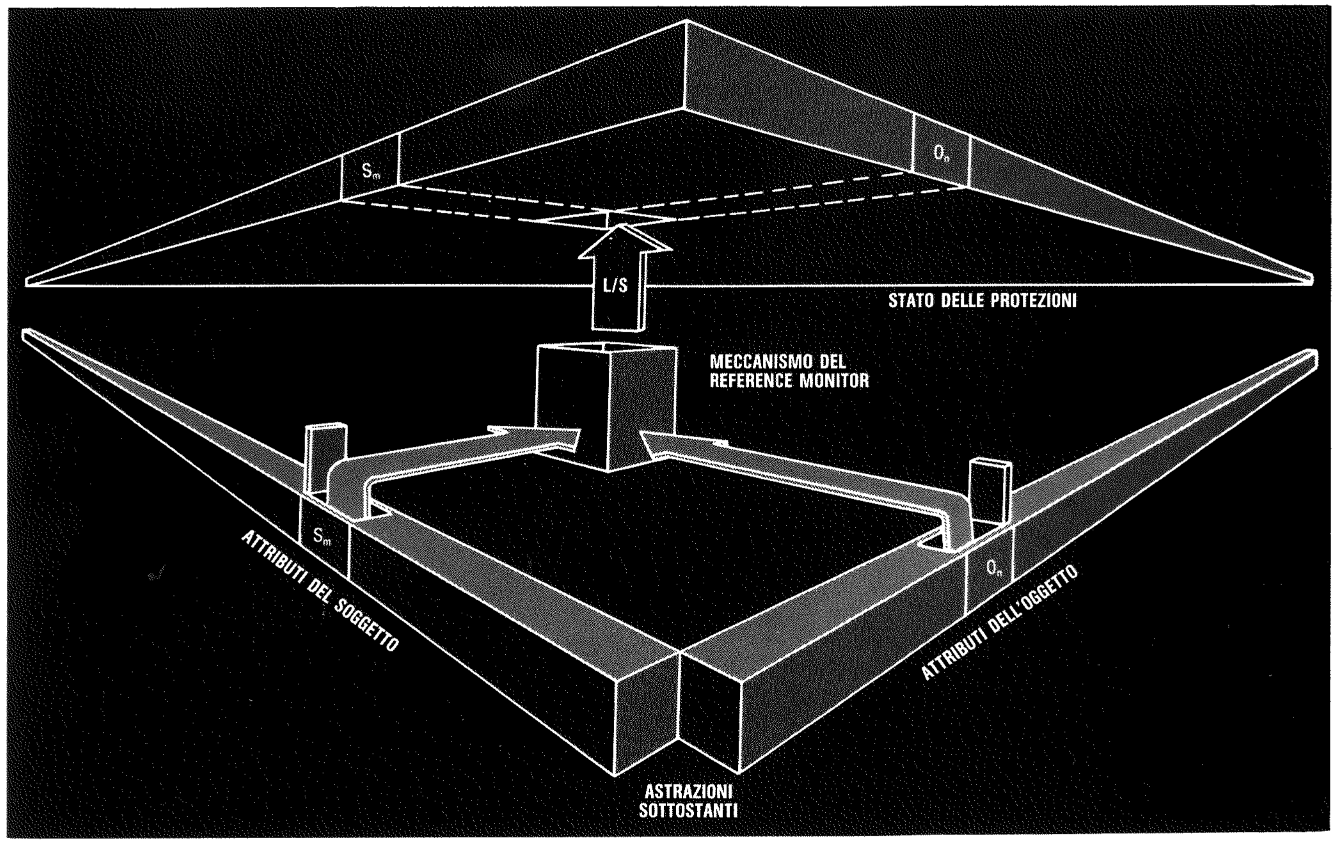


Fig. 10 - Le "astrazioni sottostanti" hanno la forma di tabelle che forniscono gli attributi di soggetti ed oggetti: tra gli attributi sono compresi quelli che interessano la sicurezza, come i livelli di sicurezza e gli acl. Le tabelle sono consultate da quella parte del reference monitor che ha il compito di inserire i valori nello stato delle protezioni.

monitor. È un meccanismo di hardware e quindi inviolabile. Non può essere aggirato poichè una istruzione non può ottenere un indirizzo senza passare attraverso la tabella intermedia ed incontrare il controllore degli accessi. Rimane ora per questo meccanismo il problema della prova.

Ritornando al nostro esame dei tracciati di mappa e dei livelli di astrazione, possiamo vedere che la tabella intermedia corrisponde ad una riga dello stato delle protezioni. La tabella è associata ad un soggetto (il programma esecutivo di cui fa parte l'istruzione che usa la tabella) e contiene la designazione di oggetti (indirizzi di segmento) e gli accessi consentiti. Questa mappa è rappresentata nella figura 14.

Abbiamo ora un meccanismo che garantisce che un valore dello stato delle protezioni, una volta stabilito, diventi rigorosamente operante. Ciò che ci manca ancora è un meccanismo che garantisca la sicurezza dei valori dello stato delle protezioni. Sappiamo che un utente "pirata" non può effettuare un accesso di lettura se non appare una r (read) nella posizione appropriata della tabella. Non abbiamo però alcuna certezza circa la provenienza di questa r, e in particolare non sappiamo se sia stato proprio l'utente "pirata" a metterla. Analogamente il soggetto non è identificato in modo esplicito da nessuna parte del meccanismo. Poichè le istruzioni si svolgono in modo anonimo dal punto di vista hardware, è possibile che l'attacco avvenga su un altro fronte: può darsi che un difetto di implementazione del software del sistema operativo permetta ad un soggetto con basso livello di sicurezza di "catturare" una tabella intermedia che era stata compilata per un soggetto con alto livello di sicurezza. Si tratta del cosidetto attacco con riutilizzo dei parametri (reused parameter attack), in cui i valori d'accesso sono i parametri riutilizzati. Tale attacco presuppone che avvenga il cambiamento di un valore critico (ad esempio un livello di sicurezza) nell'intervallo di tempo tra la concessione dell'accesso e l'attuazione dell'accesso stesso.

L'attacco verrebbe bloccato se i valori d'accesso fossero cambiati quando cambiassero gli attributi dei soggetti e degli oggetti.

I sistemi di sicurezza primitivi sono vulnerabili a questo fronte d'attacco, poichè il reference monitor è realizzato in software. L'indirizzamento segmentato è realizzato in hardware e ne consegue che, mentre le astrazioni sottostanti sono sotto forma di software, lo stato delle protezioni sta a cavallo dell'interfaccia tra software e hardware, cioè i suoi valori sono posti dal software ed utilizzati dall'hardware. Di conseguenza l'aggiornamento dello stato delle protezioni comporta un insieme complesso di interazioni tra le tabelle gestite dal software e l'hardware. In pratica la complessità di tali interazioni preclude ogni prova ine-

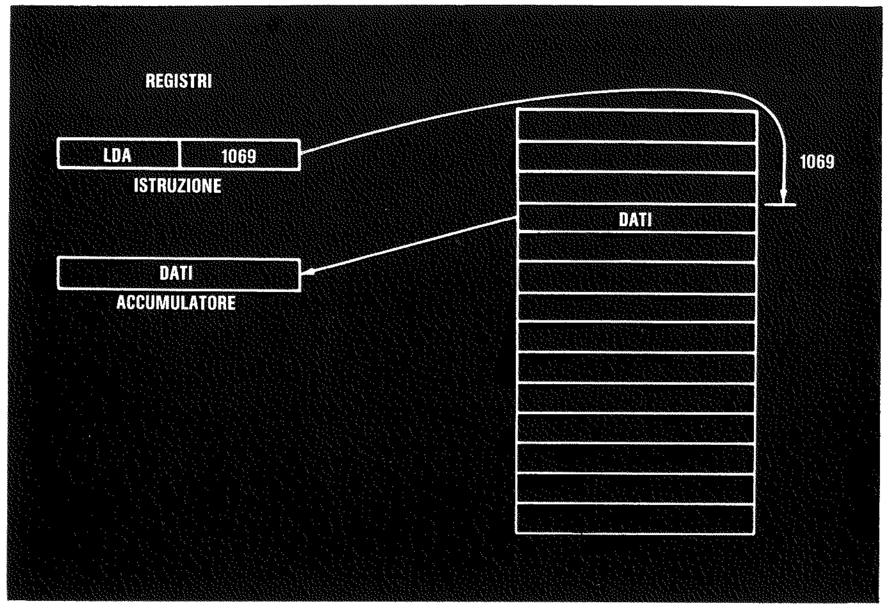


Fig. 11 - È qui rappresentato il meccanismo di indirizzamento semplice. L'istruzione ha in sè l'indirizzo dei dati di cui necessita.
Si compone di due parti: il codice che specifica l'operazione da eseguire (LDA o Load Accumulator = Caricamento accumulatore) e l'indirizzo del dato da caricare nell'accumulatore (1069). L'accumulatore è il principale registro di lavoro del soggetto; il dato che vi viene trasferito dall'istruzione sarà trattato dalle istruzioni successive. Questo modo di indirizzamento presenta il problema che ogni istruzione riferentesi ad un certo elemento di dati deve essere trovata e modificata quando il dato viene trasferito.

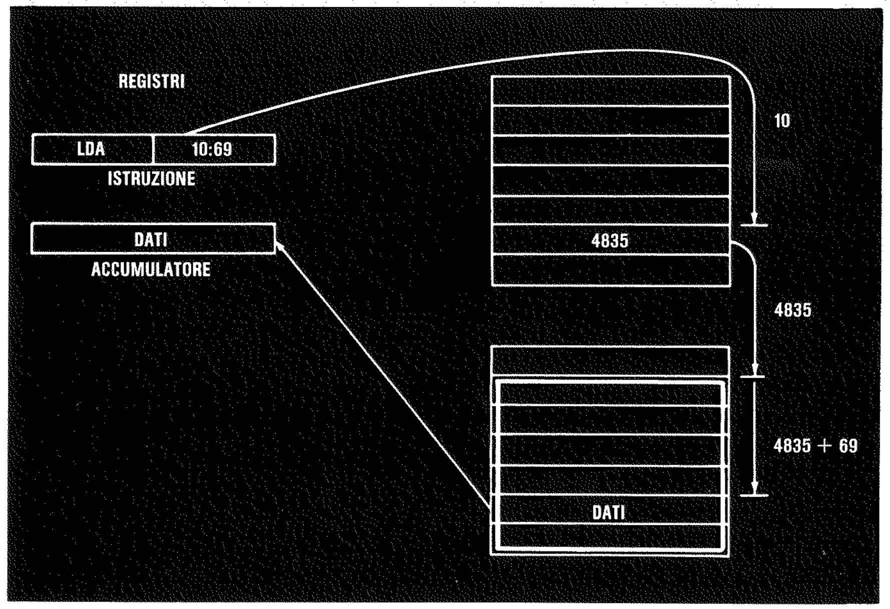


Fig. 12 - L'indirizzamento segmentato è una tecnica di indirizzamento che consente trasferimenti più agevoli dei dati. L'indirizzo di memoria del dato è memorizzato in una tabella intermedia, mentre l'indirizzo fornito dall'istruzione è un indirizzo indiretto.
La prima parte di questo indirizzo è un numero (10) che seleziona una entrata nella tabella intermedia.
L'entrata (4835) fornisce l'indirizzo di un blocco di memoria chiamato segmento. La seconda parte dell'istruzione è lo "scostamento", che specifica la posizione del dato che interessa. Se si cambia l'indirizzo di memoria del segmento, va cambiato il valore memorizzato nella decima posizione della tabella intermedia, ma le istruzioni rimangono inalterate.

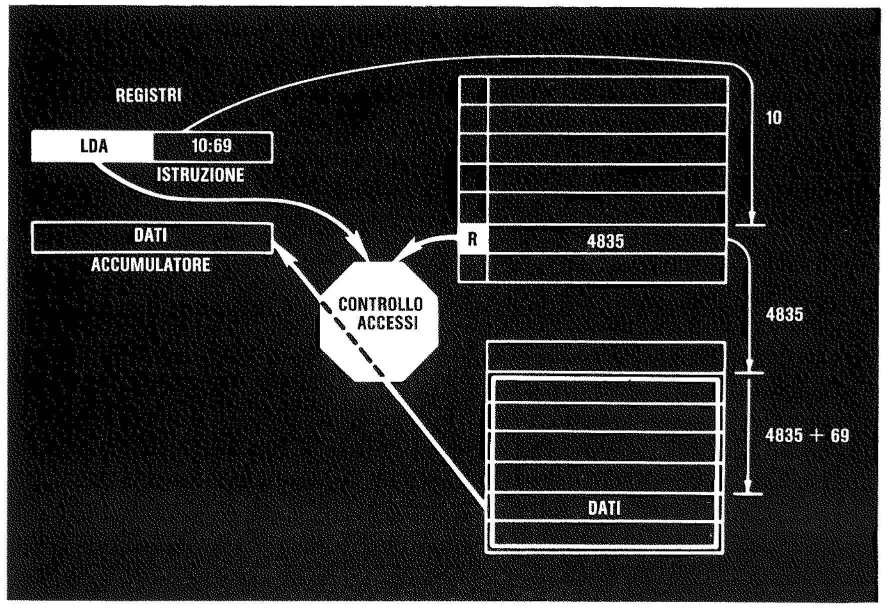


Fig. 13 - L'entrata aggiuntiva nella tabella intermedia che lista gli indirizzi di memoria dei segmenti, consente l'uso della tabella per verificare i diritti d'accesso dell'istruzione in esecuzione. Il codice di operazione dell'istruzione specifica (indirettamente) il modo di accesso chiesto dall'operazione. Il campo aggiuntivo della tabella intermedia specifica il modo d'accesso accordato al soggetto nel cui contesto viene eseguita l'istruzione. In questo esempio la tabella intermedia autorizza il modo d'accesso chiesto (lettura) e pertanto si autorizza l'esecuzione dell'istruzione.

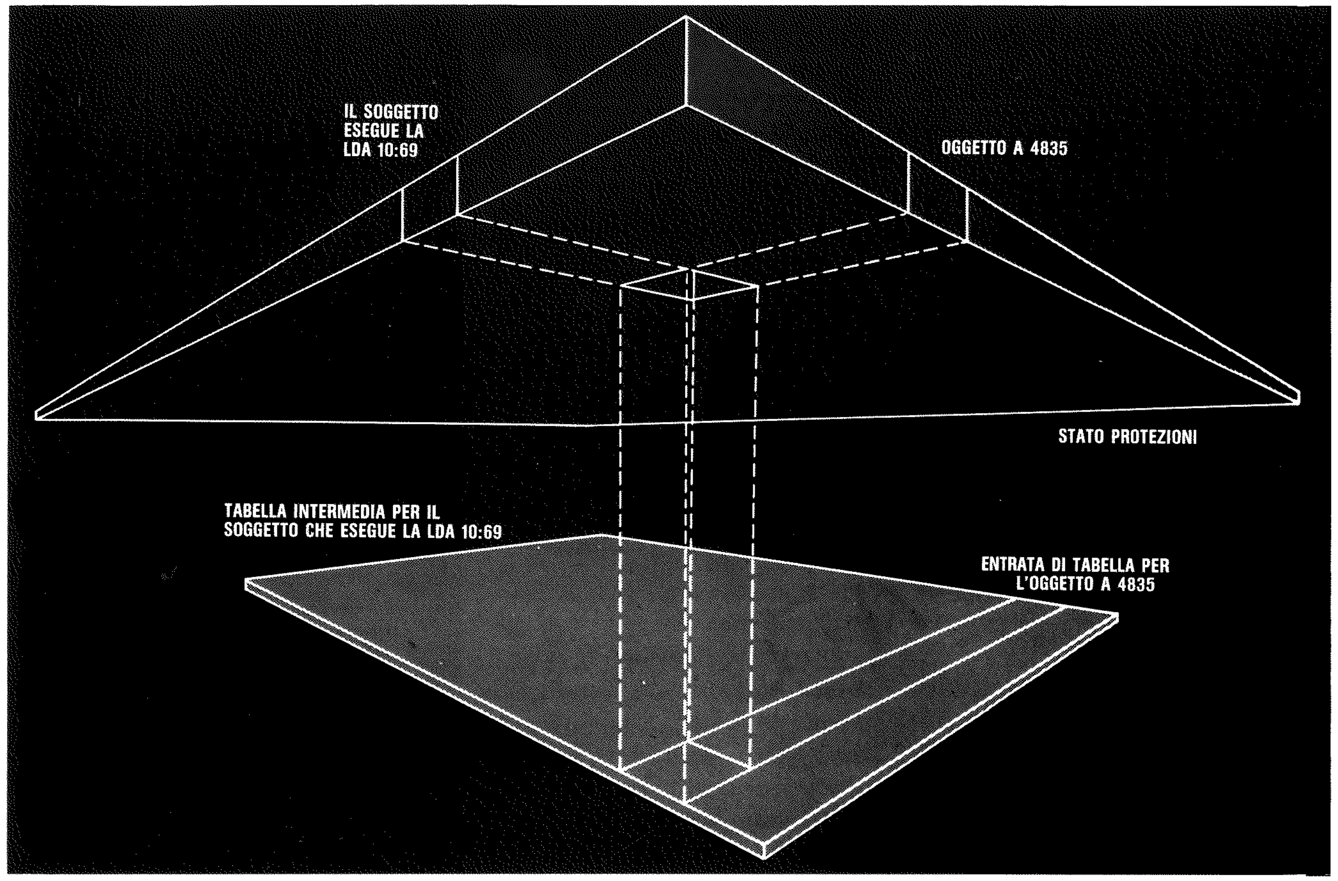


Fig. 14 - È qui rappresentata la correlazione tra la tabella intermedia in una macchina con indirizzamento segmentato e la matrice dello stato delle protezioni. La tabella corrisponde ad una riga della matrice. Essa è associata ad un soggetto e contiene designazioni di oggetti (indirizzi di segmenti) ed i modi di accesso consentiti. La tabella intermedia garantisce che l'accesso sarà accordato in armonia con i valori dello stato delle protezioni. È tuttavia difficile dimostrare che i valori della tabella riflettono gli attributi dei soggetti ed oggetti che essi correlano, cioè che lo stato delle protezioni riflette le astrazioni sottostanti nel modo rappresentato nella figura 10.

quivocabile che il reference monitor possa realizzare la sicurezza.
L'implementazione del reference monitor in software presenta anche lo svantaggio di dover condividere le risorse hardware con programmi infidi e potenzialmente intrusi. Viene a ridursi così il coefficiente di sicurezza dell'inviolabilità del reference monitor ed aumenta la probabilità che programmi "pirata" utilizzino lo stesso reference monitor per segnalazioni in contrasto con la normativa obbligatoria di sicurezza. Tali canali segnaletici, chiamati "canali nascosti", producono l'effetto "pounding on the walls" (colpi sul muro, ovvero segnalazioni in codice), così denominato da Butler W. Lampson della Xerox PARC. Un programma Cavallo di Troia può inviare un messaggio ad un programma alleato con operazioni che impegnano il reference monitor per un tempo più o meno lungo, dando per convenzione il significato "1" ad un tempo lungo ed il significato "0" ad un tempo breve. L'alleato decodifica il messaggio verificando ad intervalli regolari se il reference monitor sta utilizzando l'hardware condiviso da tutti i programmi. In tal modo è possibile degradare l'informazione anche se il reference monitor impedisce ogni tentativo di trascriverla direttamente.

Il nostro approccio a questi problemi è stato quello di trasferire completamente in hardware il reference monitor, lo stato delle protezioni e le astrazioni sottostanti. In questo modo siamo riusciti a semplificare radicalmente le mappe tra i livelli di astrazione più alti e quelli più bassi, nella nostra struttura di prova. Siamo anche riusciti a sostenere in modo più concreto e convincente che il reference monitor non può essere aggirato e che è inviolabile. Nella sezione seguente descriveremo l'architettura hardware che ne risulta.

7. Architettura del Secure Ada Target

L'hardware del Secure Ada Target implementa i diritti d'accesso tramite indirizzamento segmentato e campi di modo d'accesso. Sotto questo aspetto assomiglia ad alcune macchine contemporanee. La differenza sta nelle fasi con cui si modifica la tabella intermedia. Per consentire modifiche alla tabella in modo che essa possa essere completamente controllata dall'hardware, l'architettura del Secure Ada Target introduce una nuova forma di informazione memorizzata.

Le architetture convenzionali affidano al software l'interpretazione degli elementi di informazione memorizzati.

L'hardware non distingue una stringa di 1 e di 0, che rappresenta in codice un valore numerico, da un'altra stringa in cui è codificato ad esempio il nome di un flusso dati. Il no-

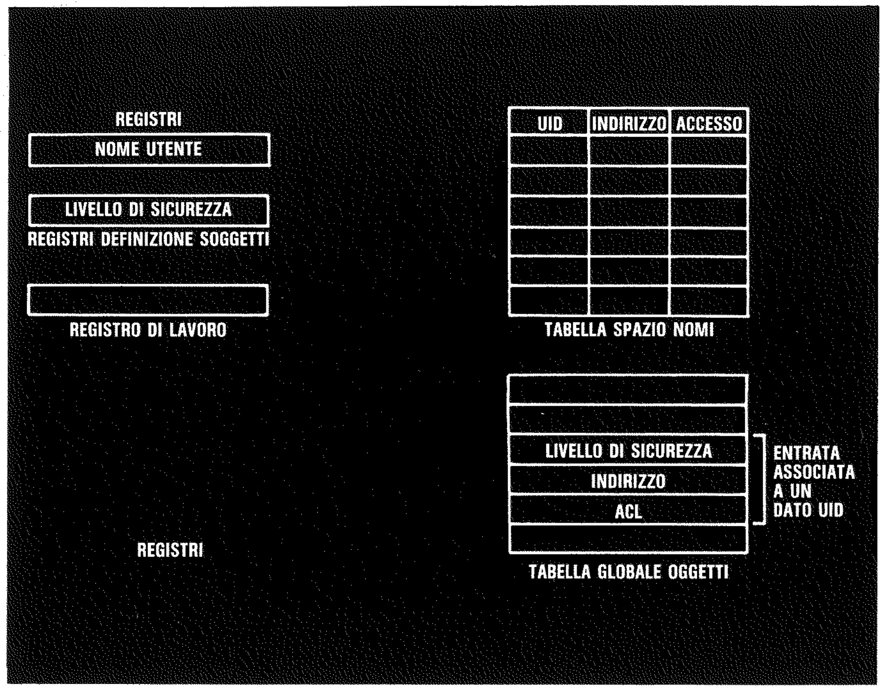


Fig. 15 - L'estensione della struttura dell'indirizzamento segmentato garantisce che soggetti ed oggetti siano identificati in modo esplicito e che lo stato delle protezioni rifletta correttamente i loro attributi. Un soggetto è identificato da due registri che gli assegnano un nome, composto dall'utente per il quale sta operando e dal livello di sicurezza a cui sta operando. Gli oggetti sono identificati dagli UID (Universal Identifiers = Identificatori Universali). Si tratta di valori contenuti nei "tagged objects" e che identificano elementi di memoria chiamati contenitori. Gli attributi dei contenitori sono elencati nella "Tabella Globale degli Oggetti" (Global Object Table, GOT), che è indirizzata dall'UID. Gli attributi comprendono il livello di sicurezza, l'acl e la posizione attuale di memoria del contenitore. Gli oggetti accessibili ad un soggetto sono identificati da UID memorizzati in una tabella intermedia appartenente al soggetto stesso, chiamata "Tabella Spazio dei Nomi". Questa strutturazione permette di costruire un meccanismo di sicurezza che verifica gli attributi sia di un soggetto che di un oggetto prima di modificare una entrata nella tabella intermedia.

stro hardware (nella forma semplificata che consideriamo in questa sede) riconosce due forme di dati memorizzati, che chiamiamo "contenitori" (containers) e "oggetti col cartellino" (tagged objects).

