Honeywell
Honeywell Information Systems Italia
Via Laboratori Olivetti - Pregnana Milanese

UNIVERSITA'
DEGLI STUDI DI MILANO
ISTITUTO DI CIBERNETICA
Via Viotti, 5 - Milano

19/20

NOTE DI SOFTWARE

UNIVERSITA'
DEGLI STUDI DI MILANO
ISTITUTO DI CIBERNETICA

Honeywell
Honeywell Information Systems Italia

Note di software.
è un bollettino trimestrale, edito a cura del Centro di Ricerca e Progettazione della Honeywell Information Systems Italia e dell'Istituto di Cibernetica dell'Università di Milano.

Note di software.
intende constituire uno strumento per la rapida e informale circolazioni di informazioni, idee ed esperienze nell'area della tecnologia del software. In particolare, intende stimolare il dibattito sulle metodologie orientate alla diminuzione dela rapporto costo/qualità dei programmi, e sugli strumenti atti ad automatizzare l'attività di produzione del software in ogni suo aspetto.

La collaborazione a **Note di software** è libera.
In particolare, si sollecitano contributi nella forma di brevi monografie e bibliografie essenziali commentate sui seguenti temi: tecniche e strumenti per la programmazione strutturata e modulare, metodologie e strumenti per il testing e il debugging dei programmi, documentazione del software, aspetti organizzativi e gestionali nella costruzione di sistemi software di grandi dimensioni.

Chi desiderasse pubblicare con contributo è pregato di prendere contatto con il Comitato di Redazione (HISI - Via ai Laboratori Olivetti - Pregnana Milanese), o di inviare direttamente il dattiloscritto in triplice copia.
I contributi non dovrebbero superare le quattromila parole.

I lavori accettati per la pubblicazione non vengono revisionati da un comitato tecnico. Pertanto, ogni contributo pubblicato riflette le opinioni dei suoi autori, e non necessariamente quelle del Comitato di Redazione.
Eventuali revisioni saranno effettuate di comune accordo fra il Comitato di Redazione e i proponenti.

Note di software.
viene distribuito a chi ne faccia richiesta alla redazione.

Direttore responsabile: Paolo Ghelfi
Comitato di Redazione: Roberto Polillo - Wanda Anselmetti - Stefania Bandini
Redazione: Franco Soresini

19/20

Maggio 1983

Indice:

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

NOTE DI SOFTWARE

QUESTO NUMERO...

La prima parte di questo numero doppio di NOTE DI SOFTWARE è dedicata ad una serie di lavori sul LIBRO ELETTRONICO.
L'occasione di questa raccolta di articoli è stata offerta dal Convegno "Editoria e Comunicazione Totale" promosso dalla Fondazione Mondadori (Milano, 1-IV-1982), nell'ambito del quale era stata allestita una mostra centrata sul problema delle conseguenze delle nuove tecnologie sulla comunicazione e che aveva visto la presentazione dei lavori qui pubblicati.
NOTE DI SOFTWARE ringrazia Bruna Zonta e Angela Masia per in coordinamento redazionale dei lavori qui presenti.

La seconda parte vede come peimo contributo tecnico un lavoro di S. Mussi riguardante le linee fondamentali dell'architettura di un programma interattivo per la simulazione funzionale discreta di un sistema real-time, corredato da un esempio.

Nel secondo lavoro, G. Castelli e M. Fiorentini presentano la descrizione di tecniche usate nell'implementazione di una famiglia di differenti compilatori PASCAL su tre differenti macchine; si tratta dell'ottimizzazione di un solo front end con tre diversi generatori di codice in un contesto di minimizzazione di sforzo della manutenzione in ambiente industriale.

Il terzo lavoro, di L. D'Amico, ha come argomento la descrizione di un package di funzioni di libreria per il linguaggio di programmazione C, implementato sotto la versione 7 del sistema operativo UNIX.

La redazione

VERSO IL LIBRO ELETTRONICO

La attenzione alle applicazioni del word processing è stata finora rivolta in misura rilevante alla stesura di testi. Poca attenzione è stata posta alla lettura degli stessi, nella ipotesi che il testo una volta prodotto finisca con l'essere diffuso per via cartacea.

Ma chi è abituato ad impiegare il wp in fase di preparazione di testi e documenti sa bene che anche la loro lettura può avvalersi delle possibilità offerte dai sistemi di calcolo e non ha difficoltà ad immaginare opzioni sofisticate con cui testi potrebbero venir letti o scorsi efficacemente al fine di estrarne informazioni.