Il contenitore è l'unità di memorizzazione secondo il significato tradizionale del termine, cioè una sequenza di elementi di dati che il software interpreterà nel modo ritenuto opportuno. I nostri contenitori corrispondono quindi ai segmenti di cui si è parlato a proposito dell'indirizzamento segmentato.

Il tagged object è il nome di un contenitore riconosciuto dall'hardware. La caratteristica che lo distingue è un bit in più, o tag (cartellino), associato all'oggetto memorizzato, da cui deriva il nome "tagged object". L'hardware vieta rigorosamente l'utilizzo di tagged objects da parte del software. Una volta creato, un tagged object non può essere esaminato nè modificato. Può tuttavia essere immagazzinato liberamente in un contenitore. I soggetti comunicano con il reference monitor principalmente tramite tagged-object.

Nella versione attuale del progetto, un tagged-object ha una lunghezza di 128 bit, che contengono un valore protetto, chiamato UID (Universal Identifier o Identificatore Universale), cioè il nome del contenitore.

Ogni contenitore ha un diverso UID ed è perciò identificato in modo univoco. L'hardware crea un tagged object, che denomina un contenitore, al momento dell'allocazione in memoria del contenitore stesso.

Si consideri per un momento la relazione tra questa rappresentazione del sistema a livello hardware e la rappresentazione al livello di astrazione immediatamente superiore. Abbiamo or ora definito oggetti (come contenitori) e fornito un mezzo attendibile per designarli. (Oggetti-cartellino "opachi", che non possono essere falsificati). È necessario ora trovare il modo di rappresentare le astrazioni sottostanti. Si tratta cioè di rappresentare gli attributi dei contenitori, come le loro dimensioni, posizioni attuali in memoria, livelli di sicurezza ed i loro acl.

Il reference monitor del Secure Ada Target gestisce le astrazioni sottostanti in un flusso privato indicizzato dagli UID, chiamato "Tabella Globale Oggetti" (Global Object Table, ovvero GOT), poichè vi dice di che cosa disponete globalmente. La GOT può essere immaginata come un grande flusso contenente gli attributi di tutti i contenitori del sistema. L'entrata in GOT per un contenitore (nella forma semplificata di architettura che stiamo trattando) specifica il livello di sicurezza, l'acl e la posizione attuale del contenitore.

Abbiamo così caratterizzato l'architettura dello stato dei valori: un insieme di contenitori, ognuno identificato in modo univoco ed associato ad attributi fisici (utilizzati nella normale elaborazione), ad un livello di sicurezza e ad un acl (utilizzato dal reference monitor).

Dobbiamo affrontare ora il problema della rappresentazione di un soggetto a livello hardware. Essa deve supportare un elaborazione efficiente, prestarsi ad un immediato tracciamento di mappe verso livelli di astrazione superiori nella nostra struttura di prova e deve garantire che le limitazioni stabilite dallo stato delle protezioni vengano imposte in modo uniforme sulle operazioni eseguite sullo stato dei valori.

Da un punto di vista generale, si prende la nota struttura dell'indirizzamento segmentato e la si estende. Il reference monitor dispone del nome dell'utente per conto del quale ogni istruzione viene eseguita (in modo che possano essere prese decisioni su accessi discrezionali), del livello di sicurezza dove il soggetto sta operando (in modo che possano essere prese decisioni di normativa obbligatoria di accesso) e delle parti pertinenti dello stato delle protezioni (in modo che le decisioni, una volta prese, possano essere attuate). L'attuazione delle norme di sicurezza viene così estesa in giù fino al li-

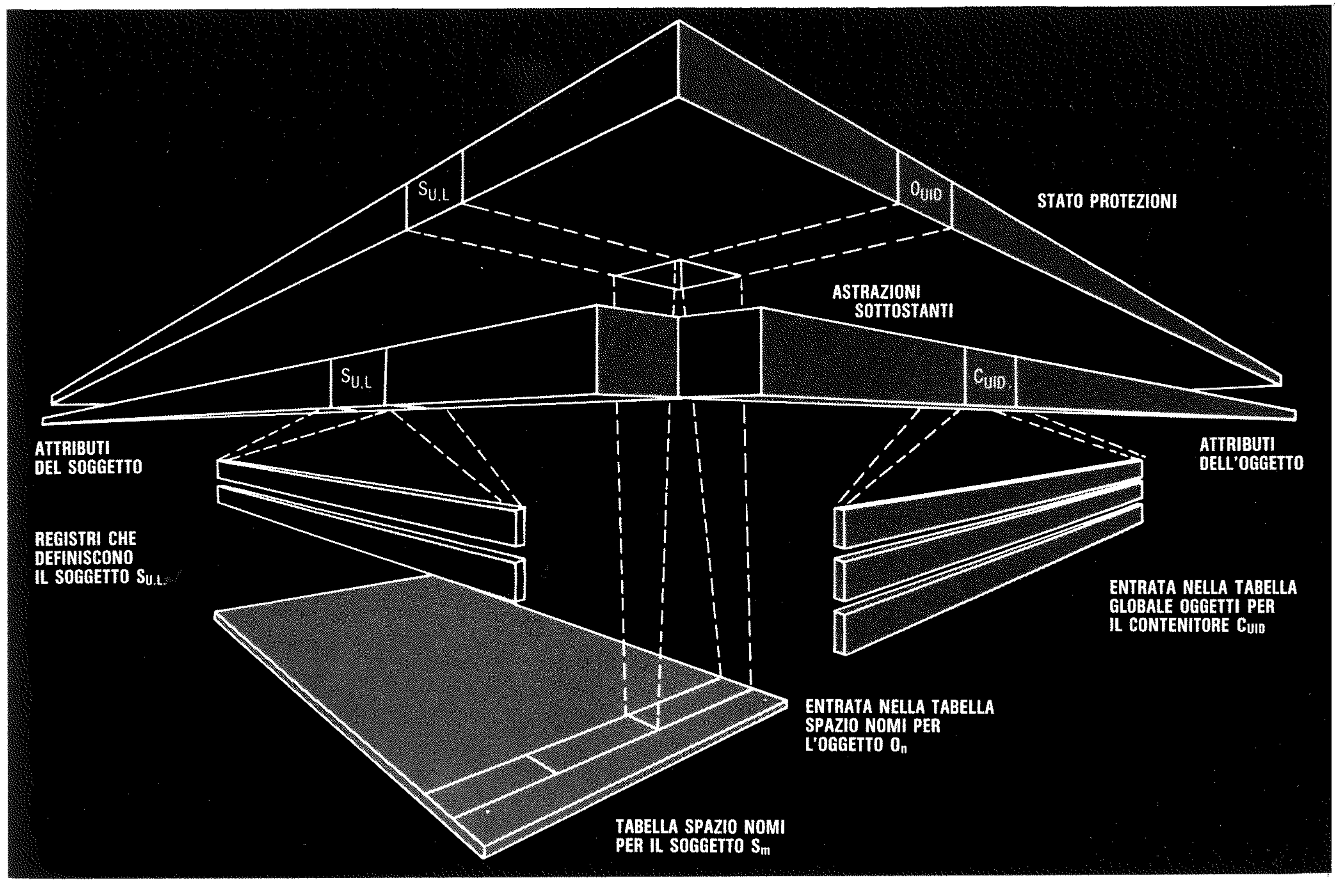


Fig. 16 - È qui mostrata la mappa o correlazione tra gli spazi di stato del reference monitor e gli elementi dell'architettura del "Secure Ada Target" (rappresentata nella figura 15).

La evidenza della dimostrabilità del "Secure Ada Target" è dovuta principalmente alla estrema semplicità delle mappe tra i vari livelli del progetto.

vello della singola istruzione di macchina.

Il primo elemento della rappresentazione di un soggetto consiste di due registri hardware che designano il soggetto stesso e ne definiscono gli attributi. Essi contengono un nome d'utente ed un livello di sicurezza. La tabella intermedia associata al soggetto viene estesa fino a contenere, per ogni entrata, l'UID del contenitore cui l'entrata stessa fa riferimento. La tabella estesa è chiamata Tabella dello Spazio dei Nomi (Name Space Table, ovvero NST), poichè definisce l'insieme di oggetti che possono essere nominati (richiamati) dal soggetto. La rappresentazione di un soggetto è completata dai registri di lavoro, utilizzati dagli elementi unitari indivisibili ("atomici") del soggetto, ossia le istruzioni. I registri si possono concepire come contenenti le informazioni private del soggetto.

La mappa tracciata tra questa rappresentazione a livello hardware e la rappresentazione a matrice dello stato delle protezioni, appare nella figura 16. Nel Secure Ada Target, ogni soggetto viene designato da un nome composito nella forma "utente, livello". Le diverse parti del suo nome sono i valori degli attributi forniti dai registri di definizione del soggetto. Si ha così una mappa tra questi registri ed un indice sull'asse dei soggetti della matrice dello stato sulle protezioni, nonchè tra questi registri e la tabella delle astrazioni che fornisce gli attributi del soggetto. Si stabilisce d'altra parte una mappa tra la NST ed una riga della matrice stato delle protezioni, benchè questa mappa non appai in figura.

Gli oggetti presentano maggiori complessità. Essi non possono avere un nome composito costituito dai loro attributi, poichè questi sono in numero eccessivo, ma sono designati invece con l'UID. L'UID si ricollega (con mappa) ad un indice sull'asse degli oggetti della matrice stato delle protezioni. L'entrata di GOT per l'oggetto si ricollega (con mappa) con la tabella delle astrazioni che fornisce gli attributi degli oggetti. (Occorre osservare che l'entrata di GOT include informazioni, quali l'indirizzo attuale dell'oggetto, che non entrano nella mappa; questa è una situazione tipica della relazione tra due livelli di astrazione). Infine, una entrata in NST per l'oggetto si ricollega (con mappa) con l'intersezione tra una riga ed una colonna nella matrice dello stato delle protezioni.

La NST conduce le operazioni eseguite sullo stato dei valori sotto il controllo dei valori posti nello stato delle protezioni, esattamente come fa la più rozza e meno sicura tabella intermedia utilizzata nell'indirizzamento segmentato. La parte indirizzo di una istruzione viene interpretata come un indice di NST ed uno scostamento. L'indice di NST seleziona una entrata nella NST appartenente al soggetto di cui fa parte

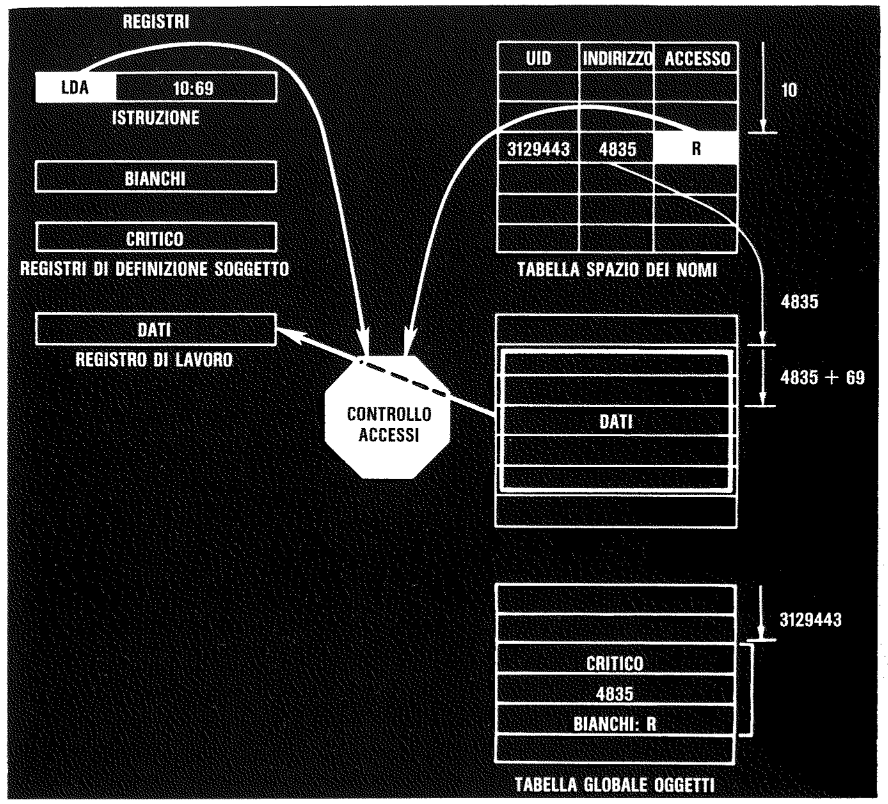


Fig. 17 - Le comuni istruzioni sono sottoposte al controllo di accesso schematizzato in figura. È rappresentato un contenitore con UID = 3129443 e posizione attuale di memoria = 4835, al quale accede un soggetto operante per conto dell'utente Bianchi a livello di sicurezza "critico". Nella verifica accessi di questo esempio, la Tabella Spazio dei Nomi svolge il ruolo della tabella intermedia rappresentata nella figura 13. In entrambi i casi, l'indirizzo specificato nella istruzione serve da indice della Tabella.

l'istruzione, e l'hardware genera l'indirizzo del item di dati utilizzando l'entrata nella NST (che fornisce l'indirizzo base del contenitore) e la parte scostamento della istruzione. Contemporaneamente il valore del modo d'accesso nella entrata di NST viene controllato per confronto con il modo d'accesso chiesto dalla istruzione. Se quest'ultimo è consentito, allora l'hardware autorizza l'accesso alla memoria; in caso contrario, l'istruzione si blocca e si ha una segnalazione d'allarme. Il procedimento sopra descritto è rappresentato nella figura 17.

L'impiego di una tecnica consolidata per la verifica dei diritti d'accesso di comuni istruzioni mette a nostra disposizione un metodo efficiente e poco costoso in quanto possiamo avvalerci della tecnologia hardware e software che è andata sviluppandosi negli ultimi 20 anni. Come mostra lo schema, la verifica degli accessi non fa uso, tuttavia, delle estensioni da noi incorporate nella rappresentazione a livello hardware dello stato delle protezioni per fissare le identità dei soggetti e degli oggetti.

Tali estensioni sono utilizzate da una seconda classe di operazioni, eseguite anche dal reference monitor, di cui la più basilare, che ci servirà per illustrare questa classe, è l'istruzione di caricamento NST (Load NST, ovvero LNST). La sua esecuzione è preceduta normalmente dall'esecuzione dell'istruzione di Creazione Entrata GOT (Create GOT Entry, ovvero CGEN).

Il soggetto esegue l'operazione CGEN per creare un oggetto. Si supponga che l'utente Bianchi, operando a livello di sicurezza "critica", desideri creare un oggetto in memoria. La CGEN crea un nuovo oggetto e gli alloca spazio di memoria.

La nuova entrata della GOT ha assegnato un livello di sicurezza congruente con il livello di sicurezza del soggetto di Bianchi. L'acl del nuovo oggetto è predisposta per concedere accesso a Bianchi e non concederlo ad altri utenti. Al termine dell'esecuzione di CGEN, il reference monitor restituisce un tagged object alla posizione di memoria che il soggetto aveva specificato nel richiamare l'operazione CGEN.

Alla fine dell'operazione CGEN, si avrà una nuova entrata nella GOT ed un nuovo tagged-object, ma il soggetto in funzione non può ancora accedere al nuovo oggetto. Per poter far ciò, deve prima eseguire l'operazione LNST che aggiunge il nuovo oggetto al suo spazio dei nomi. Il reference monitor, mentre esegue la LNST, esamina lo spazio degli stati da noi chiamato "astrazioni sottostanti" per assicurarsi che i diritti di accesso che trasferisce nella tabella Spazio Nomi attuino le norme di sicurezza obbligatoria e discrezionale.

La figura 18 illustra il funzionamento dell'istruzione LNST. L'utente Bianchi, operante ancora a livello di sicurezza "critico", richiama il comando LNST. Il monitor esige che nel richiamo siano inclusi il tagged-object che designa il contenitore da aggiungere allo spazio nomi e l'indice della NST (ovvero numero di segmento) con cui il soggetto farà in seguito riferimento al contenitore.

Non appena il tagged-object si trova nel reference monitor, il valore UID nel suo interno (312 9443 nel nostro esempio) diventa visibile ed utilizzabile per localizzare una entrata della GOT. Gli attributi del contenitore elencati nella GOT consistono di un livello di sicurezza (critico), di un indirizzo (4385) e di una acl (con

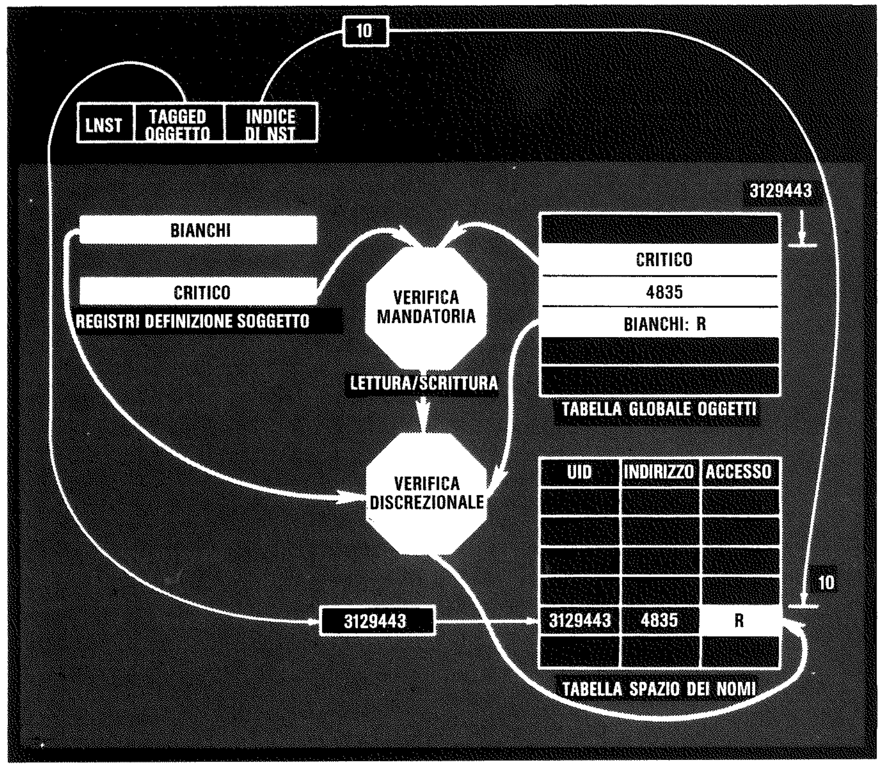


Fig. 18 - Le estensioni della struttura di indirizzamento entrano in gioco quando un soggetto aggiunge un contenitore alla sua Tabella Spazio dei Nomi, tramite l'istruzione LNST. Questa localizza un Tagged-Object che contiene un UID (Universal Identifier). L'UID diventa visibile nel reference monitor e serve da indice nella Tabella Globale Oggetti. Il reference monitor reperisce gli attributi dell'oggetto memorizzati nella Tabella Globale Oggetti, cioè il livello di sicurezza del contenitore, il suo indirizzo ed il suo acl. Il reference monitor confronta il livello di sicurezza del contenitore con quello del soggetto, per determinare se l'accesso richiesto si concilia con la normativa obbligatoria di sicurezza. Confronta il nome dell'utente con l'acl del contenitore per determinare se l'accesso richiesto si concilia con la normativa discrezionale di sicurezza. Il ruolo d'accesso autorizzato da ambedue i controlli viene introdotto nella Tabella Spazio dei Nomi, assieme all'indirizzo del contenitore. Poichè i valori della tabella Spazio dei Nomi derivano da attributi registrati in hardware sotto il controllo del reference monitor, essi sono più attendibili dei valori della tabella intermedia che viene impiegata per il controllo degli accessi nei sistemi con indirizzamento segmentato.

una sola entrata che limita l'accesso dell'utente Bianchi al solo modo lettura).

Come prima operazione, il reference monitor determina i diritti d'accesso massimi che il soggetto (Bianchi, critico) potrebbe esercitare sull'oggetto 312 9443. L'operazione consiste di due fasi, cioè le verifiche a fronte rispettivamente della normativa obbligatoria e discrezionale. La prima verifica comporta il confronto tra il livello di sicurezza del soggetto (contenuto nei registri di definizione soggetto) ed il livello di sicurezza dell'oggetto (contenuto nella GOT). Nel nostro esempio entrambi i livelli sono "critici", e si ottiene così un risultato intermedio, del massimo accesso possibile (lettura/scrittura). La verifica a fronte della normativa discrezionale comporta il confronto del nome d'utente, contenuto nei registri di definizione soggetto e per il quale sta operando il soggetto, con l'acl dell'oggetto (nella GOT). Nel nostro esempio, il nome d'utente corrisponde ad una entrata dell'acl, che limita l'accesso alla sola lettura. Il modo d'accesso che ne consegue (lettura) viene trasferito nel campo appropriato della NST. Le rimanenti operazioni dell'istruzione sono semplici trasferimenti di dati dalla GOT e dal tagged-subject, come mostrato dallo schema.

Dalla mappa rappresentata in fig. 16 risulta la prova evidente che il comando LNST tutela la sicurezza. La Proprietà di Sicurezza Semplice e la Proprietà * sono state definite in principio funzioni esprimenti relazioni appropriate tra valori dello stato protezioni ed i valori nelle matrici sottostanti. Funzioni analoghe governano le relazioni tra le entrate di modo d'accesso nella NST ed i valori nei registri di definizione soggetto e nella GTO. Poichè gli enunciati di mappa della Proprietà di Sicurezza Semplice e della Proprieta* sono incorporati nell'algoritmo usato dalla istruzione LNST, la prova che l'istruzione tutela la sicurezza è una vera e propria tautologia.