I lavori:

- *Piccoli elaboratori e comunicazione, di G. Degli Antoni*
- *Dal libro ai sussidi elettronici, di F. Soresini*
- *Tecniche CAI, di A. Andronico*
- *McLuhan, l'elettronica, il libro, di B. Zonta*
- *UNIBOOK, di M. Casazza*
- *Verso il manuale elettronico, di E. Bertolotti*
- *Un esperimento di trasporto da un libro ordinario a uno elettronico, di L. Agliati*
- *TEX, un sistema di composizione per la stampa di testi scientifici e tecnici, di G. Canzii e M. Gualzetti*
- *METAFONT, un linguaggio per la definizione sintetica di elementi di composizione grafica, di G. Canzii e A. Pilenga*
- *TEXLIDE, un sistema grafico per la generazione, la manipolazione e l'archiviazione di trasparenti per lavagna luminosa, di A. Avogadro, D. Marini, E. Paizis*
- *Paginazione per un sistema di consultazione di testi, di P. Grandi*

raccolti in questo numero di NOTE DI SOFTWARE si pongono nella ottica di considerare i sistemi di word processing alla stregua di strumenti di lettura, quasi a costituire una sorta di LIBRO ELETTRONICO. Da tale punto di vista gli strumenti suggeriti o che possono derivarne sono direttamente orientati alla comunicazione umana e ad una attività produttiva quale la predisposizione di documentazione.

Le ragioni per effettuare analisi come queste sono forse molteplici e certo tempestive: basti ricordare che la riduzione dei costi della tecnologia microelettronica probabilmente farà sì che per non poche attività elaboratori portatili sostituiranno i libri di testo.

Molte modifiche alla comunicazione ne potranno risultare e la stessa espressione letteraria potrà avvalersi di nuove forme.

Il considerare quindi da parte degli autori i sistemi di OGGI come strumenti di comunicazione è stimolante per successive analisi e certamente utile per immaginare evoluzioni future.

G. Degli Antoni

PICCOLI ELABORATORI E COMUNICAZIONE

Giovanni Degli Antoni
Istituto di Cibernetica, Università degli Studi di Milano

1. PREMESSA

Scopo della presente nota è quello di proporre una discussione sulle modalità di comunicazione rese possibili dall'avvento della microelettronica.

La nota non rappresenta pertanto nè un punto sullo stato dell'arte nè un'indagine di mercato. Riflette in parte le opinioni dello scrivente come punto di vista sullo sviluppo delle applicazioni degli elaboratori ma soprattutto vuole segnalare punti di vista anche nella consapevolezza che altri dovrebbero venir talvolta considerati.
La discussione è rivolta a coloro che intendono contribuire allo sviluppo di sistemi per applicazioni nelle comunicazioni.

2. LA COMUNICAZIONE UMANA VIA ELABORATORI

Poichè la nota intende discutere l'influenza degli elaboratori sulla comunicazione umana, una breve discussione si impone alfine di delimitare il campo.
Una rigorosa definizione di cosa si intenda per *"comunicazione umana"* è impossibile. Noi qui intenderemo con quel termine quegli atti di comunicazione che sono tradizionalmente associati alla produzione, alla archiviazione, alla manutenzione, alla distribuzione ed all'impiego di documenti cartacei.
Tali documenti costituiscono il supporto di informazioni oggetto dell'attività del comunicare. Sono ad esempio documenti i libri di testo, le lettere di corrispondenza, note tecniche, eccetera.

Il presente lavoro è stato realizzato nell'ambito delle ricerche condotte in cooperazione fra l'Istituto di Cibernetica e la Honeywell Information Systems Italia sui problemi di comunicazione (CP Project).

Questo lavoro è stato pubblicato su "BIAS 1980" Microelettronica *Tecnologie e sistemi della microelettronica,* Convegno internazionale, Fiera di Milano 4-5 giugno 1980 (FAST).

La comunicazione umana comporta che in vari momenti i documenti siano scritti con un linguaggio (anche tecnico) comprensibile da umani.

I problemi connessi con l'utilizzo di supporti elettronici nella comunicazione umana sono numerosi e spesso complessi.

Inoltre le conseguenze della diffusione dell'elettronica in relazione alla riduzione dei costi sembrano suggerire che accanto a problemi tecnologici si debbano necessariamente considerare i problemi organizzativi, quelli psicologici e quelli sociali che l'utilizzo di supporti elettronici nella comunicazione umana comporta.

La tecnologia ha già realizzato molte componenti orientate alla comunicazione umana. Lo scopo di questa nota non è dunque di considerare tali realizzazioni ma piuttosto di fornire un punto di vista che permetta di esplorare l'estensione delle applicazioni e problemi associati a tali estensioni.
La discussione non investirà problemi tecnici di dettaglio ma discuterà piuttosto aspetti applicativi e sistemistici.