Esistono, naturalmente, molti altri problemi di ingegneria nella progettazione del Secure Ada Target, relativi ad esempio alla collocazione dell'hardware del reference monitor, alle operazioni che esso esegue viste in dettaglio e alla sua interazione con il software. Il lettore interessato alla trattazione di questi problemi può consultare la monografia "Secure Ada Target: Issues, System Design and Verification", pubblicata dal "Honeywell Secure Computing Technology Center".

8. Verifica del Secure Ada Target

Vorremmo concludere con la dimostrazione informale della sicurezza del sistema che era l'obiettivo del nostro progetto:

1. Tutti i dati visibili sono raccolti in un contenitore, ciascuno dei quali ha un livello di sicurezza.
2. Per accedere ad un elemento di informazione, una istruzione (soggetto) deve procurarsi l'indirizzo del contenitore (oggetto) contenente quel dato.
3. Nel procurarsi tale indirizzo, il soggetto non può evitare l'ostacolo dell'hardware del reference monitor, che verifica il modo di accesso chiesto a fronte dei modi di accesso permessi.
4. Per poter procurarsi un indirizzo, il soggetto deve aver prima aggiunto l'oggetto al suo spazio dei nomi.
5. Nell'aggiungere un oggetto al suo spazio dei nomi, il soggetto non può evitare l'ostacolo dell'hardware del reference monitor, che determina il modo d'accesso massimo consentito,

sulla base delle norme di sicurezza, obbligatorie e discrezionali.

6. Pertanto ogni e qualsiasi istruzione eseguita sulla macchina Secure Ada Target, è disciplinata dalle norme di sicurezza, obbligatorie e discrezionali.

9. Riconoscimenti

Il nostro lavoro è stato supportato dai contratti MDA 904-82-C- 044 e MDA 904-84-C- 6011 del governo USA. Abbiamo attinto a piene mani dal lavoro dei nostri predecessori, in particolare dai progetti Multics e SCOMP della Honeywell e dal lavoro della SRI International sul "Sistema Operativo Dimostrabilmente Sicuro" (Provably Secure Operating System). Il nostro lavoro ha beneficiato dell'attento esame di revisori capaci ed impegnati; siamo particolarmente grati a questo proposito a Morrie Gasser della Mitre Corporation. Ci siamo anche giovati dei colloqui avuti con il personale del Computer Security Center del Dipartimento della Difesa degli U.S.

10. Bibliografia

[1] J.P. ANDERSON, *"Computer Security Technology Planning Study,"* ESDTR-73-51, FSD/AFSC, Hanson AFB, Massachusetts, October 1972.

[2] D.E. BELL and L.J. LA PADULA, *"Secure Computer Systems: Unified Exposition and Multics Interpretations,"* Technical Report MTR-2997, Mitre Corporation, Bedford, Massachusetts, July 1975.

[3] H.K. BERG, W.E. BOEBERT, W.R. FRANTA and T,G, MOHER, *Formal Methods of Program Verification and Specification,* Prentice-Hall Inc., 1982.

[4] W.E. BOEBERT, R.Y. KAIN, W.D. YOUNG and S.A. HANSOHN, "Secure Ada Target: Issues, Systems Design, and Verification," *Proceedings of the 1985 Symposium on Security and Privacy,* IEEE Computer Society, April, 1985.

Sistemi esperti e lavoro d'ufficio*

MAURO LANGFELDER

Consulente di Organizzazione e Informatica

GIULIO OCCHINI

Honeywell Information Systems Italia

1. Introduzione

Nella sua immaginazione l'uomo moderno vede l'elaboratore come lo strumento che riuscirà a fargli dominare la crescente complessità del mondo che lo circonda o, alternativamente, come il tiranno che lo renderà schiavo della sua razionalità assoluta e, quindi, inumana.

La storia delle applicazioni informatiche fa pensare che l'uomo sia in grado con la sua intelligenza e cultura di rendere "umano" l'elaboratore mentre è improbabile che quest'ultimo possa "automatizzare" l'uomo e i suoi comportamenti.

Ciò premesso, uno degli aspetti della informatizzazione sociale che maggiormente, e a ragione, preoccupa, è la cosiddetta "data pollution", vale a dire la proliferazione incontrollata e incontrollabile di informazioni che bersagliano il cittadino sia nella vita privata che in quella lavorativa.

La "data pollution" trova certamente alimento nella diffusione degli elaboratori e quindi nella crescita esponenziale della capacità di manipolare automaticamente i dati. Ma non sta in questo la causa ultima del fenomeno.

Di fatto è la complessità delle situazioni economiche e sociali che richiede quantità sempre maggiori di informazioni per governarle e poter reagire correttamente agli eventi imprevisti (e imprevedibili).

Ecco quindi che ciò di cui l'uomo ha idealmente bisogno nello svolgimento del suo lavoro, non è tanto il puro strumento di automazione nel trattamento dei dati, quanto piuttosto il sistema che sia in grado di elaborare per lui, sulla base dei dati conosciuti, varie alternative di azione, ciascuna con i suoi vantaggi e svantaggi.

Più che di una macchina, in altre parole, sarebbe auspicabile disporre di un sistema che, dotato della capacità, caratteristica dei moderni elaboratori, di operare a velocità elettronica su immense quantità di dati, sia però anche abbastanza "competente" nella materia da ricondurre autonomamente la complessità delle situazioni a delle sintesi significative, sulle quali poi la mente umana possa esercitare la sua peculiare capacità di discriminazione.

Visto da un'altra angolazione, si potrebbe dire che l'impiego interattivo dell'informatica (specie quello caratteristico dei sistemi di ufficio) ha generato il desiderio di fare dell'elaboratore non più uno "schiavo stupido" ma piuttosto un partner, un interlocutore "esperto", che pur lasciando all'uomo ciò che gli è proprio, lo affianchi nel lavoro riducendo la "data pollution" che lo affligge, senza per altro trascurare i molteplici aspetti della realtà di cui si deve tener conto per decidere.

L'interlocutore di cui si è parlato è precisamente quello che, in terminologia informatica, viene chiamato "sistema esperto" e che dovrebbe far compiere all'elaboratore un vero e proprio salto di qualità a partire dalla fine di questa decade.

Anzichè macchina per calcolare, per manipolare dati o per generare informazioni, il "sistema esperto" dovrebbe in sostanza essere capace di elaborare e produrre "competenza" (fig. 1).

2. Elaboratori e "competenza"

Ma che cosa è la "competenza"?

È solo riflettendo su questa domanda che ci si può rendere conto di come un elaboratore possa, in prospettiva, diventare "esperto" in una specifica area del lavoro di ufficio.

Nella letteratura sull'argomento si è più volte affermato che, nella attività di trattamento informazioni, una parte rilevante non è di tipo strutturabile e cioè riconducibile a procedure formalizzate.

L'attività del manager, o dello specialista, così come quella del generico professionista (medico, avvocato, ingegnere) non consta, se non in piccola parte, di formule o calcoli,

(*) Il materiale di questo articolo è tratto dal volume "Voi, l'automazione e l'ufficio", degli stessi autori, edito dal Gruppo Editoriale Jackson (1985).

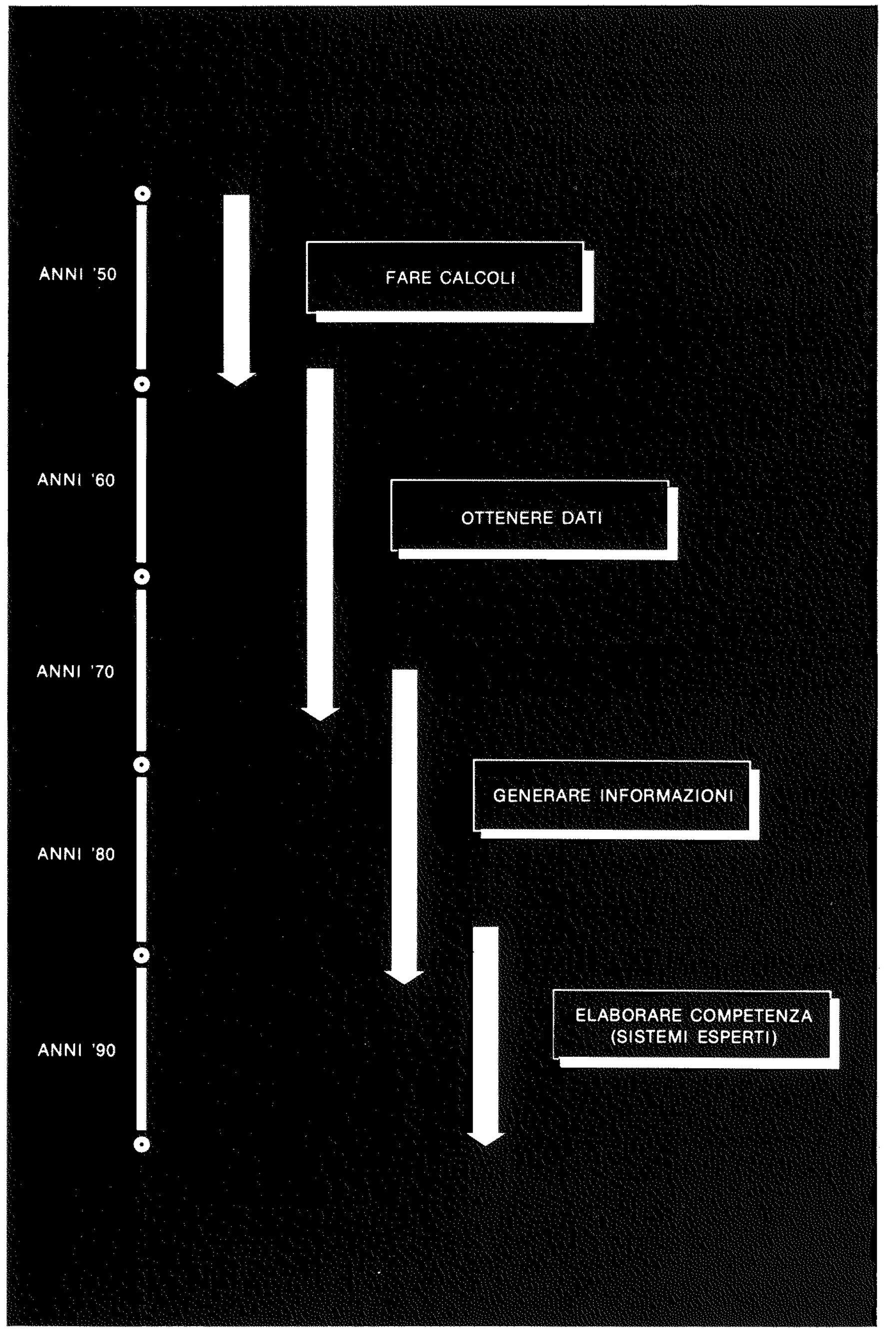


Fig. 1 - Dal "calcolatore" al "sistema esperto".

ma è piuttosto basata su una molteplicità di processi deduttivi di carattere non formalizzabile.

Tutti questi ruoli sono "esperti" nel loro campo, in quanto possiedono un ampio bagaglio di *competenze,* costituite sia da fatti e da conoscenze di base che da regole empiriche, di tipo euristico, nate dall'accumulo della esperienza quotidiana di lavoro nel particolare settore.

Questa competenza è esprimibile sotto forma di *regole euristiche,* del tipo:

"se esiste la premessa (A) allora posso trarre la conclusione (B)"

Esperto è colui che ha la capacità di collegare tra loro le regole opportune, attraverso un meccanismo di *inferenza,* ossia di deduzione logica, per arrivare ad una conclusione:

"**Se** esiste la premessa (A) **e** la premessa (B) **o** la premessa (C), **ma non** la (D) **allora** è probabile la conclusione (E)".

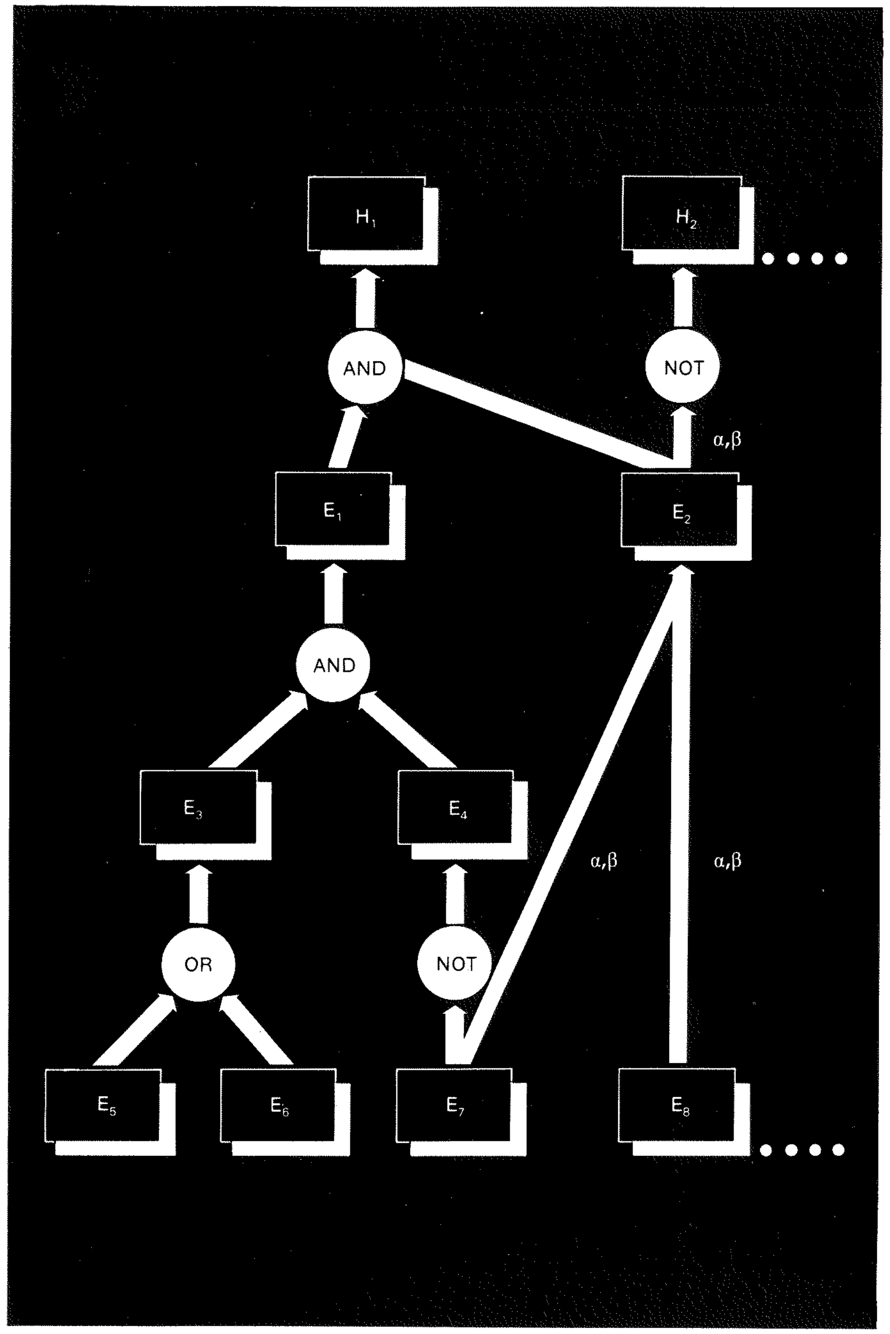
Ciò che nell'uomo viene chiamata *professionalità, capacità* (quando non addirittura intelligenza) è in buona parte l'abilità di collegare tra loro regole pratiche, frutto di precedenti esperienze.

E tanto maggiore è la capacità quanto più complessa è la rete *inferenziale* che l'uomo è in grado di padroneggiare.

Lo schema di fig. 2 mostra una tipica rete inferenziale la cui simbologia è la seguente:

- H = ipotesi (conclusione)
- E = evidenza (sub-ipotesi)

Fig. 2 - Esempio di rete inferenziale.



- α,β = grado di sufficienza, rispettivamente di necessità di una evidenza per il verificarsi di un'ipotesi o sub-ipotesi. (α e β esprimono quindi il *livello di confidenza* dell'esperto nel trarre, da certe premesse, determinate conclusioni, finali o intermedie che siano).

3. Come è fatto un sistema esperto

Da quanto detto, un sistema esperto dovrebbe quindi essere in grado di generare, a fronte di una specifica richiesta, una risposta del tipo:

"se (A) è vero e (B) è vero, vi è un X% di possibilità che (C) sia vero".

Per far ciò la struttura di un sistema esperto (fig. 3), così come oggi la si ipotizza, è costituita da:

- una *base di competenza* che contiene, opportunamente rappresentato, tutto ciò che un esperto conosce del settore specifico e che è esprimibile sotto forma di regole euristiche;
- un *sistema di inferenza* (o "motore inferenziale") che sceglie e collega le regole per risolvere il problema posto.

La base di competenza è ovviamente tipica del settore, mentre la *macchina di interferenza* ne è concettualmente svincolata; si basa infatti su meccanismi logici *standard.*

A un *sistema esperto* così concepito non si chiedono più semplicemente informazioni, ma si chiedono invece pareri, indicazioni su come affrontare un problema, cioè ci si rivolge a

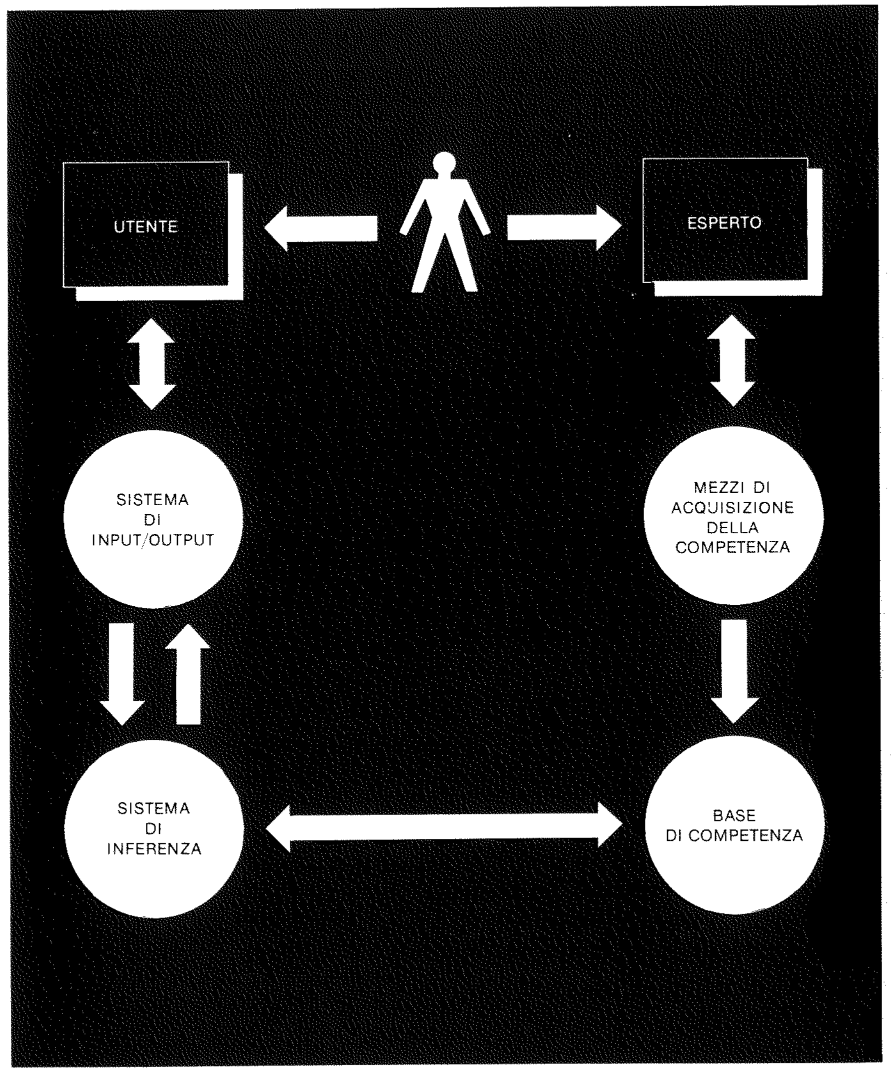


Fig. 3 - Struttura di un sistema esperto.

lui proprio come lo si farebbe con un esperto del settore.

La risposta del sistema non si limita alle conclusioni, ma le giustifica spiegando l'iter seguito per arrivarci e dando una indicazione della loro attendibilità alla luce della competenza accumulata.

Tale *competenza* può essere progressivamente arricchita da nuove situazioni man mano che il sistema si trovi ad affrontarle ("Il sistema è in grado di imparare").

4. Come opera un sistema esperto

Lo schema di fig. 4 illustra il modo di operare degli algoritmi di inferenza sulle regole e sulla rappresentazione del problema da risolvere.

Si tratta sostanzialmente di un ciclo "*riconoscimento-attivazione*", in cui, ad ogni passo, vengono individuate le regole pertinenti, viene selezionata la più appropriata e infine vengono eseguite le azioni in essa prescritte sui dati del problema, producendo in tal modo nuove informazioni che verranno utilizzate nei cicli successivi.

Gli aspetti più critici di questa impostazione e che rappresentano tuttora argomento di ricerca così negli Stati Uniti, come in Europa e in Giappone, sono:

- la definizione della struttura delle regole e le modalità della loro rappresentazione;
- lo sviluppo degli algoritmi di selezione delle regole da attivare;
- la definizione delle procedure di controllo per la gestione del funzionamento globale del sistema.

Nel contesto dei sistemi esperti nasce il nuovo ruolo dell'*ingegnere della competenza*, cioè di colui che stabilisce i criteri per individuare, estrarre e organizzare in modo efficiente le competenze possedute dall'esperto umano e che vanno trasferite al sistema.

5. Limiti e potenzialità dei sistemi esperti nell'ufficio

Cercheremo di indicare alcuni limiti e accennare alle potenzialità applicative dei sistemi esperti nell'ufficio pur rendendoci conto che, oggi, può essere azzardato fare delle previsioni in un settore carente di esperienze consolidate e per tanti versi ancora da esplorare a livello di ricerca.

Tuttavia, alcuni esempi di sistemi esperti sono operativi, in forma prototipale, non solo in campo medico ma anche nel lavoro manageriale ed è sulle loro prestazioni che si può tentare qualche estrapolazione.

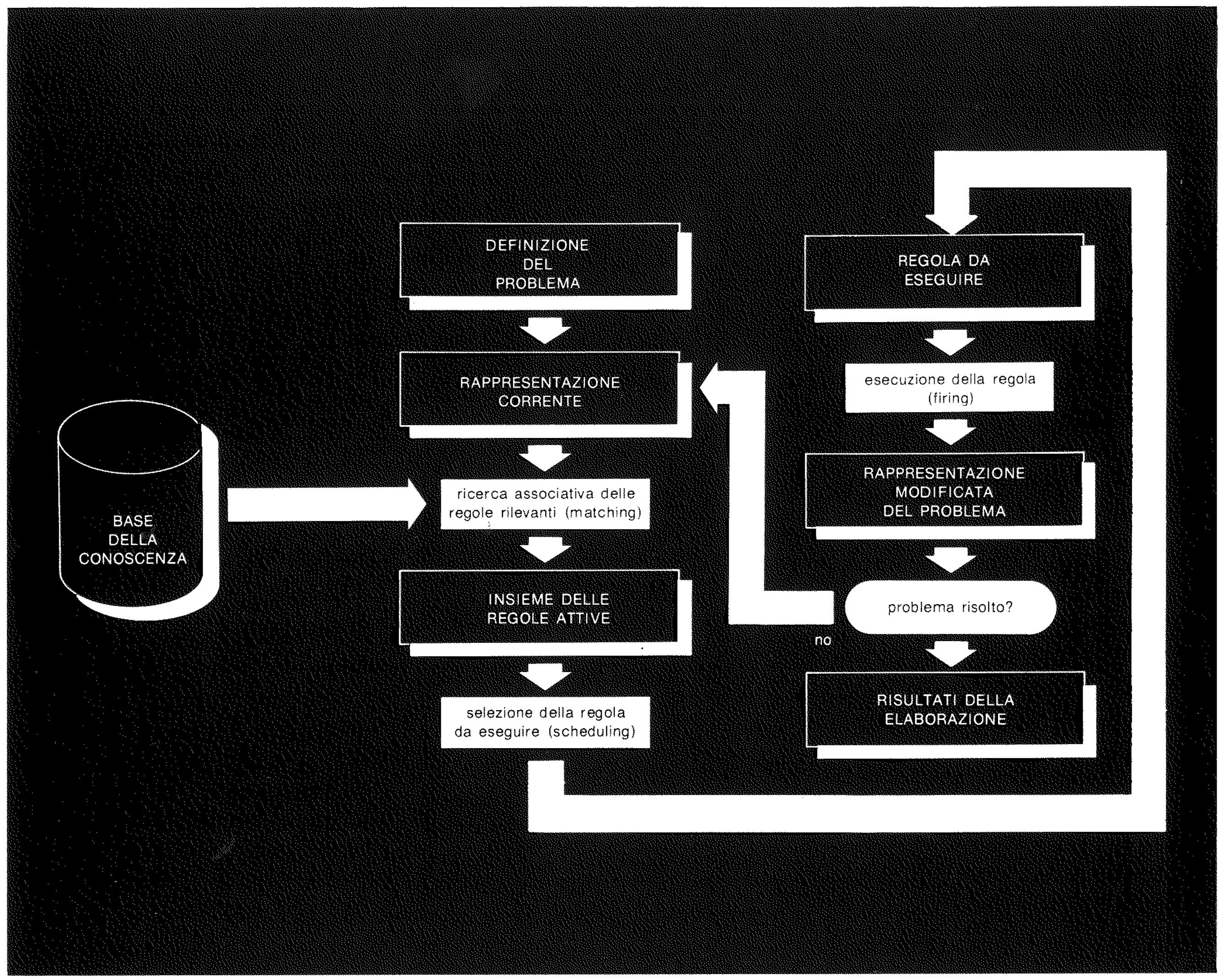


Fig. 4 - Ciclo di riconoscimento-attivazione di un sistema esperto.

Tra le potenzialità, la più generale e importante riguarda un aspetto concettuale dei sistemi esperti di particolare rilievo nell'ufficio e, più generalmente, nell'azienda.

L'ufficio è per definizione un ambiente dove la mobilità del personale dovrebbe essere (in molti casi lo è) elevata e la competenza professionale di chi subentra in una funzione nuova deve essere costruita attraverso un laborioso ciclo di formazione e di esperienze maturate in prima persona. Questo perchè l'unico «serbatoio» di regole codificate della azienda, la raccolta, cioè, delle sue norme e procedure, è finalizzato a compiti di controllo e non di accumulazione e trasmissione di know-how.

In questo ambito il sistema esperto potrebbe giocare il ruolo di presidio della professionalità di base richiesta per l'espletamento delle varie funzioni aziendali, aiutando sin dall'inizio chi vi subentra a evitare quantomeno quei comportamenti che l'esperienza ha dimostrato negativi.

In altre parole, il sistema esperto che è in grado, data una situazione e una linea di azione suggerita, di documentare il processo logico che lo ha portato a sceglierla tra le tante alternative, potrà essere impiegato per addestrare tutti coloro che non siano ancora autonomi in una specifica attività.

L'analisi del processo decisionale sviluppato in situazioni diverse, siano esse reali o opportunamente simulate, costituirebbe infatti una metodologia didattica pragmatica ed efficace per i futuri specialisti e manager.

Per quanto concerne le limitazioni prevedibili, queste riguarderanno certamente:

- il campo di azione dei sistemi esperti che, almeno all'inizio, sarà necessariamente ristretto per la difficoltà oggettiva di costruire basi di competenza non rigorosamente limitate
- la incapacità umana di descrivere in modo esaustivo le competenze relative a fatti e situazioni riducendole a regole di produzione del tipo SE ... E ... ALLORA... a

principi euristici o ad altre forme di espressione

- i costi di sviluppo che, in particolare nel lavoro manageriale, saranno rilevanti proprio perchè si tratta di un'area nella quale la limitazione del campo di azione e le difficoltà di formalizzazione della competenza, rappresentano grossi ostacoli ad un impiego efficace.

Il rischio è che i sistemi esperti per il management siano o troppo complessi da utilizzare o, se ragionevolmente semplici, incapaci di fornire risposte non banali.

Queste considerazioni, tuttavia, non devono far dimenticare che i sistemi esperti sono alla base del progetto della cosiddetta 5ª generazione di sistemi informatici (fig. 5) su cui sono mobilitati i migliori centri di ricerca industriale e universitaria del Giappone, degli Stati Uniti e dell'Europa e che uno sforzo congiunto di tale entità non ha forse precedenti nella storia dell'informatica.

I risultati saranno certamente significativi e il loro impatto sul lavoro di ufficio di grande momento proprio perchè rivolto a ridurre l'area di soggettività (e aleatorietà) oggi connaturata con la sua struttura.

Nel seguito si propongono alcuni possibili scenari di impiego produttivo dei sistemi esperti con l'obiettivo di sottolinearne le componenti operative.

6. Sistemi esperti nell'area commerciale

Non è facile per una azienda misurare efficacemente la qualità del singolo cliente soprattutto se il loro numero è rilevante.

Eppure si possono ipotizzare sistemi valutativi acquisibili dal computer per farsi una *base di competenza* sulla clientela.

Alcune *evidenze* possono essere:

- lo scarto fra il prezzo al quale il cliente acquista e il prezzo di listino;
- le condizioni di pagamento nominali che con lui sono state nel tempo convenute;
- il rispetto con il quale tali condizioni sono praticate dal cliente (giorni di ritardo nel caso di rimessa diretta, numero valore medio e percentuale degli insoluti nel caso di ricevute bancarie, ...);
- la tempestività con la quale il cliente corrisponde al lancio di un nuvo prodotto, acquistando;
- il costo per servirlo (numero di visite, frammentazione delle consegne, condizioni particolari di resa, ...);
- la fedeltà con la quale acquista (la quota di mercato);
- il dosaggio ("mix") di prodotti che acquista, misurato attraverso il margine di contribuzione complessivo;
- .....

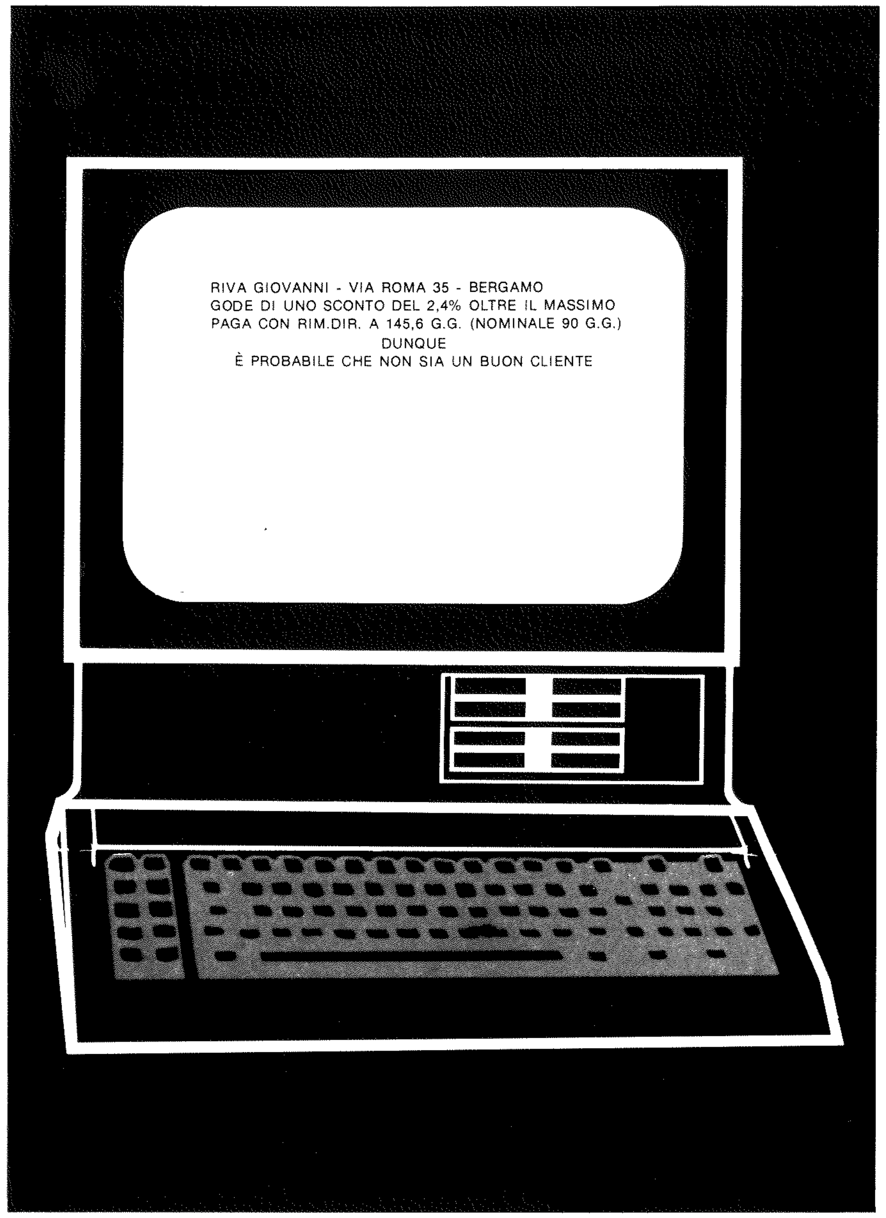
Alcune di queste evidenze e subipotesi possono agevolmente aggregarsi in *ipotesi*.

Sulle precedenti evidenze e sulle conseguenti ipotesi si possono certamente sviluppare delle conclusioni atte a misurare o pesare la bontà del cliente con un conveniente *livello di confidenza.*

ASPETTO	TECNOLOGIA
INTERFACCIA UTENTE	LINGUAGGIO NATURALE
RISOLUZIONE PROBLEMI	SISTEMI ESPERTI IN VARIE AREE
ELABORAZIONE	ALTO PARALLELISMO DI ESECUZIONE
PRESTAZIONI	MILIONI DI INFERENZE AL SECONDO

Fig. 5 - I componenti tecnologici della quinta generazione di sistemi di elaborazione.

Fig. 6 - Un sistema esperto per la valutazione della "qualità" dei clienti.



Non è impensabile già oggi costruire di riflesso un *sistema di inferenza* che assista il comportamento della direzione commerciale o del controllo della gestione commerciale con criteri discutibili ma certamente più efficaci ed efficienti di altri più grossolani, istintivi, soggettivi e meno concretamente applicabili (fig. 6).

Un'altra area interessante è la distribuzione di beni di largo consumo dove molte decine o poche centinaia di prodotti vengono venduti a diverse decine di migliaia di clienti attraverso sistemi logistici diversificati e complessi. Questa area si presta certamente all'applicazione di un conveniente modello inferenziale, "delegato" alla gestione dell'elaboratore.

Considerazioni analoghe potrebbero farsi nel settore bancario (o in quello tributario) per interpretare la "qualità" di un'azienda attraverso alcuni parametri o dati classici (si parla di "analisi di bilancio") ma, ancor più, integrando tali dati con altri relativi all'andamento dei vari rapporti intrattenuti con la banca o col sistema bancario (centrale dei rischi) o alla informazione corrente (bollettino dei protesti cambiari,...).

Un sistema esperto "configura" i grandi elaboratori

DAVE ROLSTON

Honeywell
Large Computer Products Division
Phoenix, Arizona (USA)

1. Introduzione

Alla lunga, val meglio insegnare ad un uomo affamato a pescare, che regalargli un pesce. Similmente, è meglio insegnare a un elaboratore a trovare la soluzione dei problemi, che fornirgliene una. È ancora lontana l'apparizione di programmi di elaboratore che abbiano la capacità umana di risolvere un vasto campo di problemi, tuttavia vi sono sistemi di software che possono eguagliare o superare gli esseri umani in un limitato campo di problemi. Fra questi, quelli che hanno maggior successo sono i sistemi fondati sulla conoscenza, detti comunemente Sistemi Esperti.

L'importanza data alla conoscenza specifica, o competenza, è relativamente recente. I primi tentativi di progettare programmi intelligenti presupponevano che l'intelligenza fosse in gran parte capacità di ragionamento, e che un programma che avesse una solida padronanza della logica formale avrebbe potuto risolvere un vasto campo di problemi. Un esempio è dato dal QA3, programma di domanda e risposta capace di risolvere giochi di abilità, semplici problemi di chimica, di movimento di robot, e di programmazione automatica.

È risultato però che il QA3 e programmi simili non potevano risolvere che problemi molto semplici. Per risolvere qualsiasi problema occorre disporre di informazioni specifiche nell'area del problema stesso; una delle più sconcertanti scoperte fatte con i primi lavori sull'intelligenza artificiale fu che per risolvere la maggior parte dei problemi occorre una enorme quantità di informazione. Se però gli elementi di informazione vengono trattati come oggetti indistinti e manipolati alla cieca, la ricerca di una soluzione diventa tanto lenta da risultare impossibile.

Così la ricerca sui programmi generalizzati di soluzione di problemi ha condotto a capire che i metodi generali non sono efficaci a meno che non si accompagnino ad una estesa conoscenza del dominio del problema. Per risolvere dei problemi di fisica occorre conoscere le leggi della fisica e saperle applicare.

Conseguentemente, l'interesse principale della ricerca si è spostato dal meccanismo di ragionamento alla base di dati su cui esso opera e la base di dati è divenuta una base di conoscenza. In altre parole, le conoscenze specifiche sono registrate in memoria insieme alle istruzioni, implicite o esplicite, per il loro impiego.

Questo modo si affrontare il problema è risultato il più redditizio: i programmi basati su conoscenze specifiche possono risolvere problemi estremamente complessi in un tempo ragionevole.

Ma proprio come è risultato difficile costruire un solutore generale di problemi, è anche molto difficile realizzare una base di conoscenza enciclopedica.

Una difficoltà sta semplicemente nella enorme quantità di cose che costituiscono lo scibile; un'altra è la specificità dei metodi normalmente usati per rappresentare la conoscenza: ogni metodo permette di risolvere certi problemi con maggior efficienza di altri, nessuno è efficiente per tutti i problemi. Gran parte delle ricerche in corso sull'intelligenza artificiale riguarda il modo di rappresentare la conoscenza. Per ora i sistemi basati sulla conoscenza sono limitati a problemi il cui dominio è relativamente ristretto. Essi sono quindi "esperti" tanto nel senso che sono specializzati, quanto in quello che sono eccezionalmente abili.

I problemi più adatti ai sistemi esperti sono quelli molto complicati ma di dominio relativamente limitato. In questo caso la base di conoscenza è di dimensioni trattabili. Se il problema è complicato, la velocità di operazione e l'infallibile memoria del calcolatore danno a questo un vantaggio sull'operatore umano. Esempi di tali problemi sono la diagnosi di infezioni da batteri nel sangue (dominio del ben noto sistema esperto detto *Mycin*), l'interpretazione degli esami polmonari (dominio del sistema *Puff*), la determinazione della struttura di molecole organiche (dominio di *Dendral*), la localizzazione di depositi petroliferi (dominio di *Prospector*).

Un altro problema caratterizzato da un dominio circoscritto e da una elevata complicazione è quello di "configurare" un grande calcolatore (mainframe). La Divisione Grandi Calcolatori della Honeywell ha sviluppato un sistema esperto, detto SYSCON (da SYSTEM CONFIGURATION), per effettuare tale compito per il DPS 90, il più recente ed il più potente dei mainframe della Honeywell.

2. Il problema.

Un grande elaboratore non è un oggetto che si prende direttamente da magazzino: esso viene infatti costruito su ordinazione del cliente.

Per esempio, il numero e la dimensione delle unità di alimentazione di energia dipendono dalla complessiva capacità di memoria richiesta dal cliente, e così pure la configurazione interna dell'unità fisica che contiene la memoria. Allo stesso modo, il numero dei pannelli necessari per contenere le schede a circuiti stampati che formano un canale di entrata-uscita dipende dal numero dei canali richiesti dal cliente.

Il SYSCON lavora su 479 diversi tipi di componenti che variano, in complessità, dai cavi di collegamento ai processori. Supponendo per un momento che un mainframe abbia 3.500 componenti, e che tutte le combinazioni siano effettuabili, il numero delle combinazioni possibili è 479^{3500}. In pratica, molte combinazioni non sono ammissibili, ma anche così il numero di configurazioni del sistema è enorme e non si può stabilire quale è la configurazione migliore calcolando tutte quelle possibili e paragonandole fra loro. La scelta della miglior configurazione deve essere guidata dall'intelligenza.

Prima del SYSCON, il DPS 90 era configurato a mano. Un funzionario commerciale, in collaborazione con il cliente, forniva le specifiche del sistema, e cioè una sua descrizione generale, formata da "identificatori di marketing", quali il numero delle unità centrali di elaborazione (Central Processing Unit, CPU), delle unità di memoria, e degli altri principali componenti richiesti. Tali specifiche, e le indicazioni circa il locale di installazione, erano trasmesse ad un esperto di configurazioni, che elaborava una descrizione dettagliata del sistema. Poichè il procedimento è stato formalizzato, ora sappiamo che si dovevano prendere da 500 a 800 decisioni. Ne risultava un documento di 40-80 pagine, che identificava ogni componente e ne descriveva l'interconnessione. Esso veniva a far parte dell'ordine di acquisto ed era usato per realizzare il sistema.

La configurazione manuale aveva diversi inconvenienti; in primo luogo, poche persone avevano la competenza necessaria; in secondo luogo, essa è costosa: un esperto impiegava in genere una settimana a configurare un sistema. Infine la configurazione ottenuta doveva sempre essere rielaborata: data la complessità del problema ed il livello di dettaglio richiesto per la soluzione, la prima configurazione non era mai corretta.

Il SYSCON ha eliminato questi inconvenienti: le configurazioni che produce sono coerenti e corrette. Il procedimento richiede circa dieci minuti, e può essere effettuato da chiunque, dopo un corso di istruzione di meno di un'ora.

3. L'intelligenza artificiale

In che senso alcuni programmi sono "più intelligenti" di altri? Molti sono stati i tentativi di definire l'intelligenza artificiale, ma nessuna definizione è completamente soddisfacente ed accettata da tutti. Una definizione ragionevole, tuttavia, è che essa è l'insieme delle tecniche che permette di incorporare in un programma procedimenti flessibili atti alla soluzione di problemi.

La differenza fra l'approccio del programmatore tradizionale, e quello di chi lavora nell'intelligenza artificiale può essere spiegato con un esempio. Supponiamo di avere due secchi vuoti, uno della capacità di sei litri e uno della capacità di otto litri. Nessuno di essi ha indicazioni di livello, e il compito è di riempire esattamente a metà il recipiente da otto litri.

L'approccio diretto a questo problema è semplicemente quello di dare al calcolatore il procedimento per risolverlo. Il programmatore determina con mezzi adatti una sequenza di azioni che danno come risultato un recipiente pieno a metà, e poi codifica questa sequenza in un linguaggio compreso dal calcolatore.

Un approccio del tutto differente è quello di fornire al calcolatore una rappresentazione generale del dominio del problema, e una procedura per ricercare una soluzione entro questo dominio. Molti dominii possono essere visti come spazi costituiti da stati discreti, ognuno dei quali rappresenta una possibile disposizione degli elementi del problema.

In questo caso il modo ovvio di definire gli stati è in termini del contenuto dei due recipienti. Si definisce la coppia (R, S) di valori come lo stato che corrisponde alla quantità R nel recipiente da otto litri, e della quantità S nel recipiente da sei litri. Poichè all'inizio i recipienti sono vuoti, lo stato iniziale è (0,0).

Dato uno stato iniziale, solo alcune azioni sono possibili. Se lo stato iniziale è (0,0), le sole possibilità sono di riempire l'uno o l'altro o ambedue i vasi. Dato lo stato iniziale, le regole che definiscono le azioni permesse consentono di identificare gli altri stati validi. Se lo stato iniziale è (0,0) e l'azione compiuta è quella di riempire il recipiente da otto litri, allora lo stato risultante è (8,0).

Lo stato iniziale e le regole stabilite definiscono implicitamente una struttura ad albero. Lo stato iniziale è la radice, le regole sono i rami, gli stati validi sono i nodi di diramazione. Secondo questa rappresentazione, il trovare la soluzione corrisponde al trovare un cammino dallo stato iniziale a quello richiesto. Poichè si richiede di riempire esattamente a metà il recipiente da otto litri, lo stato richiesto è quello in cui R è uguale a quattro. I rami compresi nel cammino dallo stato iniziale a tale stato definiscono una sequenza di azioni che danno il risultato richiesto.