3. UNA PRIMA DISCUSSIONE: LA PREPARAZIONE DI TESTI

Lo sviluppo della microelettronica ha solamente in parte influenzato lo sviluppo della comunicazione attraverso testi scritti e limitatamente al momento della loro produzione.

Gli elementi attuali della tecnologia elettronica nello sviluppo di testi sono editors e sistemi di fotocomposizione nonchè stampanti a laser elettrostatiche o altro associate a sistemi di stampa (eventualmente a duplicatori). La riduzione dei costi di questi componenti o la disponibilità di servizi orientati al loro utilizzo potrà cambiare un po' il panorama ed avere una notevole influenza almeno nelle aree in cui i testi vengono prodotti in numero limitato. Infatti in tali aree potrebbe risultare più economica la preparazione di copie a richiesta piuttosto che la stampa di un numero prefissato di copie.

Ciò comporta numerosi problemi in relazione alla probabile difficoltà di realizzare uno standard secondo cui il testo viene codificato soprattutto nell'ipotesi che i sistemi di stampa siano realizzati da costruttori diversi. Analizzeremo successivamente a grandi linee tali problemi.

Un altro aspetto riguarderà la organizzazione del lavoro: la riduzione dei costi di sistemi di editors o, se si preferisce, l'evoluzione della macchina da scrivere potrebbe rendere possibile la distribuzione della preparazione di un manoscritto su supporto magnetico (o teletrasmesso) laddove il testo viene preparato. Se ciò può sembrare fantascientifico per la preparazione di romanzi, nel caso dello sviluppo di testi tecnici e scientifici è già certamente realtà presso varie organizzazioni.

Un aspetto non trascurabile delle conseguenze possibili dell'elettronica nella produzione di testi è la possibilità di ottenere direttamente dalla casa editrice descrizioni bibliografiche orientate alla realizzazione di archivi come sottoprodotto della modalità di produzione riducendo così alquanto la difficoltà ed il costo per realizzare biblioteche da parte degli acquirenti.

4. TESTI E TRASMISSIONE DEI DATI

Qualora all'impiego di tecnologie produttive di tipo elettronico venga associata la trasmissione di dati attraverso reti, dotate di capacità di archivio, il panorama muta profondamente e si realizza un notevole conflitto (economico) fra il vantaggio di disporre la copia cartacea di un testo (a non elevata diffusione) e l'accesso a parti dello stesso attraverso una rete di trasmissioni dati.
Infatti la disponibilità su archivio di tutto il testo rende possibile la ricerca di informazioni sullo stesso rendendo talvolta inutile la disponibilità di tutto il testo nella sua versione cartacea.

Casi del tipo qui proposto sono già attuali nel campo delle informazioni disponibili da reti di elaboratori ed ottenibili anche in forma cartacea. È ben vero che la disponibilità della copia cartacea in tali casi può significare un mercato più ampio, ma è anche vero che l'utente che non utilizzerà a fondo certe informazioni potrebbe preferire un accesso saltuario all'acquisto di tutta un'opera (talvolta costosa come ad esempio nel caso di informazioni bibliografiche).
Se ad esempio lo sviluppo di articoli di giornali scientifici viene effettuato mediante reti di elaboratori, potranno comparire altre conseguenze, fra le quali un graduale abbandono del mezzo cartaceo ed il conseguente isolamento culturale da parte dei partecipanti a tale attività rispetto ad altri interessati alla stessa attività che non utilizzino la stessa rete.

Tale isolamento potrebbe ridursi con lo sviluppo delle reti di trasmissione dei dati, ma potrebbe anche rinforzarsi a causa dell'assenza di interfacce uniformi fra reti.

5. LA PORTABILITÀ DEI TESTI

La portabilità dei testi codificati assicura la possibilità ad utenti dotati di differenti attrezzature di utilizzare tali testi oltre a rendere possibile l'evoluzione delle attrezzature ad uno stesso utente, senza un eccessivo costo.

Il problema della portabilità ha molte dimensioni che vanno dalla natura del supporto, alla codificazione scelta fino alla forma definitiva del testo. L'ultimo punto è quello che ci preme qui sottolineare essendo tipico del settore che stiamo indagando. Gli altri problemi di portabilità indicati non differiscono da problemi simili incontrati per applicazioni differenti.

La possibilità di codificare un testo fornendo anche elementi che rendano possibile assegnare al testo codificato una forma cartacea implica l'individuazione di elementi specifici del testo e l'assegnazione a questi elementi di attributi (ad esempio la fonte del carattere).