Il problema di andare da qui a lì sembra abbastanza semplice. In realtà lo spazio degli stati per il pro-

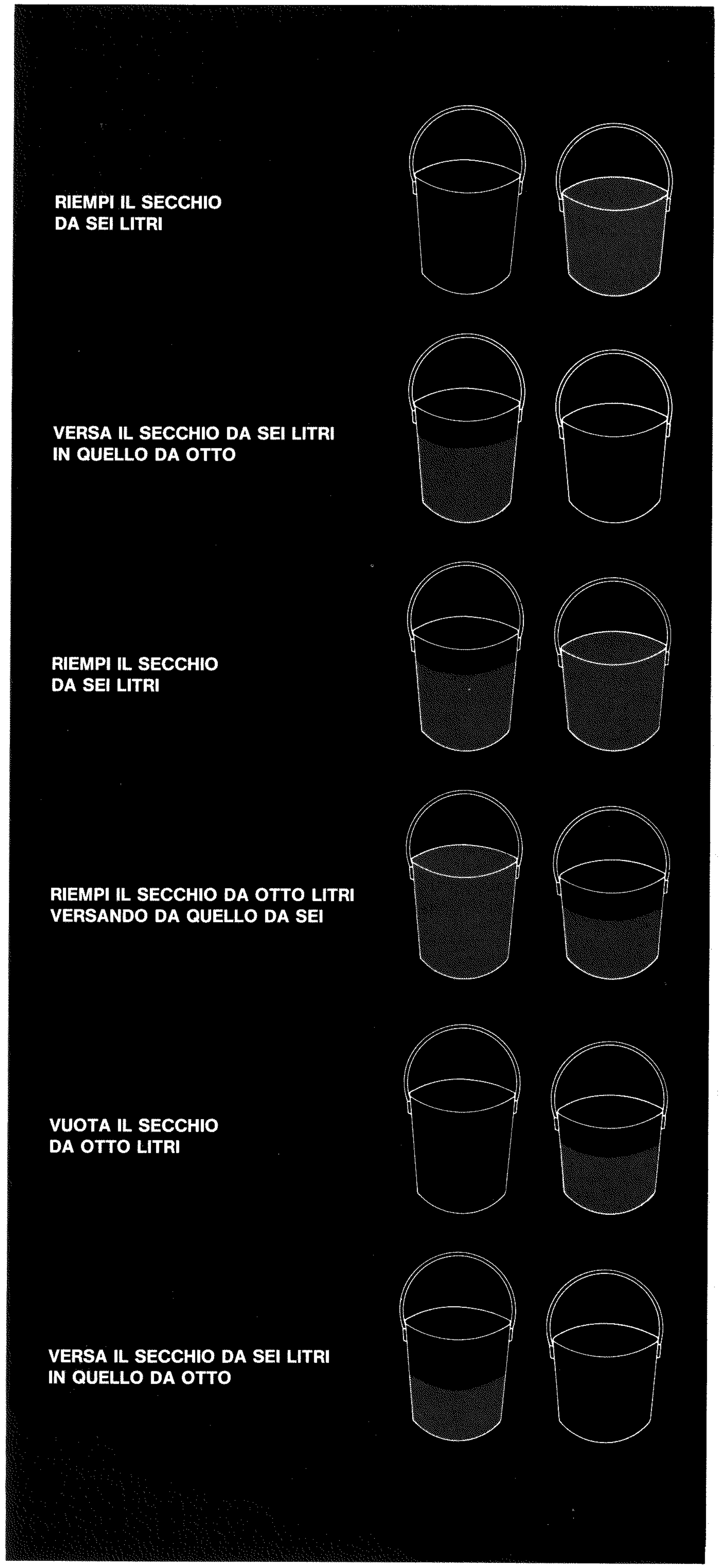


Fig. 1 - La figura illustra un semplice algoritmo per il problema dei due secchi. Dati due secchi, uno dei quali con una capacità di sei litri e l'altro di otto litri, senza nessuna graduazione, come si può riempire quest'ultimo esattamente a metà? Nel modo tradizionale, il programmatore esamina il problema, individua una soluzione e scrive un algoritmo costituito da passi attraverso cui si raggiunge la soluzione. Questo costituisce un approccio efficiente per risolvere uno specifico problema.

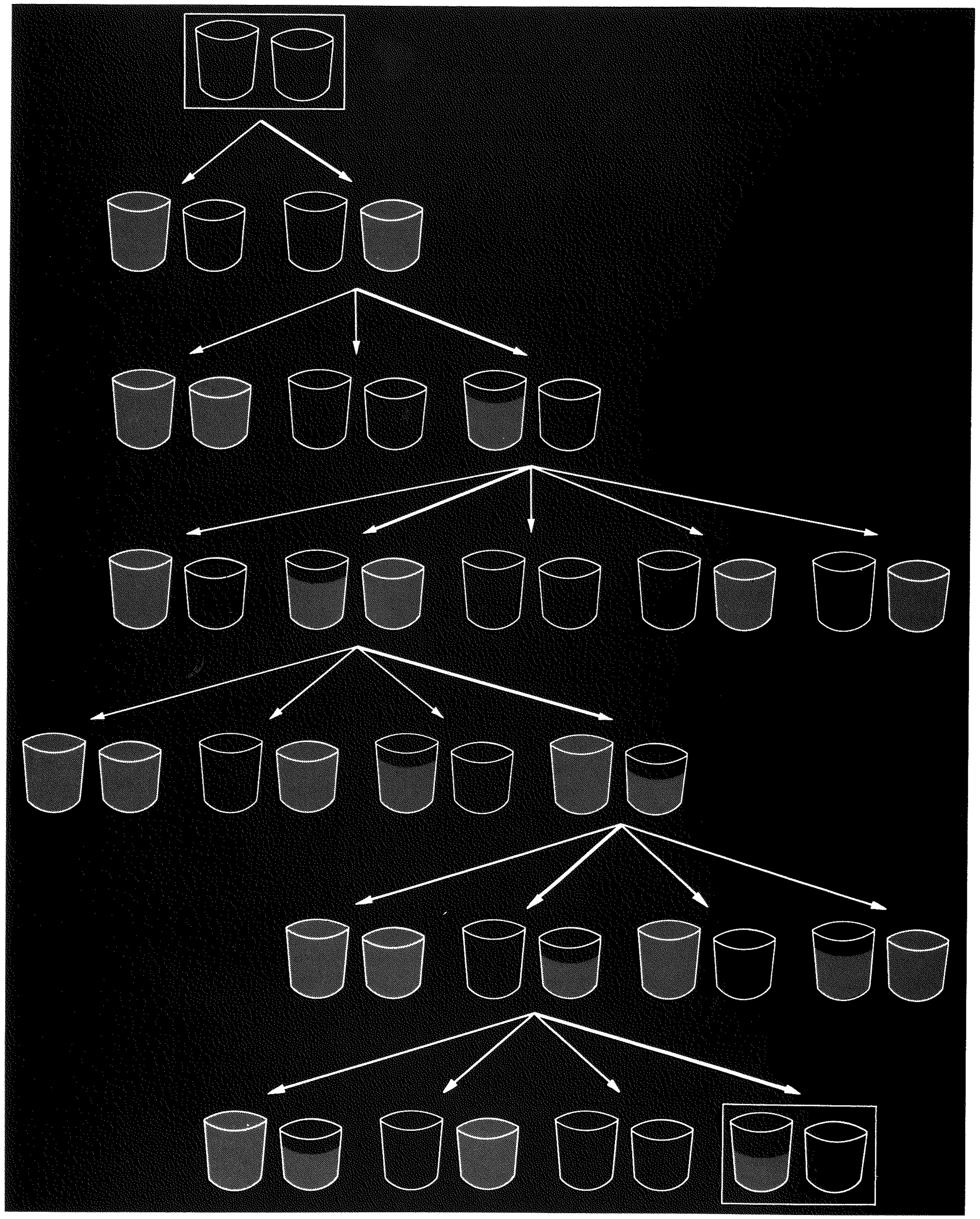


Fig. 2 - Un programma di intelligenza artificiale fornisce, anziché i passi con cui uno specifico problema può essere risolto, gli strumenti necessari per trovare una soluzione ad un tipo di problemi. Il problema dei due secchi può essere rappresentato mediante un diagramma come in figura, cioè con un albero di transizioni di stato. Ogni stato rappresenta un paio di possibili valori del contenuto dei secchi e ciascuna transizione corrisponde ad una azione, ad esempio riempire un secchio. Data una rappresentazione nello spazio degli stati del dominio del problema, lo stato di partenza e le regole che definiscono le transizioni ammesse, trovare una soluzione del problema dei due secchi corrisponde a trovare un percorso dallo stato di partenza a quello desiderato. Poiché lo spazio degli stati è amplissimo, il programma deve includere un criterio di ricerca, ossia una strategia che consenta all'elaboratore di decidere quale strada scegliere in corrispondenza di ogni possibile ramificazione. La figura mostra la parte dello spazio degli stati relativo al problema dei due secchi, che include il percorso fornito dall'algoritmo della figura 1.

blema dei due recipienti è infinito. Se si lascia che il programma vaghi alla cieca entro di esso, vuotando e riempiendo vasi come l'apprendista stregone, può non giungere mai ad una soluzione. Perciò il programma deve contenere anche qualche mezzo per determinare la migliore mossa successiva, e cioè deve includere una procedura di ricerca.

L'approccio diretto al problema dei due recipienti è, evidentemente, un esempio estremo del metodo tradizionale di programmazione. Tali tecniche tradizionali richiedono, generalmente, che il programmatore esamini a fondo, con il ragionamento, il problema, prima di scrivere il programma; perciò il programma ne incorpora una soluzione predeterminata. L'altro modo di affrontare il problema fa parte dell'insieme di tecniche proprie dell'intelligenza artificiale. Esse tentano esplicitamente di introdurre il ragionamento entro il programma: ciò fornisce i mezzi per risolvere tutti i problemi di un certo tipo anzichè un problema specifico. Per esempio, il programma per i due recipienti includerà una rappresentazione del dominio, la conoscenza delle azioni possibili, le condizioni necessarie per compiere queste azioni, e una procedura di ricerca.

Un programma costruito secondo il primo metodo è flessibile solo in quanto può in genere accettare un certo campo di valori di ingresso, per esempio i numeri interi positivi, mentre il cammino dall'ingresso alla soluzione è rigidamente predisposto nel programma. Un programma che contenga in sè la capacità di ragionare sarà assai più flessibile. In primo luogo, potrà adattarsi a diversi modi di enunciare il problema. Per esempio, una sua forma dovrebbe poter determinare i passi necessari per far sì che un recipiente venga a contenere un terzo di acqua in meno dell'altro.

Inoltre, il programma dovrebbe poter effettuare parte del compito richiestogli anche se gli vengono fornite informazioni incomplete o parzialmente inesatte. Per esempio, il programma deve poter sostituire una informazione mancante con un'ipotesi ragionevole e, se questa conduce ad un vicolo cieco, tornare indietro e cambiarla. D'altra parte, il programma dovrebbe sapere quando ha raggiunto il proprio limite di comprensione, in modo da non affrontare problemi insolubili e fornire false soluzioni. Infine deve poter spiegare il suo modo di ragionare in maniera accettabile da un essere umano.

La flessibilità non è il solo vantaggio dei programmi intelligenti, e forse non ne è nemmeno il più importante. Benchè l'esempio fatto possa far credere che il metodo diretto sia anche il più semplice, non sempre è così. Se il problema è complesso, può essere estremamente difficile trovare una soluzione diretta. Può essere più facile scrivere un programma per trovare la soluzione, che non risolvere direttamente il problema.

4. Sistemi basati sulla conoscenza

Come vi sono modi di risolvere i problemi, così vi sono molti modi di incorporare il ragionamento in un programma. I programmi esperti danno particolare importanza ad uno degli elementi del ragionamento, la conoscenza. Essi hanno due elementi fondamentali: una base di conoscenza ed un sistema di inferenza. La base di conoscenza è un insieme di informazioni pertinenti ad un particolare dominio di problemi. Il sistema di inferenza è un meccanismo che deduce nuove conoscenze dalle conoscenze contenute nella base di conoscenza. Generalmente il sistema di inferenza differisce dalla base di conoscenza in quanto esso non è specifico del dominio, ma si fonda su semplici regole applicabili a diversi dominii. Lavorando congiuntamente con la base di conoscenza, il dispositivo di inferenza ricava la conoscenza necessaria per risolvere un problema specifico. Per fare un semplice esempio: se si dice al sistema che Fido è un cane, e se la base di conoscenza include la regola: "tutti i cani sono animali", il meccanismo di inferenza conclude che Fido è un animale.

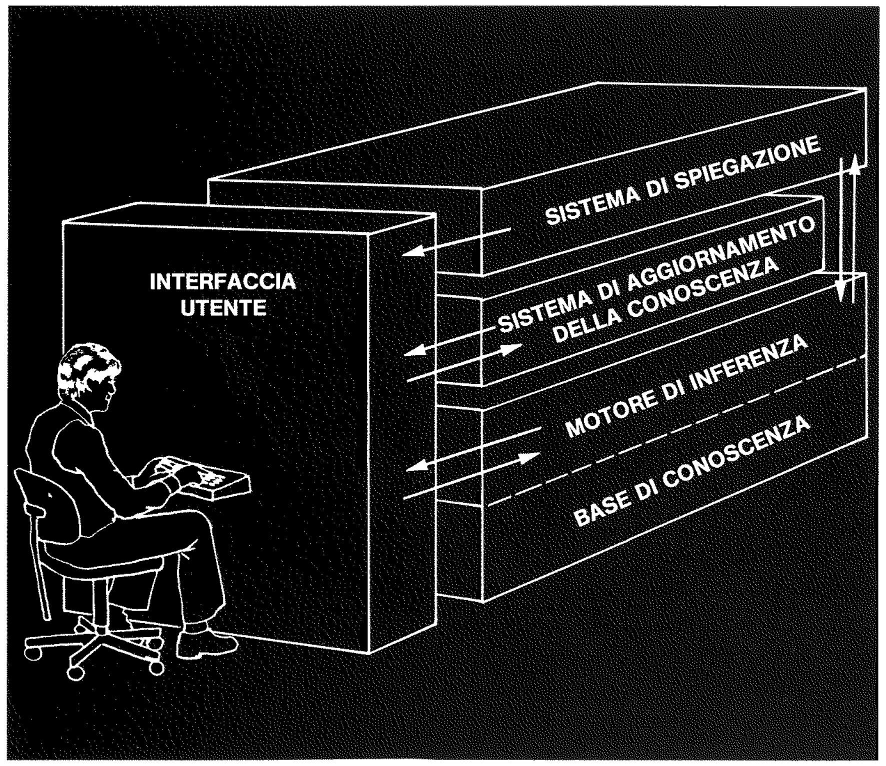


Fig. 3 - Un sistema esperto ha due componenti fondamentali: una "base di conoscenza" e un "motore di inferenza". La base di conoscenza contiene le informazioni e i ragionamenti che un esperto umano impiega nel risolvere un particolare tipo di problema. Il motore di inferenza è un meccanismo logico che consente di dedurre nuove informazioni dagli enunciati del problema e di estrarre argomenti dalla base di conoscenza. Il "sistema per l'aggiornamento della conoscenza" aiuta l'utente che non ha familiarità con la struttura della base di conoscenza, ad inserirvi nuovi argomenti o a cambiare quelli esistenti. Il "sistema di spiegazione" fornisce passo per passo la traccia del ragionamento effettuato dalla macchina, in modo che l'utente non debba accettarne le soluzioni a scatola chiusa.

I sistemi esperti comprendono altri tre componenti i cui compiti sono più specifici: un'interfaccia di utente, un sistema di spiegazione, e un sistema per aggiornare la base di conoscenza. Essi aiutano l'utente ad aggiungere informazioni alla base di conoscenza, a introdurre il problema e le informazioni necessarie per superare eventuali difficoltà, e a verificare il ragionamento con cui il sistema raggiunge la soluzione.

Come programma di intelligenza artificiale, l'abilità di ragionamento di un sistema esperto non è particolarmente impressionante. Può sembrare paradossale che un sistema avente una capacità di ragionamento rudimentale possa risolvere problemi complessi, ma il paradosso è solo apparente. La base di conoscenza comprende, sia nella formulazione delle singole voci che nella loro organizzazione, il ragionamento circa l'informazione ivi contenuta.

Naturalmente, non è detto che questo sia il modo migliore per affrontare qualsiasi problema. Ogni programma per la soluzione di problemi deve poter utilizzare le conoscenze appartenenti al dominio del problema, ma tali conoscenze sono meno importanti in certi casi che in altri. Un esempio è dato dai problemi relativi ai giochi: i dispositivi di ricerca sono altrettanto importanti quanto la conoscenza del mondo semplificato in cui il gioco è condotto. L'elaborazione delle immagini e delle parole, i programmi di robotica, sono altri esempi di programmi di intelligenza artificiale che non dipendono in maniera così importante dalle conoscenze specifiche.

5. Sistemi di produzione

Una delle più importanti decisioni da prendere nel progetto di un sistema esperto è la scelta di uno schema di rappresentazione e di organizzazione della conoscenza.

Poichè il modo di rappresentazione determina l'efficacia del modo di ragionare, esso deve essere tale da adattarsi al problema che il sistema deve risolvere.

Un modo di rappresentazione è il calcolo dei predicati, una notazione di logica formale da lungo tempo in uso fra i matematici. Un'altra rappresentazione, inventata come modello della memoria associativa umana, è la rete semantica. Essa consiste in nodi che rappresentano oggetti o eventi; e di collegamenti internodali che rappresentano le loro relazioni reciproche. Un terzo schema, particolarmente adatto al ragionamento per analogia è la "cornice" (*frame*), una struttura di dati che elenca gli attributi di un oggetto.

La maggioranza dei sistemi esperti, ivi compreso il SYSCON, impiegano una rappresentazione della conoscenza che Allan Newell ed Herbert Simon, dell'Università Carnegie-Mellon, trovarono verso la fine degli anni '60. Nel processo di costruire modelli di conoscenza umana, essi giunsero alla sorprendente conclusione che gran parte delle conoscenze specifiche dell'uomo possono essere espresse mediante regole del tipo "se-allora". Più generalmente, ogni compito da eseguire che possa essere visto come una sequenza di transizioni da uno stato al successivo, può essere espresso in questo modo, poichè ogni transizione può essere rappresentata da una regola "se-allora". I sistemi esperti basati su tali regole sono chiamati "sistemi di produzione".

La base di conoscenza di un sistema di produzione è formata da regole di produzione: queste sono semplici enunciati "se-allora", che descrivono azioni da compiere quando si verificano determinate condizioni. Per esempio, un sistema di produzione per diagnosticare il guasto di un motore a combustione interna può includere la regola di produzione: "Se c'è la scintilla, se c'è l'aria, se il serbatoio di carburante contiene carburante, se la pompa di alimentazione funziona, allora verificare se il condotto di alimentazione è ostruito". Le clausole "se", in un simile enunciato, sono dette "enunciati di condizione" e la clausola "allora" è detto "enunciato di azione".

L'insieme delle regole di produzione opera in connessione con due altri componenti: un'area di memoria, che tiene conto dello stato attuale del problema, e un programma interpretativo che esamina lo stato attuale ed esegue le regole di produzione che sono applicabili. I tre componenti costituiscono il sistema di produzione.

Il programma interpretativo, o interprete, è una specie di "motore di inferenza". Il suo compito è di ricercare le regole applicabili, e cioè quelle che soddisfano agli enunciati di condizione, e di eseguire l'enunciato di azione della regola stessa. Se l'interprete non trova nessuna regola applicabile, si arresta. Se d'altra parte ne trova più di una, segue una strategia di risoluzione di conflitto, per determinare la regola da eseguire. La strategia seguita da SYSCON, tipica di tali casi, consiste nei seguenti passi: primo, scartare quelle regole di produzione che sono già state eseguite; secondo, scegliere quella regola i cui enunciati di condizione sono stati introdotti più recentemente nella memoria operativa (questo assicura la preservazione dell'ordinamento delle regole di produzione che debbano essere eseguite in sequenza); terzo, se il conflitto non è ancora risolto, scegliere la regola più specifica, e cioè quella con più enunciati di condizione; quarto, se c'è ancora un dubbio, scegliere una regola a caso.

6. L'approccio SYSCON al problema delle configurazioni

Al SYSCON viene fornito un elenco delle principali unità fisiche necessarie per soddisfare le richieste del cliente. SYSCON fa all'utente alcune domande (non più di 26), circa tali unità. Da queste informazioni, SYSCON produce un "abbozzo di configurazione". Dopo che l'utente ne ha confermato la esattezza, SYSCON ne controlla la validità per mezzo di regole di configurazione

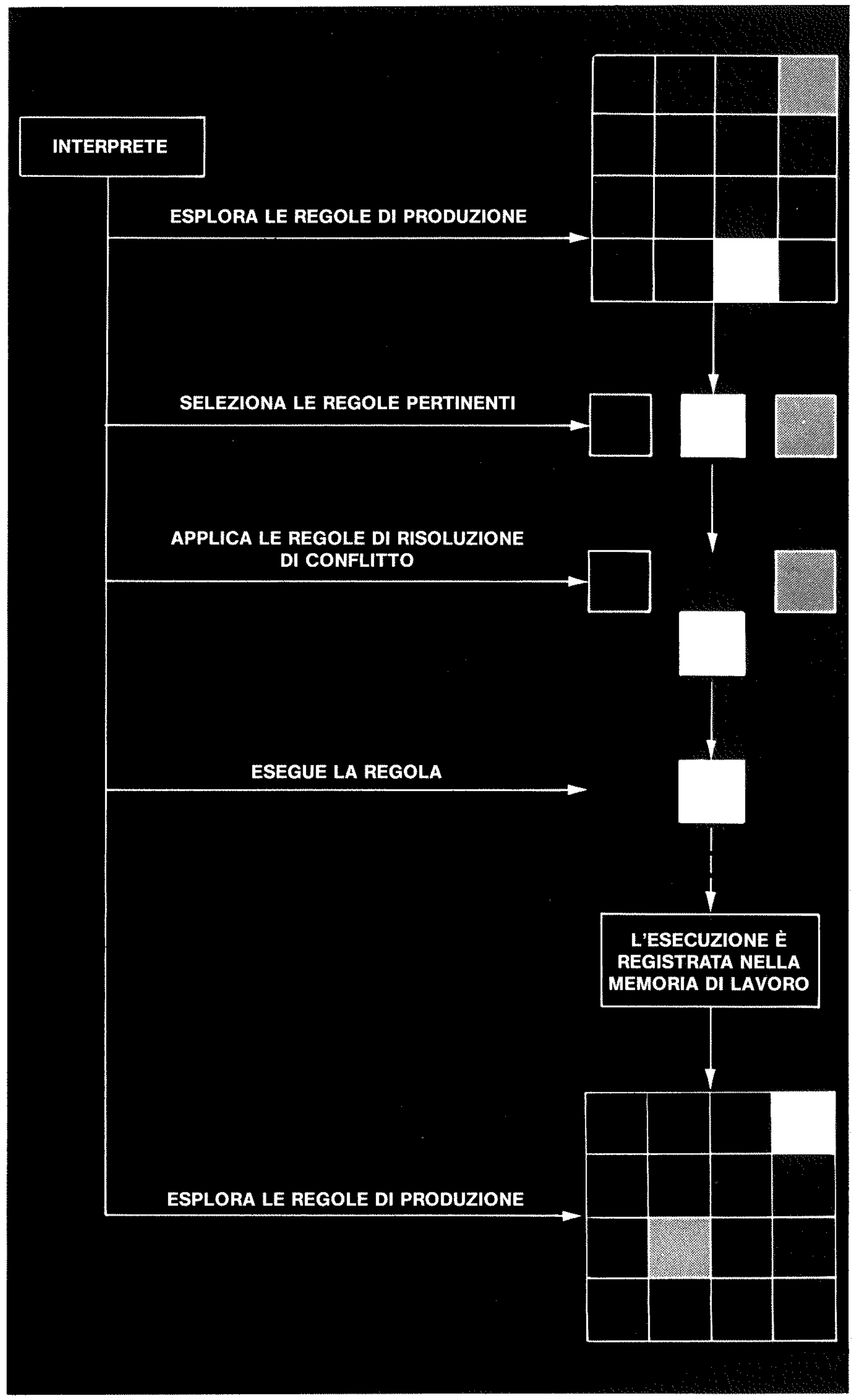


Fig. 4 - L'interprete è un tipo particolare di motore di inferenza che si trova nei sistemi esperti basati su "regole di produzione". La base di conoscenza in sistemi così fatti consiste di regole del tipo "se-allora" ossia aventi la forma di enunciati di condizione e azione. L'interprete ricerca le regole che contengono le condizioni messe in gioco dal problema. Se vi sono più regole che contengono tali condizioni, l'interprete sceglie una di esse in base ad una certa strategia di risoluzione di conflitto. Per esempio, una strategia può essere di scegliere la regola più specifica, cioè quella che contiene il maggior numero di enunciati di condizione. Quando una regola è stata eseguita, il suo enunciato di azione viene inserito nella memoria di lavoro, e costituisce una condizione verificata che servirà per definire il gruppo successivo di regole applicabili.

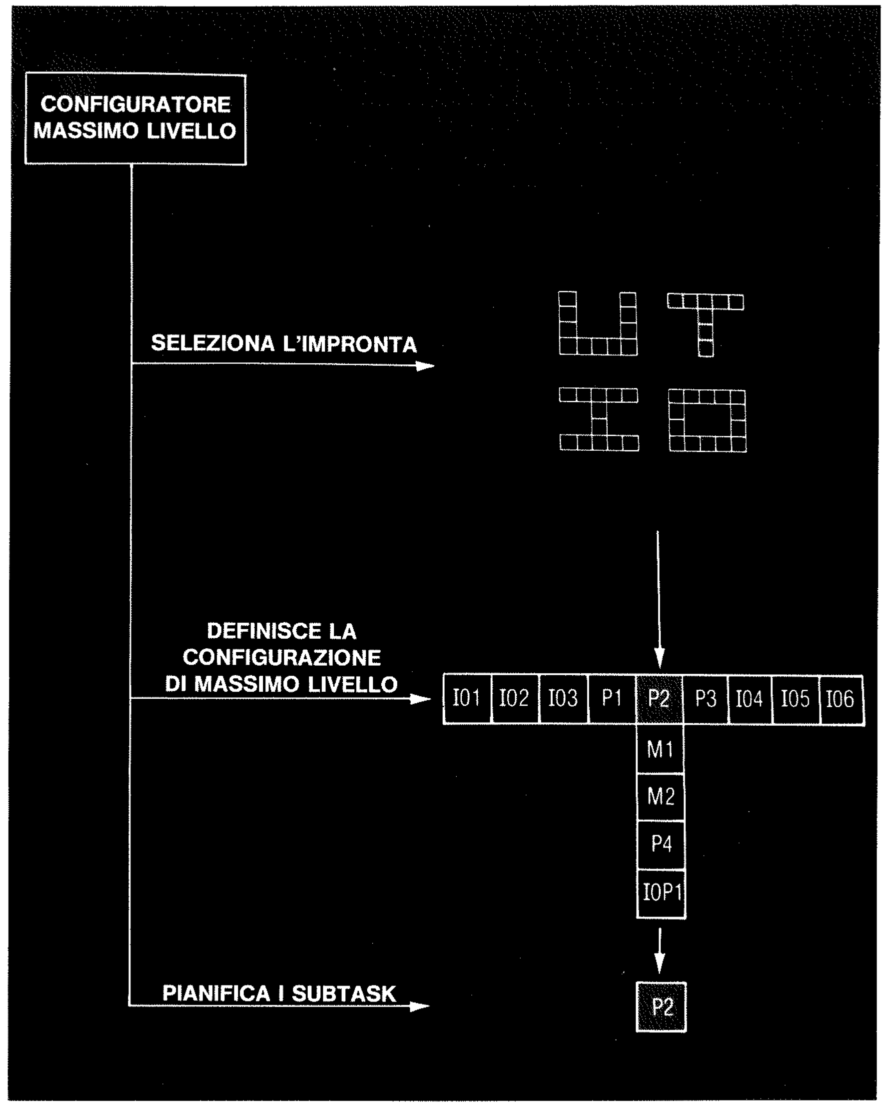
ad alto livello. Due di tali regole sono: "Deve esserci un'unità di elaborazione principale (C.P.U.) e non ce ne possono essere più di quattro" e "Se la memoria eccede le 16 megaparole, devono esserci due armadi di memoria". Se l'"abbozzo di configurazione" conferma che il problema può essere trattato, allora SYSCON intraprende il compito di trovare la soluzione.

Non è affatto ovvio quale sia il miglior modo di iniziare il procedimento di configurazione. Si consideri lo spazio degli stati comprendente tutte le configurazioni a cui si può giungere per una qualsiasi sequenza di regole di produzione.

Non è uno spazio infinito, perchè vi è un limite al numero totale di componenti di un sistema, ma è certo troppo grande per una ricerca alla cieca. Occorre trovare un metodo che porti il sistema a raggiungere una configurazione accettabile. In un certo senso, ciò significa che occorre anticipare la soluzione del problema, prima di risolvere il problema stesso. Il paradosso viene superato identificando una soluzione ad un alto livello di astrazione, lasciando ad un secondo tempo la decisione sui particolari.

Benchè il DPS 90 comprenda migliaia di componenti, ad un alto livello di astrazione esso consiste di non più di 24 unità funzionali. (Il numero di esse dipende dall'ordinazione del cliente). Tali unità sono

Fig. 5 - Il configuratore di massimo livello semplifica la ricerca della configurazione migliore risolvendo il problema ad un elevato livello di astrazione. Partendo dalla lista dei maggiori componenti richiesti dall'ordine del cliente, il configuratore in questione sceglie una tra un certo numero di disposizioni in pianta standard ("impronte"). Successivamente, determina una possibile disposizione dei blocchi logici, applicando regole di produzione specifiche dell'impronta scelta. La configurazione di massimo livello guida poi la ricerca di una configurazione fisica dettagliata. Per configurare ciascuna delle unità logiche costituenti il sistema, vengono attivate specifiche attività (subtask); un subtask viene attivato anche per determinare le interconnessioni tra le varie unità. I subtask sono eseguiti nell'ordine corrispondente alla complessità delle loro interconnessioni; cioè viene configurata per prima l'unità fisica (armadio) avente il maggior numero di interfacce con le altre unità.



dette "complessi logici" e ognuna può corrispondere ad una sola unità fisica (armadio) o a diverse di esse. Dato un modello semplificato del sistema, il problema della configurazione può essere suddiviso in un insieme di sottoproblemi, ognuno dei quali riguarda uno o più complessi logici. Il compito di configurare il sistema diventa allora quello di configurare i complessi logici e di definire i compiti secondari che determinano la configurazione propria di ogni complesso logico.

Col definire un modello astratto del sistema si raggiunge lo scopo di semplificare lo spazio degli stati e, con ciò, di facilitare il raggiungimento di una soluzione particolareggiata. Questi compiti secondari richiedono uno spazio di stati dettagliato, ma il modello astratto fornisce i mezzi per limitare l'ampiezza dello spazio che deve essere esaminato da ognuno dei compiti secondarî.

Anche così, il progetto deve iniziare con una congettura ben indirizzata. La configurazione dei complessi logici, ossia la configurazione di massimo livello, è basata su una tra 22 sagome, dette "impronte", corrispondenti a disposizioni in pianta standard. La scelta dell'impronta è guidata da regole come: "Sistemi con doppio processore devono usare l'impronta a T, anche se vi è un'unità addizionale di entrata-uscita"; "Sistemi in tandem saranno basati su un'impronta ad U" (I sistemi in tandem sono quelli che hanno in doppio ognuno dei componenti maggiori"). L'impronta scelta viene convalidata prima di continuare nel procedimento. Se l'impronta scelta non può soddisfare alle specifiche del sistema, allora se ne sceglie un'altra.

Il SYSCON determina la configurazione di massimo livello con l'applicare regole specifiche per l'impronta scelta. Pe esempio, una delle regole per la configurazione a massimo livello è: "Se esiste un'unità n. 1 di controllo del sistema, e un'unità n. 2 di controllo del sistema, e non c'è una memoria principale n. 2, allora occorre includere i complessi logici detti SCUO, SCU1, MUO1, e MUO2."

Il risultato del processo di configurazione al massimo livello è una lista di massimo livello, e cioè un elenco dei complessi logici e delle loro interconnessioni. Questa lista diviene la prima sezione del documento finale di configurazione ed è usata per generare i "subtask", o compiti secondari. Per ogni complesso logico, un primo compito secondario riguarda il complesso logico stesso, e un secondo le sue interconnessioni con gli altri complessi.

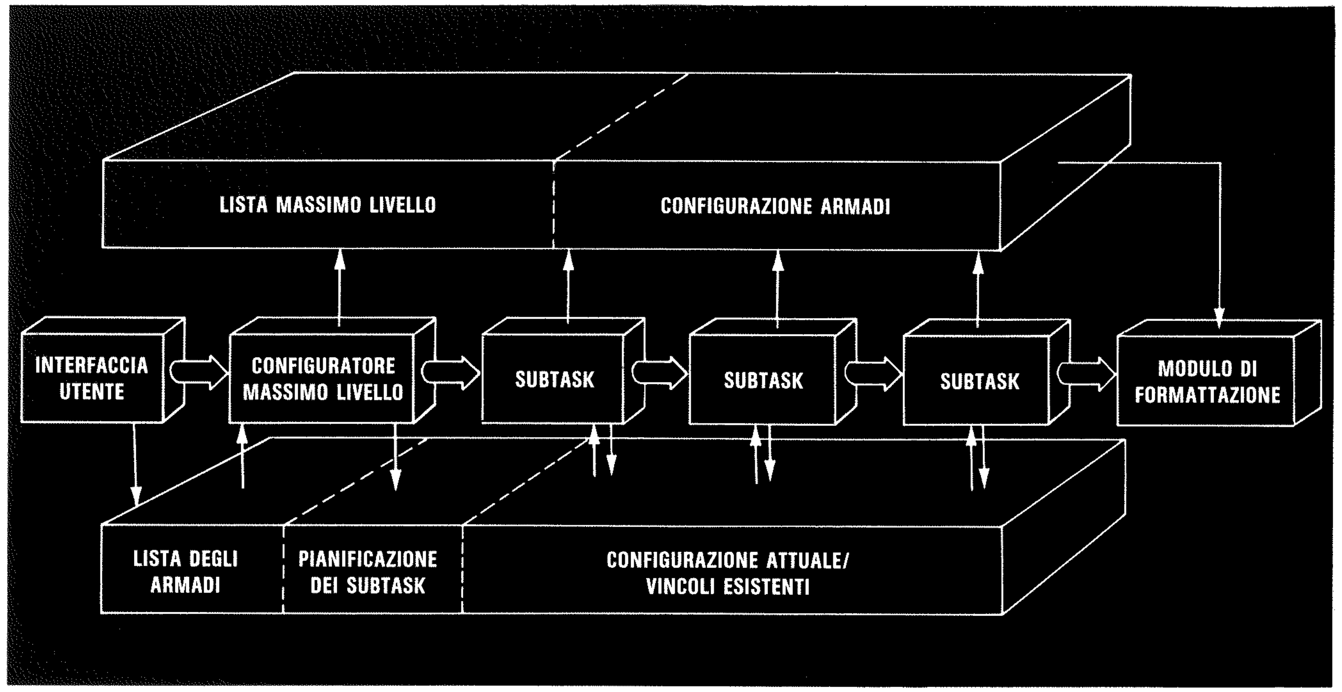


Fig. 6 - All'interno di SYSCON, i vari elementi del programma comunicano mediante una memoria comune (in basso nel disegno). I risultati dell'esecuzione di ciascun modulo del programma vengono registrati in un archivio temporaneo (nel disegno in alto), da cui vengono estratte le informazioni per redigere il documento sulla configurazione definitiva. La lista delle maggiori unità fisiche (armadi) richieste dall'ordine del cliente costituisce l'input di un modulo che definisce una configurazione di massimo livello. Esso consente di suddividere il problema in attività specifiche (subtask) e di determinare l'ordine con cui queste devono essere eseguite. Man mano che ciò avviene, le azioni delle regole di produzione attivate vengono eseguite e ne viene fatta registrazione nella memoria di lavoro. In questo modo, la memoria comune consente la propagazione da un subtask al successivo delle limitazioni via via introdotte, assicurando la coerenza della configurazione finale. Infine, il modulo di formattazione estrae le configurazioni delle varie unità fisiche (armadi) dall'archivio temporaneo, le organizza opportunamente e genera il documento della configurazione.

La difficoltà successiva deriva dal fatto che i complessi logici non sono indipendenti: essendo interconnessi, la configurazione ottenuta per ogni complesso può dipendere dall'ordine in cui vengono eseguiti i compiti secondari. Inoltre, se ciò avviene nell'ordine errato, il SYSCON può finire in un vicolo cieco, e può risultare impossibile configurare l'ultimo complesso in modo coerente con la configurazione dei precedenti.

Per evitare questa difficoltà, i compiti secondari sono eseguiti nell'ordine corrispondente alla complessità delle loro interconnessioni. SYSCON configura dapprima il complesso logico che ha il maggior numero di interfacce con altri complessi, e poi continua con quelli con minor numero di interfacce; e configura per ultimo il più semplice.

Poichè i complessi più semplici sono i più flessibili, questo modo di operare assicura che il SYSCON ha ancora spazio di manovra quando si avvicina al termine del suo compito.

Per giungere ad una configurazione coerente, SYSCON deve tener conto delle limitazioni che insorgono dalla esecuzione di ogni compito secondario, e che si propagano a quelli che seguono. Tali limitazioni sono propagate dalla memoria comune, in cui ogni compito secondario eseguito registra le informazioni relative al complesso logico configurato. Ogni compito secondario esamina, durante l'esecuzione, le informazioni della memoria comune, ed è limitato a quelle azioni che sono valide alla luce delle azioni precedenti. Per esempio, si consideri la seguente regola: "Se ambedue le unità di controllo n. 0 e n. 1 sono incluse nel sistema, e la memoria totale è maggiore di 16 megabyte, e se tutti i canali ad alta velocità dei processori sono occupati, e c'è una piastra cieca della unità centrale di elaborazione, allora aggiungere un secondo modulo di controllo del canale in questa unità di entrata-uscita". È chiaro che il risultato della esecuzione del compito secondario che comprende questa regola è condizionato dai compiti secondari che hanno configurato le unità di controllo del sistema, i processori dei canali ad alta velocità, e le unità di memoria.

Quando tutti i compiti secondari sono eseguiti, il sistema raccoglie le loro uscite e compila il documento finale di configurazione. I risultati seguono un ordine stabilito, secondo la convenienza del personale che emette gli ordini e di quello della fabbrica.

7. Organizzazione del programma

Il SYSCON è scritto in OPS5, un linguaggio di programmazione di sistemi di produzione progettato da Charles G. Fargy dell'Università Carnegie-Mellon. Gli enunciati dell'OPS5 sono interpretati durante l'esecuzione del programma. Il programma interpretativo è scritto in MacLisp, una delle versioni del lin-

COMPONENT NAME CPU

777-123456-103-00 CABINET IDENTIFICATION CPU

NAME	IDENTIFICATION	QTY	ADDRESS CABINET MOUNTING
JUMPER PACKAGE	177-519333	1	PW01 BAB-P16
BAFFLE (HW1C)	177-531924-100	1	PW01 A1
PWP812-X0A	177-414500-107	1	PW01
PW01 PROCESSING ASSEMBLY	177-429928-106	1	PW01 EC203
EPU CCM-SCI CABLE	177-421035-102	1	PW01 BAB-POP
CPU CABINET NAME PLATE	177-414500-444	1	PW01
CPU 099-HPWP	177-305349	1	PW01

Fig. 7 - Il documento generato da SYSCON fornisce la lista dei componenti del sistema da configurare, con le istruzioni per la loro interconnessione. Una lista completa per un sistema DPS 90 può includere fino a 4.000 componenti. La figura mostra un esempio della documentazione generata da SYSCON (NAME = nome del componente; IDENTIFICATION = codice del componente; ADDRESS CABINET MOUNTING = istruzioni di interconnessione).

guaggio di programmazione Lisp. La maggior parte del sistema è scritto in OPS5, ma le sezioni che non si potevano codificare efficientemente in tale linguaggio sono scritte direttamente in MacLisp. Per esempio, le 800 regole di produzione del sistema sono scritte in OPS5, ma la procedura che produce i documenti finali è scritta in MacLisp.

I componenti funzionali del SYSCON che partecipano direttamente al compito di configurazione sono: l'interfaccia di utente, che riceve la lista dei componenti che costituisce l'enunciato del problema; il configuratore di massimo livello, che produce la lista di massimo livello e pianifica i compiti secondari; i compiti secondari che configurano i complessi fisici (armadi) e un modulo di formattazione che legge i risultati dei compiti secondari da archivi provvisori, e produce il documento finale di configurazione.

Tutti questi componenti, nell'effettuare i loro compiti, si riferiscono alla base di conoscenze. Questa è suddivisa in sezioni, relative alla verifica della configurazione, alla configurazione dei complessi logici, a quella delle unità fisiche, alla raccolta finale e al controllo generale.

Il dispositivo di aggiornamento della conoscenza del SYSCON è del tipo detto "editore intelligente". Poichè esso conosce la struttura della base di conoscenza, esso permette ad utenti, che non abbiano completa famigliarità con il programma, di introdurre nuove regole o modificare quelle esistenti.

Il dispositivo di spiegazione del SYSCON è essenzialmente un dispositivo a traccia: esso registra tutte le regole di produzione che sono state eseguite e le condizioni che hanno condotto a ciò. Un inconveniente della realizzazione attuale è che la traccia conserva molto vocabolario dell'OPS5, ed è quindi difficile da leggere. Si sta provvedendo alle modifiche per produrre la spiegazione in lingua corrente. Un altro inconveniente è che essa non può rispondere a domande circa la configurazione finale; ma si spera che possa raggiungere anche questa capacità.

8. Sviluppo del Sistema.

Nello sviluppo di un sistema esperto, il compito che richiede più tempo è generalmente quello di costituire la base di conoscenza, perchè esso richiede di formalizzare un campo di esperienze; gli esperti umani che non riflettono molto sulla loro competenza, sanno molte cose che non sanno di sapere. Noi siamo stati più fortunati della maggior parte di coloro che debbono sviluppare un sistema esperto, perchè la configurazione di un mainframe ha sempre richiesto una accurata documentazione, e perchè il nostro esperto umano, Dale Mallek della Large Computer Product Division, ha cooperato con entusiasmo al progetto. Dall'inizio alla fine, lo sviluppo del SYSCON ha richiesto solo quattro mesi.

Le informazioni necessarie furono acquisite soprattutto dalla documentazione dei prodotti e mediante interviste con Dale Mallek. La documentazione ha fornito l'informazione dettagliata necessaria per la configurazione a basso livello e le interviste si sono concentrate sulle regole di produzione.

Il processo di configurazione al massimo livello è stato completato e messo a punto prima di ogni altro processo a livello inferiore. Ciò ha permesso una immediata verifica della interfaccia di utente e del funzionamento del sistema.

I procedimenti di livello inferiore furono verificati ponendo al SYSCON diversi problemi ipotetici, comprendenti due problemi tipici e due problemi di "caso peggiore". Questi erano dati da specificazioni difficili ma tuttavia trattabili, quale un sistema con quattro processori principali ed una sola unità di entrata-uscita. Le risposte del SYSCON a questi problemi vennero poi paragonate alle configurazioni ottenute manualmente. In una seconda serie di prove, si chiese al SYSCON di configurare tutti i DPS 90 già consegnati ai clienti, e i risultati furono paragonati alle configurazioni manuali. Al primo tentativo, al SYSCON non riuscì di fornire configurazioni completamente corrette per nessuna delle specificazioni; ma gli errori rilevati permisero di identificare i difetti del sistema e correggerli.

Il SYSCON intraprese a configurare DPS 90 sette mesi fa. Da allora ne ha configurato 32, e tutte le configurazioni funzionarono fin dall'inizio.

9. Conclusione

Per la maggior parte, l'approccio e il linguaggio scelti risultarono essere adatti al problema. È tuttavia insorta una difficoltà a lungo termine, quella di integrare il sistema esperto con i sistemi tradizionali esistenti. Se il SYSCON potesse essere collegato con altri sistemi di software e basi di dati, come quelli che contengono le informazioni circa i mainframe dei clienti, i prezzi dei prodotti, i dati circa le prestazioni, potrebbe essere usato per migliorare i mainframe esistenti. Uno degli ostacoli a tale integrazione è il linguaggio in cui è scritto. Il SYSCON fu sviluppato usando un elaboratore fornito del sistema operativo Multics. Benchè questo permettesse di realizzare e mettere a punto con efficienza il SYSCON, la maggior parte dei sistemi di software e di basi di dati con cui dovrebbe essere integrato il SYSCON operano su mainframe usanti il sistema GCOS che generalmente non ammette OPS5. Ora siamo di fronte al compito di convertire OPS5 perchè possa funzionare con GCOS.

Il SYSCON si è dimostrato utile, non solo per configurare il DPS 90, ma anche in un senso più ampio. Nessuno sa realmente fin dove possa arrivare la tecnologia dei sistemi esperti. Benchè si sappia per certo che molti problemi interessanti siano fuori della loro portata, è anche vero che essi permettono la soluzione di molti problemi refrattari alla tecnologia tradizionale. SYSCON aggiunge un altro complesso problema all'elenco di questi.

10. Bibliografia

[1] A. BARR and E.A. FEIGENBAUM, *The Handbook of Artificial Intelligence,* William Kaufman Inc., 1981.

[2] F. HAYES-ROTH, D.B. LENAT and D.A. WATERMAN, *Building Expert Systems,* Addison-Wesley, 1983.

[3] N.J. NILSSON, *Principles of Artificial Intelligence,* Tioga Publishing, 1980.

[4] E.A. RICH, *Artificial Intelligence,* Mc-Graw-Hill, 1983.

[5] H.W. WINSTON, *Artificial Intelligence* (2nd. edition), Addison-Wesley, 1984.