Una proposta uniforme attualmente non esiste. Pur tuttavia una sorta di standard di fatto si sta diffondendo. È stato indicato con il termine di "generic coding" da parte dell'associazione GCCA (Graphic Communication Computer Association) che ne ha curato la diffusione. Si tratta di un procedimento che suggerisce di separare gli elementi testuali dalla loro immagine che riceveranno sul testo cartaceo. Inoltre viene suggerita l'individuazione di una serie di elementi di testo (titolo, paragrafo, ...). I codici associati a questi elementi vengono utilizzati per la definizione di un sistema gerarchico. Gli elementi del testo vengono così contrassegnati con un elemento del codice gerarchico. Nella fase di stampa a tali contrassegni verrà attribuito un valore (fonte, posizione, ecc.).
Si tratta quindi di una semplice standardizzazione di una contrassegnatura (mark-up) effettuata sulla base di ben noti aspetti relativi alla forma dei documenti cartacei.
I valori assegnati ai contrassegni possono essere modificati. In ogni caso i problemi di trasporto risultano modesti.

La proposta discussa non è la sola possibile. Infatti una volta accettato di separare il contenuto del testo dalla sua forma definitiva nel testo cartaceo si può, ad esempio, lasciare che un operatore interattivamente stabilisca la forma del testo prima della stampa. In questo caso si può evitare la contrassegnatura.

6. ARCHIVI REMOTI, ARCHIVI LOCALI

La storia dello sviluppo di elaboratori è costellata di

tentativi di impiego di concetti di economie di scala alfine di ridurre i costi, economizzare risorse, ...

Non è difficile osservare che molti di quei tentativi sono falliti o sono stati superati dalla riduzione dei costi. Sarebbe tuttavia superficiale ritenere che la situazione attuale, in ordine all'utilizzo di mezzi di calcolo, sia insoddisfacente.

Una valutazione accurata ed inoppugnabile della visione ottimistica qui sostenuta è al di fuori degli scopi della presente nota.
Tuttavia il lettore si chieda se i libri di testo delle biblioteche sono usati efficientemente.
È ben facile stabilire che la percentuale di utilizzo dei testi di una biblioteca non è certamente più elevata della percentuale di utilizzo dei mezzi di calcolo (piccoli o grandi) distribuiti nelle varie organizzazioni.
D'altra parte un utente è sempre uno specialista. Ed anche nel caso che costui decida di impiegare archivi remoti di dati, avrà molti vantaggi a disporre di una copia locale.

Ciò potrebbe comportare la necessità di sviluppare archivi locali specializzati in cui ricercare informazioni con tecniche più sofisticate e personalizzate alla sua attività.

Il modello proposto non è altro che la trasposizione del modello di prelievo di una *copia* (in prestito?, in acquisto?) di un testo da una biblioteca.

Le implicazioni e le possibilità sono molte:

- messa a punto di archivi specialistici con conseguente creazione di valore aggiunto legato alla riorganizzazione dei dati;

- facilità di accesso alle informazioni e ritrovamento a costo più basso di informazioni già reperite precedentemente;

- costruzione di archivi adatti alla misura dell'utente.

L'ultima discussione tende a sottolineare la possibilità di trasformare una parte dell'attività di ricerca (scientifica, tecnologica, di mercato, ecc.) in un prodotto.
Ma forse ciò che più interessa è che un'accurata organizzazione della raccolta di informazioni omogenee può contribuire allo sviluppo di archivi comuni.

7. COMUNICAZIONE E COMPUTAZIONE

In un testo tecnico compaiono spesso calcoli formali effettuabili con l'ausilio di elaboratori.
L'impiego di elaboratori per la comunicazione suggerisce che il testo possa non solo comunicare con il suo lettore umano, ma possa in parte essere interpretato direttamente dall'elaboratore che viene utilizzato per la sua comunicazione.

Ciò significa che l'utente dispone di un efficace strumento didattico anche "attivo" in grado di accettare una gamma di richieste di elaborazioni spiegate dal testo e suggerite dal lettore.

Questo punto di vista è del tutto coincidente con l'attitudine impiegata in ormai innumerevoli applicazioni delle tecniche CAI (Computer Aided Instruction).

Le tecniche CAI vengono sempre più impiegate in connessione con piccoli elaboratori ed una certa quantità di s/w didattico di buona qualità è commercialmente disponibile.