La compressione dei dati

LEONARDO FELICIAN
ALESSANDRA GENTILI

Istituto di Matematica, Informatica e Sistemistica
Università di Udine

1. Presentazione del problema

Oggetto di questo studio sono le tecniche di compressione dei dati nel trasferimento degli stessi tra memorie di massa diverse per capacità o utilizzo.

Il termine "compressione di dati" indica un processo reversibile che viene utilizzato per ridurre lo spazio necessario alla memorizzazione di informazioni. Ciò è reso possibile dall'impiego di diverse tecniche che solitamente sono raggruppate in due classi: quelle che fanno riferimento al contenuto logico dei dati, e quelle che ne sono indipendenti ed hanno quindi un campo di applicazione più vasto.

In genere le prime portano a risultati migliori, poichè, prendendo in esame un determinato tipo di dati per volta, sono in grado di sfruttarne le particolarità. Una buona conoscenza di ciò che si vuol comprimere costituisce, in questo caso, un presupposto fondamentale per il raggiungimento di risultati significativi.

Non sempre è però possibile avere informazioni sui dati, anzi accade più frequentemente che di essi non si conosca quasi nulla. In tale situazione si possono applicare tecniche più generali appartenenti alla seconda classe. Tra esse la più nota e adottata è quella che sfrutta l'algoritmo di Huffman [Huff. 52] e ricodifica ogni simbolo in base alla sua probabilità di occorrenza.

Una stringa compressa con questo metodo non può essere direttamente letta o elaborata dal computer, poichè essa non avrebbe significato nella rappresentazione classica, e deve perciò essere decodificata. Il processo di decodificazione, detto anche decompressione, è strettamente legato al metodo di compressione usato; infatti compressione e decompressione applicate in sequenza devono costituire una operazione nulla (NO-OP), che non apporta modifiche ai dati. Questa caratteristica di reversibilità distingue i metodi di compressione da quelli di compattamento, che pur eliminando solo le ridondanze, in genere, non consentono di ripristinare il dato nella sua forma originaria.

Va ricordato che le tecniche di compressione dati, oltre a rivestire un notevole interesse teorico, sono di applicazione comune in svariati campi dell'Informatica.

Un campo in cui è impossibile pensare di operare senza l'ampio utilizzo di tecniche di compressione (in questo caso di tipo Huffman modificato), è la memorizzazione delle immagini. Una pagina di documento in formato A4 (testo o grafico senza distinzione, giacchè l'informazione viene memorizzata a livello di significante e non di significato, identificando cioè soltanto i punti "neri" da quelli "bianchi") richiederebbe circa 200 Kbyte per essere memorizzata così come viene scandita dagli appositi lettori ottici (scanner). La compressione di questa stringa di dati permette però di ridurre all'incirca di un fattore 10 lo spazio richiesto, memorizzando una pagina di documento in circa 20-30 Kbyte (comunque molto di più di una pagina di testo memorizzata a livello di caratteri in circa 2 Kbyte).

Nonostante l'enorme impiego di spazio sulle memorie di massa, la tecnica di memorizzazione delle immagini è oggi qualcosa di più di un prototipo grazie anche alla disponibilità dei dischi ottici, ma non può essere implementata senza efficienti algoritmi di compressione. Un altro campo di applicazione, meno recente ma non meno importante, delle tecniche che sono studiate in questo lavoro riguarda la trasmissione di dati, dove in questo modo si riscontrano notevoli vantaggi quali la riduzione di tempi e costi di trasmissione e della possibilità che si verifichino errori (poichè questa probabilità è proporzionale alla quantità di dati trasmessi). I dati compressi inoltre sono di difficile interpretazione per chi non ne conosca il codice e quindi danno un'ampia garanzia di riservatezza delle informazioni (crittografia).

Per tornare al trasferimento di dati da una memoria di massa all'altra, va detto che non esistono nè problemi di errori causati da disturbi di linea nè problemi di riservatezza, ma una valida ragione per l'uso della compressione è semplicemente il risparmio di spazio.

Sono molte le situazioni in cui ciò è utile sia per memorie di massa in linea sia per quelle fuori linea.

La compressione per memorie in linea è spesso applicata a dati permanenti, presenti in quantità superiore al limite ritenuto accettabile. Se non può essere aumentato lo spazio disponibile, è evidente che una alternativa è ridurre quello occupato, comprimendo alcuni archivi: questa operazione è nota come "shrink". La scelta di questi archivi deve essere fatta in modo da non appesantire troppo i tempi di elaborazione a causa di continue compressioni e decompressioni: lo spazio risparmiato si paga infatti in termini di tempo di accesso. Sono preferiti pertanto gli archivi di consultazione meno frequente: è così possibile contenere il numero di decompressioni necessarie per rendere leggibile l'archivio e di compressioni di aggiornamento dopo ogni modifica. Comunque per gli archivi compressi in linea si presume che le letture siano molto più numerose degli aggiornamenti e, perciò, soprattutto la velocità di decompressione risulta essere un parametro importante per valutare le prestazioni del metodo.

La compressione di dati su memorie fuori linea è motivata invece o dall'esigenza di copiare, per ragioni di sicurezza, delle intere unità di memoria (copie di back-up), o dalla necessità di scaricare dalla memoria in linea degli archivi di dati che non siano stati usati di recente, ma di cui non si voglia perdere il contenuto (dump di archivi). In questo caso la decompressione ha un ruolo poco rilevante rispetto alla compressione, poichè si comprime prevedendo di non dover decomprimere se non in circostanze eccezionali.

2. Definizioni di base

Prima di esporre i metodi di compressione è necessario introdurre alcune definizioni prese dalla teoria dell'informazione.

Innanzitutto si definisce rapporto di compressione la frazione

E[N]/E[L]

dove E[X] indica il valore medio della variabile aleatoria X e le due variabili L e N sono le lunghezze delle parole del codice (le parole del codice sono le codifiche assegnate ai simboli) e, rispettivamente, delle sequenze primarie [Long. 80] (cioè della rappresentazione iniziale dei simboli). Per la compressione dei dati sulle memorie di massa E[N] è uguale al numero di bit del codice usato dall'elaboratore (normalmente 8 bit). In ogni applicazione di compressione si tende a massimizzare il precedente rapporto e perciò deve essere reso minimo il denominatore o la media delle lunghezze dei caratteri compressi. Inoltre, poichè ciò sarà utile per l'algoritmo di Huffman che costituisce uno dei temi centrali del lavoro, si riportano le principali distinzioni tra vari tipi di codice.

Si definisce codice a lunghezza variabile un codice per il quale le lunghezze $l(i)$ delle parole non siano tutte uguali tra loro. In caso contrario, cioè se la lunghezza delle parole è la stessa per ogni $l(i)$, il codice è detto a lunghezza fissa o codice blocco.

Con codice a blocchi si intende invece un codice che permette la codificazione simbolo per simbolo e non obbliga ad attendere la fine della stringa o messaggio.

Si definisce codice singolare un codice non iniettivo, tale cioè che a più simboli corrisponda una unica parola di codice o che esistano parole di codice uguali.

Tra i codici non singolari si distinguono quelli univocamente decifrabili e quelli non univocamente decifrabili. I primi hanno la proprietà di permettere, data una stringa di codifica, una e una sola possibile decodificazione. Siano ad esempio A=01 B=010 C=1 D=10 le parole di un codice non singolare; se ora si deve decodificare la stringa 010101 esistono quattro diverse possibilità:

AAA oppure ABC oppure BCA oppure BDC

e quindi il codice è non univocamente decifrabile.

Si definisce infine prefisso di una parola di codice, di lunghezza $l(i)$, la sequenza composta dalle sue prime $l(j)>l(i)$ cifre. Se le parole di un codice univocamente decifrabile sono tali che nessuna di esse è prefisso di un'altra parola allora il codice si dice a prefisso o codice istantaneo. Questo tipo di codici hanno in fase di decodificazione le stesse caratteristiche che i codici a blocchi presentano in fase di codificazione, cioè non necessitano della conoscenza dell'intero messaggio per poter iniziare la decodificazione dello stesso.

3. Compressione dei caratteri ripetuti

Esaminando archivi di dati, accade frequentemente di rilevare sequenze di caratteri (stringhe) che si ripetono. Spesso gli archivi contenenti dati numerici presentano sequenze di zeri e gli archivi di testo numerosi spazi bianchi contigui.

Queste stringhe possono essere ridotte a due oppure tre caratteri soltanto se si utilizza il seguente accorgimento: al posto della sequenza di caratteri si sostituiscono le informazioni di:

marca di soppress.	lungh. stringa	carattere ripet.

La marca di soppressione non deve necessariamente occupare lo spazio di un byte, ma può essere codificata anche con uno o due bit.

Nel caso del codice EBCDIC (ad otto bit), molte configurazioni esadecimali che nonhanno rappresentazione grafica si possono ritenere non significative [Mart. 77]. Parte di queste hanno il bit in posizione 1 uguale a 0 (le posizioni sono numerate da 0 a 7), cioè si sintetizzano come X0XXXXXX dove X può assumere indifferentemente valore 0 o 1. In altri casi i caratteri non significativi si configurano con i primi due bit uguali a "00".

Dunque si può assegnare alla marca di soppressione la codifica "00" di due bit ed i restanti sei bit possono essere utilizzati per la rappresentazione binaria della lunghezza della stringa, cioè n se n sono i caratteri in sequenza (con n<=31). In questo modo sono sufficienti due byte per la compressione.

Altri elaboratori però utilizzano quasi tutte le configurazioni del codice (a causa prevalentemente della definizione di simboli grafici speciali a video, caratteri di controllo per le stampanti, etc.) non permettendo quindi di codificare con un solo byte la marca di soppressione e la lunghezza della stringa.

Per tali elaboratori la compressione avviene in tre byte: uno per la marca, uno per la lunghezza e uno per il carattere.

Con sei bit si possono rappresentare numeri interi da 0 a 63 e con un intero byte da 0 a 255: perciò nel primo caso in due byte e nel secondo in tre si possono ridurre stringhe di lunghezza fino a 63 e, rispettivamente, 255 caratteri uguali.

Per gli archivi di testo a volte risulta conveniente utilizzare dei byte, normalmente non significativi, per codificare separatamente le ripetizioni di due, tre, quattro, etc., spazi bianchi contigui che si presentano con apprezzabile frequenza [Wied. 77].

La compressione dei caratteri ripetuti usata per proprio conto ha trovato vasto campo di applicazione soprattutto nella gestione di archivi di tipo testuale, ma combinata con altri metodi trova largo impiego in molte applicazioni sui dati.

4. Algoritmo di Huffman

Questo metodo è stato introdotto da David Huffman nel 1952 [Huff.52] e inizialmente rivolto al campo della trasmissione dati. La sua validità si estende oggi a tutte le applicazioni di compressione che tendono a ridurre le ridondanze di codice.

Il metodo consiste nella costruzione di un nuovo codice con codifiche di lunghezza variabile, che assegna ad ogni simbolo una codifica più o meno lunga in rapporto alla sua probabilità di occorrenza.

Il codice costruito gode di due importanti proprietà:

- è univocamente decifrabile, cioè non esistono due messaggi diversi con la stessa codifica;
- è istantaneo perché, noto l'inizio della codifica, non sono necessarie altre informazioni per procedere alla decodificazione del messaggio.

In altre parole, se

$x(1), x(2), \ldots x(n),$

sono i simboli usati, ordinati in modo tale che per le rispettive probabilità

$p(1), p(2), \ldots p(n),$

sussista la relazione

$p(1) \geq p(2) \geq \ldots \geq p(n),$

la proprietà della codifica di Huffman stabilisce che le lunghezze $l(i)$ delle codifiche

$c(1), c(2), \ldots c(n-1), c(n)$

siano

$l(1) \leq l(2) \leq \ldots \leq l(n-1) = l(n).$

Si può dimostrare che il codice ottimale ha le seguenti proprietà:

1. rende minima la lunghezza media delle codifiche definita come

$$\langle L \rangle = \sum_{i=1}^{n} p(i)\, l(i);$$

2. $l(n-1)=l(n)$ ed almeno una delle codifiche di lunghezza maggiore differisce da $c(n)$ solo per l'ultima cifra;
3. ogni stringa di lunghezza $l(n)$ è una codifica valida oppure contiene una codifica in uno dei suoi prefissi.

5. Metodo "Huffman modificato"

Il metodo detto di "Huffman modificato" [Gall.78] è stato studiato per superare le difficoltà imposte dal non adeguamento di un testo alla statistica di probabilità, permettendo, durante il processo di codificazione (runtime), di adattare le probabilità a priori alle frequenze effettivamente riscontrate nel messaggio in elaborazione e di aggiornare le codifiche. Inizialmente si dispone di frequenze e di codice standard, quest'ultimo costruito con il metodo classico di Huffman e contenuto in uno speciale albero binario implementato a lista. Per un alfabeto di n simboli vi sono n−1 elementi della lista o nodi dell'albero.

Il contatore delle occorrenze di un simbolo viene incrementato ogni volta che si incontra il simbolo stesso, così come vengono incrementati i contatori intermedi che si trovano sul cammino tra la foglia in esame e la radice.

Per una migliore utilizzazione del metodo si vogliono sfruttare quelle caratteristiche di località dei simboli che si riscontrano negli archivi, cioè si intende valorizzare le variazioni di frequenza – spesso sostanziali – che i simboli hanno tra parti diverse di uno stesso archivio. Questo si può ottenere se viene data un'importanza maggiore agli ultimi simboli letti e codificati rispetto ai primi, attraverso periodici ridimensionamenti dei contatori. In pratica ogni N simboli letti dall'archivio da codificare, i contatori subiscono una moltiplicazione per un fissato coefficiente $0<a<1$. Dalla scelta di N e di a dipende la velocità con cui le stime delle occorrenze cambiano il loro ordinamento. Un valore alto di N non conduce ad un buon impiego della tecnica, ma un valore troppo basso rende l'applicazione estremamente lenta sia per le continue moltiplicazioni sia per le eccessive modifiche. Altrettanto si può dire per il coefficiente a, poiché più si avvicina ad 1 meno sono ridotte le differenze tra i contatori e meno possibilità di variazioni ci sono; viceversa un valore troppo prossimo allo 0 implica frequenti variazioni.

Si stima che $a=1/2$ sia un buon coefficiente perché rende semplice la divisione e permette di assumere un valore sufficientemente alto di N.

L'adattamento delle codifiche alle frequenze testate e così calcolate avviene tramite confronto di contatori e scambio di puntatori all'interno dell'albero.

6. Rappresentazione unica dei dati

La rappresentazione unica dei dati fa parte (come i metodi successivi) di quell'insieme di tecniche che sono state precedentemente definite dipendenti dalla struttura o dal contenuto dei dati e che non sono l'oggetto principale di questo scritto, ma che vengono comunque illustrate per completezza di esposizione.

Il metodo sfrutta la caratteristica di molti archivi di avere dei campi nelle registrazioni che possono assumere un limitato numero di valori diversi. Se per esempio nel tracciato record di un archivio si inserisce un campo contente i giorni della settimana, è evidente che i diversi valori sono al più sette e quindi si può definire un rango per tali valori.

Si può applicare la compressione a questi dati costruendo per ogni rango un nuovo archivio in cui ad ogni diverso valore corrisponde una codifica binaria.

La stessa codifica può essere riportata non solo all'interno di un archivio, ma in tutti quelli in cui compare un campo con la stessa definizione, in sostituzione del valore a cui è stata assegnata.

In pratica, se N sono i possibili valori che un certo attributo può assumere, sono sufficienti

$n=[\log N]$

bit per ogni codifica [Young80], dove le parentesi denotano il minore intero non inferiore all'argomento (parte intera). Ogni valore infatti può essere identificato univocamente da un numero intero tra 1 ed N, che in rappresentazione binaria richiede n cifre.

Il rapporto di compressione per l'attributo è 1/n (in cui 1 è la lunghezza massima in bit che un valore tra quelli considerati può raggiungere) dato che negli archivi si sostituiscono n bit agli 1 presenti.

Il metodo risulta conveniente se il numero N dei diversi valori dell'attributo è sostanzialmente inferiore al numero totale delle registrazioni in cui compare l'attributo, poiché alla riduzione dello spazio negli archivi esistenti si deve aggiungere quello necessario alla costruzione dell'archivio di corrispondenza codice-valore.

7. Cenni sulla compressione di dati ordinati

Si vuole accennare ora ad un altro metodo dipendente dal contenuto dei dati e di grande utilità nel caso di archivi ordinati.

La tecnica si basa sull'ipotesi che la probabilità del ripetersi del valore di un campo sia molto maggiore della probabilità che due dati contigui presentino valori completamente diversi.

Questo è ad esempio cio che si verifica nelle guide telefoniche e in archivi anagrafici di notevoli dimensioni, in cui molti cognomi si ripetono più volte e numerose sono anche le variazioni dei soli caratteri terminali.

Si può in queste situazioni applicare un metodo di compressione che confronta ogni dato con il precedente e ne indica le differenze.

Ad esempio viene adottato il seguente codice [Alb.83]:

0000 il dato è identico al precedente
1000 differisce per l'ultimo carattere
1001 differisce per gli ultimi due caratteri
1010 differisce per gli ultimi tre caratteri
1011 differisce per gli ultimi quattro caratteri
11(n) nessuna relazione con il precedente ed è lungo n caratteri

e si codifica ogni dato con il byte iniziale diviso in due parti: il primo semibyte contiene una di queste codifiche; il secondo semibyte è posto a Hex "0" nei primi cinque casi, mentre nell'ultimo caso concatenato al terzo e quarto bit del primo semibyte costituisce la rappresentazione binaria del numero n. Nei byte successivi al primo si riportano le eventuali differenze.

Questo esempio di compressione presuppone una grande maggioranza di ripetizioni, poiché codificando con lo 0 in prima posizione solo il caso di un dato identico a 1 precedente, favorisce la distinzione di un dato uguale al suo diretto predecessore da uno disuguale, anziché riconoscere se le differenze risiedono – per esempio – negli ultimi due o tre caratteri. In altre parole se dal test del primo bit si identifica uno 0, non sono necessarie altre operazioni per ricavare il dato (noto che sia il dato precedente); invece nel caso in cui il bit risulti uguale ad 1 si devono considerare anche i bit successivi.

Se le ipotesi su un archivio prevedono una frequenza diversa da quella riportata per i dati ripetuti e per le differenze dei caratteri terminali tra i dati contigui, si può adattare il codice precedente. La modifica mirerà ad ottimizzare il numero medio di test.

8. Compressione per indici

Un'altra tecnica di compressione ampiamente impiegata riguarda gli archivi organizzati ad indici, per i quali è molto importante ridurre lo spazio riservato all'indice per poterlo contenere tutto o gran parte in memoria centrale, evitando ripetute operazioni di lettura su memoria di massa e velocizzando così il processo di ricerca di un elemento.

L'indice solitamente è composto da una chiave e da un puntatore che permette di desumere la localizzazione nel target dell'archivio in cui la registrazione è scritta con i suoi attributi.

La funzione dell'indice è di permettere la distinzione di un elemento dall'altro per mezzo del confronto tra le chiavi, perciò si può ritenere ridondante ogni parte della chiave che si ripete per più elementi o che non sia oggetto di confronto. Su questo principio si basano due delle tecniche più comuni [Wagn.73], che agiscono l'una sull'estremo sinistro e l'altra su quello destro della chiave

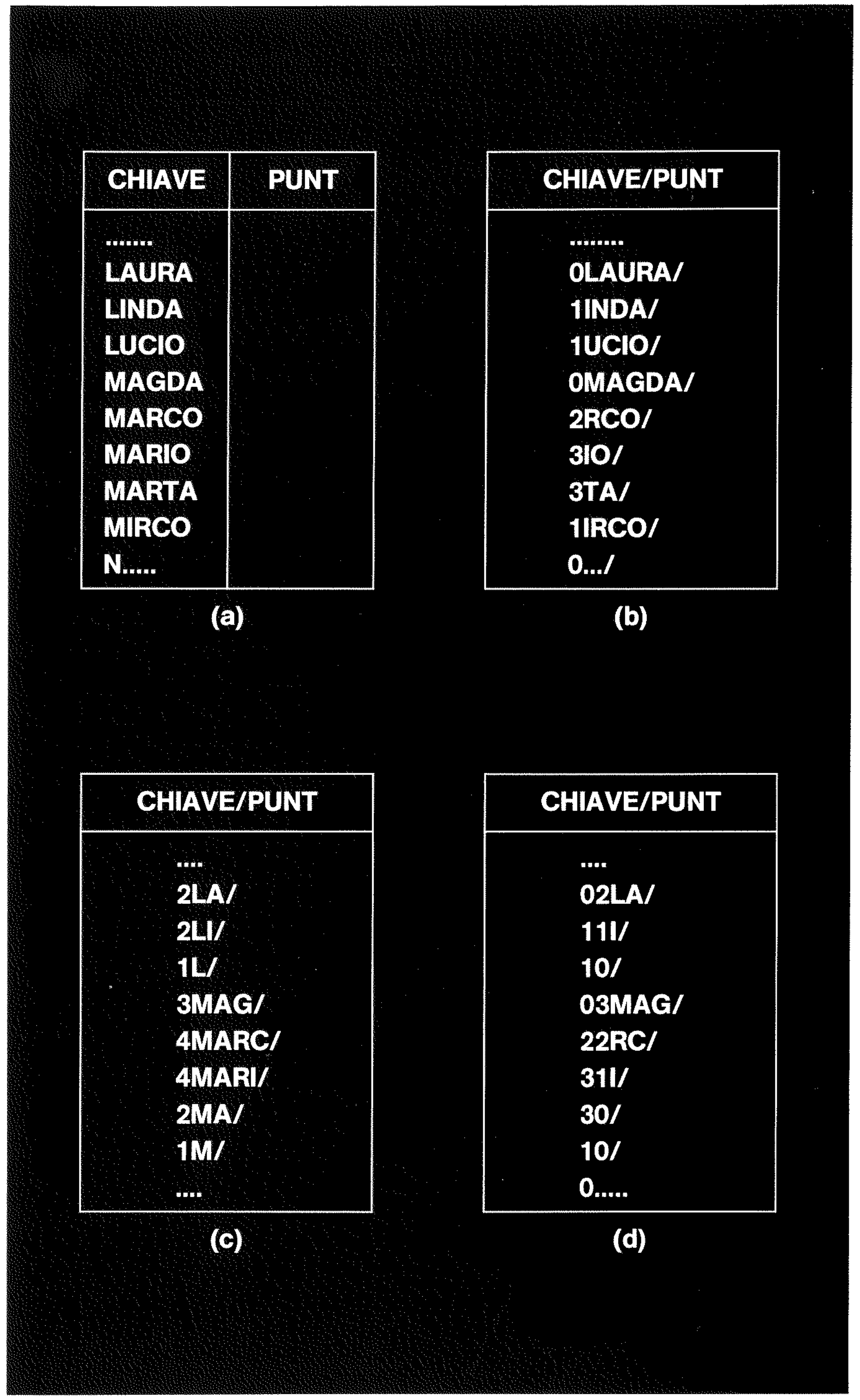
Per chiarezza di esposizione si discute un esempio di chiavi alfabetiche, ma il metodo è di validità generale: si abbiano dunque chiavi, contenenti nomi propri, di lunghezza fissata uguale a cinque e siano esse in ordine crescente come in tabella (a) (si trascurano i puntatori perché ininfluenti).

La differenza tra MARCO e MARIO consiste negli ultimi due caratteri, mentre i tre iniziali sono uguali. Il metodo di compressione sulla parte sinistra della chiave, chiamato front key compression, sfrutta queste ripetizioni degli stessi caratteri, all'inizio delle chiavi, per sostituirli con una cifra. Nel caso di MARCO e MARIO, il secondo si comprime in 3IO, poiché i primi tre caratteri sono gli stessi della chiave precedente. La tabella (a) si riduce così alla tabella (b).

Un altro metodo è la compressione sulla parte destra della chiave (rear key compression).

Questa tecnica elimina tutti quei caratteri della chiave che non sono utili per la ricerca di un elemento dell'indice, cioè quei caratteri finali

Fig. 1 - Esempi di tecniche di compressione:
(a) indice originario; (b) "front key compression"; (c) "rear key compression"; (d) "front" e "rear key compression".



che non sono oggetto di confronto tra una chiave e la successiva.

Poiché LAURA e LINDA si distinguono già dall'esame delle prime due lettere, è sufficiente scrivere LA per LAURA; così dal confronto di LINDA e LUCIO si riduce la prima ad LI. Per quanto riguarda MAGDA e MARCO, esse si differenziano per il terzo carattere così si scrive MAG anziché MAGDA, ma non si può scrivere MAR per MARCO perché esiste anche MARIO con gli stessi tre caratteri; dunque MARCO si comprime in MARC.

Inoltre essendo variabile il numero di caratteri per ogni chiave è necessario specificarlo all'inizio della chiave stessa.

Per l'esempio di tabella (a) si ottiene l'indice di tabella (c), ed usando entrambi i metodi quello di tabella (d).

È il caso di precisare, comunque, che in base alle definizioni di compressione e di compattamento date in precedenza è più esatto ritenere questi metodi tecniche di compattamento anziché di compressione, poiché la ricostruzione dell'indice è possibile soltanto dal confronto con l'intera registrazione memorizzata nel target dell'archivio.

Pur presentando seri inconvenienti in caso di separazione dei nodi dell'albero contenente l'indice a seguito di inserimenti (la front e la rear compression hanno senso solo all'interno di un blocco, altrimenti richiederebbero più accessi di I/O per comprendere il significato di una chiave), va detto che tecniche di compressione indici sono ampiamente adottate al fine di migliorare le prestazioni degli accessi.

9. Bibliografia

[1] A. ALBANO, *Organizzazione e Gestione di Archivi Integrati di Dati,* Università di Pisa, 1983.

[2] G. BRACCHI, G. MARTELLA, G. PELAGATTI, *Tecniche di Organizzazione degli Archivi,* ISEDI, 1982

[3] C. DATE, *An Introduction to Database Systems,* Addison Wesley, 1981

[4] DAVISSON GRAY, *Data Compression,* Dowden Hutchinson & Ross Inc.

[5] R. Gallager, *"Variation on a Theme by Huffman",* IEEE Transaction on Information Theory, no. 6, 1978

[6] HANSON, *Design of Computer Data Files,* Pitman, 1982

[7] E. HOROWITZ, S. SAHNI, *Fundamentals of Data Structures,* Pitman, 1977

[8] E. HOROWITZ, S. SAHNI, *Fundamentals of Computer Algorithms,* Computer Science Press, 1978.

[9] D. HUFFMAN, *"A Method for the Construction of the Minimum Redundancy Codes",* Proc. IRE 40, 1952

[10] E. KNUTH, *The Art of Computer Programming,* vol. 3, Sorting and Searching, Addison Wesley, 1973

[11] G. LONGO, *Teoria dell'informazione,* Boringhieri, 1980

[12] J. MARTIN, *Computer Database Organization,* Prentice Hall, 1977

[13] E. PAGE, L. WILSON, *Rappresentazione delle Strutture Informative e Algoritmi di Base,* ISEDI

[14] A. VITERBI, J. OMURA, *Principles of Digital Communications and Coding,* Mc Graw Hill, 1979

[15] R. Wagner, *"Indexing design Consideration",* IBM System Journal vol. 12, no. 4, 1973.

[16] G. Wiederhold, *Database Design,* Mc Graw Hill, 1977

La flotta ha ancora bisogno della nave ammiraglia?

ANTONIO PIZZARELLO

Honeywell
Large Computer Products Division
Phoenix, Arizona (USA)

Un tempo esistevano solo mainframe racchiusi sia entro la barriera fisica delle pareti del centro di calcolo, che entro la barriera burocratica di una organizzazione gestita separatamente. Molti utenti potenziali erano tenuti lontani dalle apparecchiature di calcolo da queste due barriere. Successivamente, dapprima in modo piuttosto inefficiente, il terminale remoto e poi, molto più efficientemente il personal computer ed il mini, portarono la potenza di calcolo più vicino all'end-user potenziale.

Molte persone sono state così coinvolte nella rivoluzione del computer. Tra di loro si devono annoverare soprattutto gli ingegneri, ma anche le segretarie, gli economisti, gli addetti alla produzione ed un gran numero di addetti a mansioni d'ufficio, come gli agenti di borsa e gli agenti assicurativi. Questo fenomeno è stato chiamato la rivoluzione del computer, o la rivoluzione del micro, e si manifestò in modo molto evidente circa cinque anni fa. Questa rivoluzione generò molte nuove idee e prodotti; noi le dobbiamo, ad esempio, i sistemi distribuiti ed i terminali grafici. La rivoluzione del micro diede origine anche ad una serie di previsioni circa la sparizione del mainframe dal mondo dei computers.

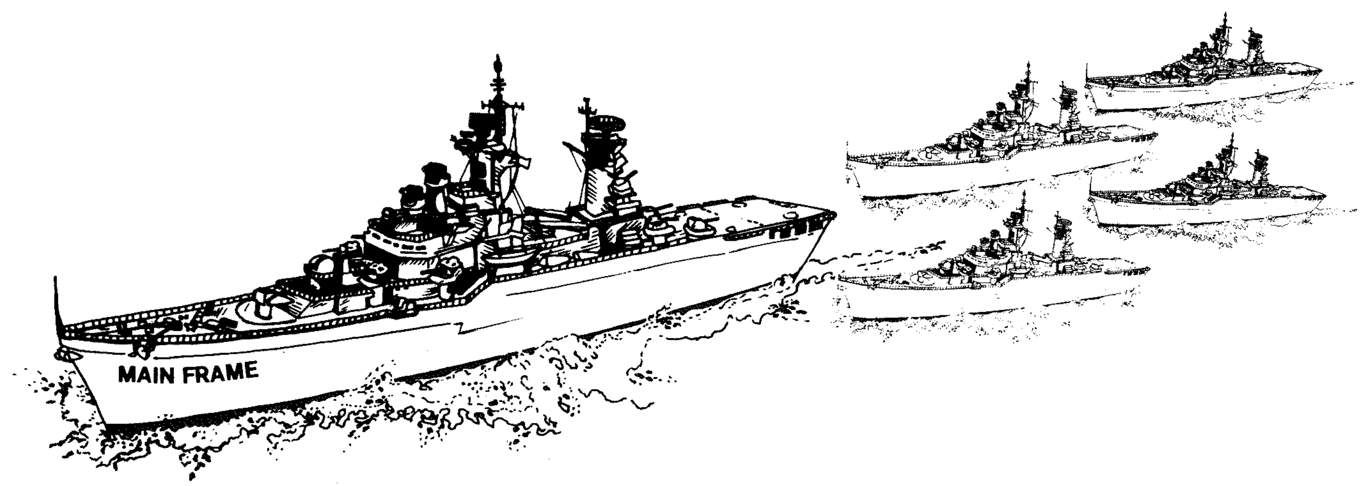
Ma se si ammette che il mainframe stia veramente sparendo, alcuni fenomeni cui oggi assistiamo parrebbero piuttosto strani. Durante gli scorsi cinque anni, cioè nel pieno della rivoluzione del computer, la IBM ha aumentato il volume dei MIPS spediti sotto forma di grossi mainframes al tasso del 40% all'anno. (Il MIPS, ovvero Milione di Istruzioni per Secondo, è una misura comune della potenza dei computers).

Il tasso di incremento è limitato solo dalla capacità produttiva dell'IBM. Il mercato ne domanda di più, forse molto di più.

Un altro fatto sorprendente è che il costo del MIPS su un mainframe rimane almeno il doppio di quello su ogni altro tipo di computer. Nonostante questo costo più elevato, un gran numero di persone, presumibilmente sane di mente, continuano a comperare ad un ritmo crescente.

Può essere utile notare che il numero di MIPS dell'unità centrale di elaborazione (CPU) di un mainframe varia da 6 a 10 circa, mentre tutte le altre fonti di potenza di calcolo forniscono solo da 0.5 a 2 MIPS per CPU. Tuttavia alcuni dicono che la maggiore potenza della CPU in un mainframe non è significativa in quanto è possibile collegare insieme molte CPU meno potenti, creando un sistema distribuito che fornisce il numero desiderato di MIPS.

La computerizzazione dell'ingegneria è un banco di prova interessante che ci può aiutare a capire perchè si comperano ancora mainframes. Secondo l'IBM i MIPS addizionali



di mainframe venduti negli ultimi anni sono stati utilizzati per applicazioni scientifiche e di ingegneria. Molte classi di applicazioni, come ad esempio l'elaborazione testi, sono uscite tout-court dal mercato dei mainframes ed hanno cominciato a spostarsi verso i sistemi basati su microprocessore.

Ma se andiamo a ritroso ancora di qualche anno, constatiamo che anche moltissimi PC e mini furono acquistati per applicazioni scientifiche e di ingegneria. In effetti le grandi società aerospaziali hanno sostituito nei loro laboratori di progettazione i tavoli da disegno con microcomputer.

I calcoli di ingegneria, che ancora cinque anni fa erano eseguiti a mano (gli ingegneri erano restii a chiedere ai guru del mainframe di eseguire i calcoli per loro), vennero trasferiti rapidamente sul nuovo strumento.

Gli ingegneri della Honeywell si trovano tuttora in questa fase. Gran parte di essi usano dei PC, che consentono loro di eseguire elettronicamente oggi ciò che eseguivano a mano ieri.

Come si spiega questo successo iniziale dei micro in questo mercato? Nella maggior parte della comunità degli ingegneri la rivoluzione dei micro produsse l'effetto di eliminare la barriera del centro di calcolo. L'introduzione dei packages applicativi (in particolare i packages grafici) in combinazione con la disponibilità di MIPS in quantità abbastanza elevata, rese possibile per la prima volta, l'uso esteso del calcolo per prevedere il comportamento dei prodotti e per progettarli meglio. Più precisamente, il micro non era solo uno strumento che consentiva agli ingegneri di migliorare la loro produttività velocizzando quello che già facevano; esso apriva in realtà nuove vie per rendere più efficiente il loro lavoro. Tutte quelle magnifiche equazioni dei libri di testo divennero utili improvvisamente poichè potevano ora essere risolte dal micro senza eccessiva fatica e in un tempo ragionevolmente breve.

Non senza qualche sorpresa, queste equazioni, una volta applicate all'oggetto fisico reale anzichè ad una sua versione notevolmente semplificata (cui il calcolo manuale aveva costretto gli ingegneri) si dimostrarono veramente capaci di predire il comportamento dei prodotti, risparmiando così innumerevoli prove, riducendo sostanzialmente i costi ed accorciando i tempi. Risultati simili furono constatati (poco dopo) dagli economisti, dagli analisti finanziari, dai biochimici e da altri.

Quanto detto finora presuppone la fondatezza delle previsioni sulla decadenza del mainframe. Sappiamo tuttavia che le previsioni si sono rivelate errate. Perchè? Ci sono stati molti buoni motivi; anzitutto i micro non erano all'altezza delle analisi più sofisticate che gli ingegneri volevano eseguire. L'analisi ingegneristica esige per lo più che la forma di un pezzo meccanico venga esplosa in molte sezioni, per le quali si eseguono poi i calcoli. L'analisi riuscirà tanto più precisa quanto maggiore sarà il numero di sezioni, ma il tempo richiesto cresce con la potenza quadrata o cubica del numero di sezioni. Ci si può pertanto trovare di fronte all'impossibilità pratica di effettuare calcoli più raffinati su una macchina in grado di seguire solo mezzo milione di istruzioni al secondo. Inoltre, con l'aumentare del numero di sezioni aumentano anche le dimensioni dei risultati dei calcoli. Su una macchina a 32 bit, gli errori introdotti dal troncamento dei valori possono essere così rilevanti da rendere l'analisi priva di qualsiasi valore.

Se si tratta di un micro, è necessario ricorrere a espedienti di software per poter registrare un valore in due o tre parole da 32 bit. Tali espedienti comportano tuttavia un appesantimento del calcolo, con conseguente notevole rallentamento di un micro con doppia o tripla precisione aritmetica. I mainframe attuali, che realizzano aritmetica a doppia precisione tramite hardware, presentano a questo riguardo un enorme vantaggio in fatto di precisione.

Un altro tipo di lavoro in cui i micro hanno rivelato i loro limiti consiste nella gestione di grandi archivi di informazione. Gli ingegneri hanno sempre fatto uso di archivi. Essi creano enormi archivi di cianografie che vengono archiviate e gestite normalmente con costi elevati. Anche altri documenti, come specifiche, piani di prove, risultati di prove, standards ecc. devono essere archiviati e gestiti. Anche in questi casi i micro ebbero molto successo all'inizio e incrementarono il business dell'archiviazione automatica. Ma presto emersero molti problemi. Troppo spesso le stesse informazioni venivano registrate in due o più macchine diverse ed aggiornate da persone diverse; il risultato fu una... micro-torre di Babele che non poteva conciliarsi con la mentalità ordinata propria dell'ingegnere. Così il mainframe che è in grado di gestire data base di dimensioni enormi, divenne all'improvviso estremamente attraente. I dati ingegneristici subiscono aggiornamenti molto più frequenti rispetto ad altri campi, come quello dell'analisi finanziaria, e le modifiche apportate hanno potenzialmente ripercussioni molto più vaste.

Una operazione di prelievo da una banca può comportare la modifica del valore di un campo nel data base.

Per contro, un ingegnere che ha trovato una migliore soluzione ad un problema di progetto, ha probabilmente l'esigenza di eliminare un intero sottoparagrafo della struttura di un prodotto o di modificare pesantemente la struttura stessa. A questo riguardo il mainframe dispone di molti canali di input/output che consentono veloci ed estese modifiche dei dati.

Ci si possono aspettare a questo punto le domande dello scettico: «Se si usassero i sistemi distribuiti?» e «Se si inserissero questi micro in una rete, in modo da coordinare le loro attività?» Queste domande hanno avuto risposta da due tipi di esperti.

Da una parte i teorici, che dopo essere saltati sul carro del "micro con distribuzione" arrivarono alla conclusione che dopo tutto non si trattava di un'idea così grandiosa.

D'altro canto, gli stessi utilizzatori scoprirono di preferire il servizio di un mainframe piuttosto che affidare

i loro dati ai sistemi distribuiti disponibili.

I teorici furono indotti dalla popolarità del microprocessore a studiare algoritmi che consentissero la soluzione efficiente di un problema usando diversi elaboratori separati ma cooperanti. Fin dall'inizio si incontrarono difficoltà impreviste; in particolare, si constatò che un numero sorprendente di problemi non potevano essere spezzati rapidamente in sotto-problemi che potessero essere risolti in parallelo. Per problemi con queste caratteristiche si definì una classe e gli algoritmi per la loro soluzione furono chiamati "greedy algorithms" o "algoritmi ingordi".

La velocità di soluzione di un problema di questa classe dipende dalla velocità della CPU, non dal numero di CPU. Molti problemi comuni di elaborazione dati e di analisi numerica rientrano in questa classe.

Un'altra scoperta curiosa fu che una rete di computer chiamati a risolvere insieme un problema comune è intrinsecamente meno affidabile di un computer singolo, anche se la rete ha un numero molto maggiore di parti duplicate. Ciò è dovuto al fatto che i micro godono di un "grado di copertura" molto inferiore. (Grado di copertura è la probabilità che il sistema possa essere ripristinato con successo e rapidità dopo una "caduta"). Un basso grado di copertura si riflette in modo bizzarro, ma comunque confermato dai fatti, sulla affidabilità, cioè la probabilità che il sistema funzioni correttamente per un determinato intervallo di tempo. Se il grado di copertura di un sistema scende anche solo di 0,001%, l'affidabilità del sistema può diminuire di oltre il 10%. Inoltre, se il grado di copertura è basso e si aumenta la ridondanza, l'affidabilità del sistema può in effetti diminuire anzichè aumentare.

Tutto quanto sopra può sembrare solo una fumosità dei teorici. In questo caso purtroppo l'esperienza pratica conferma la teoria. Ad esempio, occorsero più di sette anni di lavoro alla IBM per rendere ripristinabile un sistema IBM formato da due computer. E la procedura di ripristino è talmente delicata e complessa che è molto dubbio che il sistema possa essere consegnato a utenti qualsiasi. Esperienze spiacevoli di questo tipo si sono avute anche in casa nostra.

Gli utenti hanno rifiutato le reti di microprocessori per altre ragioni. Con l'evoluzione delle elaborazioni da forme semplici a forme sempre più complesse, cominciarono a manifestarsi errori sia nei packages standard che nel software sviluppato dall'utente, con conseguenti distruzioni dei dati. Il risultato è stato che l'utente di micro è ora interessato ad un fornitore in grado di offrire una garanzia più elevata di integrità dei dati e di affidabilità del sistema, di quanto possano offrire i sistemi basati sui micro.

Tutte le considerazioni sopra esposte stanno orientando l'utente di micro verso il mainframe. Ciò non significa che sia in corso una migrazione degli utenti e dei loro programmi verso il mainframe, abbandonando il già diletto micro.

La complessità del mainframe impedirà che esso diventi lo strumento principale dello end-user. Ogni timido tentativo di padroneggiare questa complessità da parte di gente che ha altri pensieri oltre i computer, è destinato al fallimento. Di conseguenza, il passaggio al mainframe è in realtà semplicemente ciò che l'IBM chiama "trasporto di funzioni" dalla periferia al centro.

Quali funzioni sono e saranno trasferite? Secondo i suggerimenti emersi dalla nostra discussione, sono essenzialmente due: "macinazione" di numeri ed integrazione del data base. Affichè queste funzioni vengano eseguite adeguatamente è necessario disporre di una macchina dotata di aritmetica veloce, di un gran numero di canali di input/output, di assoluta integrità dei dati e di estrema disponibilità, in altre parole si ha bisogno di un mainframe. Questo non supporterà funzionalità di end-user come la stampa di qualità, la grafica ed altre interfacce raffinate, poichè esse vengono meglio trattate su microprocessore. Questi diventeranno dei satelliti del mainframe, cioè i punti d'accesso dell'utente finale alla potenza di calcolo di cui non ha visibilità.

Lungi dallo sparire, il mainframe diventerà più potente. La potenza della CPU dei mainframe sta aumentando ed aumenterà ancora più nel prossimo futuro. Aumenterà anche il numero di CPU per sistema, benchè non fino ai numeri astronomici suggeriti da alcune architetture d'oggigiorno. La potenza di CPU effettivamente usata diventerà tuttavia percentualmente minore, in quanto si sceglie la potenza di CPU per il carico di punta (almeno per quanto riguarda i clienti IBM) e non per il carico medio. Attualmente i grossi sistemi IBM sono utilizzati mediamente all'80% della potenza totale.

La tendenza è verso un livello di utilizzo inferiore. Si prevede che grossi problemi da risolvere in tempi accettabili faranno precipitare il livello di utilizzo fino al 20%, di cui una frazione importante (forse la metà) riguarderà l'esecuzione del software di sistema, soprattutto a causa dell'esigenza di una elevata disponibilità e quindi di un grado di copertura molto prossimo a uno.

A questo punto chi fosse ancora scettico potrebbe chiedere se un potente macinatore di numeri insieme con una macchina per data base, come il sistema prodotto dalla Teradata in California, potrebbe sostituire il mainframe. La risposta è no, per due motivi: il primo deriva dalla natura delle applicazioni, il secondo è tecnico.

Dal punto di vista applicativo il ruolo del mainframe consiste nella memorizzazione ed elaborazione di tutte le informazioni su scala aziendale (o divisionale).

Ci sono e ci saranno sempre funzioni centralizzate di elaborazione dati. Allo stesso modo che alcune funzioni di una società multi-divisionale sono gestite meglio dalla sede centrale, alcune funzioni di un sistema di computer sono gestite nel modo più naturale ed efficiente da un potente computer centralizzato. Le macchine per data base e i puri macinatori di numeri non sono adatti per questo ruolo, anche se possono svolgere, ed effettivamente svolgeranno, un ruolo ausiliario di supporto.

La motivazione tecnica per includere un mainframe centrale è il ripristino della rete. Ho accennato prima al ruolo che svolge il grado di copertura nel determinare l'affidabilità. Il solo modo per aumentare il grado di copertura di un sistema consiste nel ridurre la quantità di informazioni perse quando si verifica una caduta di sistema. Ciò si può ottenere memorizzando una copia delle informazioni necessarie per ripristinare il sistema, in uno stato adatto in un data base centralizzato; tali informazioni potrebbero essere usate per ricostruire le informazioni memorizzate localmente che sono andate perse in una caduta di sistema.

Tenuto conto della situazione delineata, non dovrebbe sorprendere che l'IBM abbia appena annunciato un nuovo e più potente mainframe (il multiprocessore Sierra, di 20-25 MIPS). Il nostro accordo con la NEC, da cui acquistiamo l'hardware di mainframe, ci ha posto in una posizione migliore per affrontare la prossima fase nell'evoluzione dei grandi sistemi. Noi siamo attualmente in grado di collegare potenti workstation ad un mainframe GCOS8 e stiamo portando avanti altri progetti più sofisticati per accoppiare la potenza del mainframe con il volto amico del micro.

Collana dei Quaderni di Informatica

La collana di testi "Quaderni di Informatica" rappresenta la proiezione sul piano librario della presente rivista. La direzione scientifica della collana (che è edita da F. Angeli) è del "Centro Studi Informatica e Automazione" della Honeywell I.S.I. A titolo di rassegna sintetica, riproduciamo qui di seguito le copertine dei volumi finora usciti, che coprono alcune delle più importanti aree dell'informatica.