I problemi legati allo sviluppo di sistemi CAI personalizzati sono molti. Ne sottolineiamo alcuni con brevi discussioni:

- Dimensione degli archivi
  L'accesso ad archivi remoti per l'utilizzo di tecniche CAI di larga diffusione e non limitata ad ambiti specifici può essere costoso. Ciò può comportare l'opportunità di archivi locali. Il costo di tali archivi è ancora elevato se l'intero corso deve essere memorizzato su supporto magnetico. Ciò fa si che attualmente per applicazioni di basso costo ci sia spesso separazione fra testo di spiegazione e testo "attivo".
  La possibilità di disporre di supporti a disco di basso costo sembra poter migliorare la situazione per il caso di sistemi personali.
  Ovviamente non sussistono difficoltà nel caso di impiego di sistemi di media dimensione dotati di dischi di notevole capacità.

- Natura dell'interfaccia uomo-macchina
  La tecnologia ha sviluppato molte possibilità per lo sviluppo di interfacce semplici e veloci nella comunicazione uomo-macchina. Tuttavia nessuna ha ancora assunto il carattere di accettazione che il tipo di applicazione dovrebbe comportare in relazione al costo dello sviluppo di materiale didattico di qualità.
  Si può tuttavia ritenere che siano indispensabili mezzi grafici di comunicazione (diapositive o direttamente terminali grafici) oltre ad ausili di comunicazione che limitino la necessità di impiego della tastiera per accedere a informazioni (ad esempio con indicazione a dito, joy-stick, light-pen, ecc.).
  La comunicazione vocale (in uscita) ha qualche interesse e si presenta indispensabile nella comunicazione con handicappati.
  La comunicazione vocale in uscita potrebbe essere semplicemente ottenuta da un registratore a cassetta musicale controllato dall'elaboratore accanto al proiettore di diapositive.
  Va in ogni caso segnalata la necessità di una risposta pronta da parte dell'interfaccia se deve essere mantenuto l'interesse dell'utente durante il dialogo.
  Quest'ultima osservazione comporta che gli elaboratori personali nelle applicazioni didattiche siano dotati di notevole capacità di calcolo.

- Linguaggi per la stesura dei testi CAI
  Le tecniche CAI richiedono un notevole sforzo per la preparazione di materiale didattico. Questo viene solitamente preparato con linguaggi ad hoc o immerso sostanzialmente sotto forma di commenti in linguaggi di programmazione ordinari. Il caso più comune nel settore dei microsistemi è quello del linguaggio BASIC per cui alcune organizzazioni hanno curato vari livelli di standardizzazione corrispondenti ad altrettanti livelli di portabilità.
  È inutile sottolineare che sulla standardizzazione operano tendenze contrastanti.

- Trasporto dei testi CAI
  Uno dei mezzi attuali di trasporto verso piccoli sistemi è la cassetta magnetica. Tuttavia, come nel caso degli archivi, molto rimane da fare se anche il testo di accompagnamento viene letto mediante un elaboratore a causa della necessità di supporti di più elevata capacità delle attuali cassette o dei dischi flessibili (dell'ordine di Mbyte).

8. LA RAPPRESENTAZIONE DELLA CONOSCENZA

Il passaggio dalla semplice comunicazione dei testi alla loro integrazione con sofisticate tecniche di ricerca delle informazioni e di problem-solving richiede lo sviluppo di opportune metodologie per la rappresentazione della conoscenza.

In questo settore l'intelligenza artificiale ha notevolmente contribuito, sebbene l'enfasi dell'intelligenza artificiale sia stata solitamente posta sulla difficoltà del problema da risolvere piuttosto che sugli aspetti di comunicazione.

Dunque lo sviluppo della piccola informatica richiede che si utilizzino tecniche di problem solving in funzione della comunicazione soprattutto onde semplificare il dialogo per renderlo più attraente e quindi per estrarre informazioni sia dalla conoscenza memorizzata che dagli atti di comunicazione dell'utente.

Ne segue che la rappresentazione della conoscenza è uno dei problemi dello sviluppo di semplici ma sofisticati sistemi CAI. Le modalità per la rappresentazione della conoscenza sono molte. Pressochè tutte utilizzano reti in cui "elementi" di conoscenza sono connessi con opportune "relazioni": La ricerca di informazioni sarà pertanto legata alla percorrenza di tali reti da parte di opportuni enti.

La possibilità di organizzare la conoscenza in modo sistematico può permettere la costruzione di "archivi di conoscenza", di archivi cioè in cui sono rappresentate leggi ed ipotesi.

Lo studio della rappresentazione della conoscenza ha estremo interesse nella medicina.
Infatti, in questo settore, abbandonata l'illusione di sistemi omnicomprensivi, sembra possibile impiegare sistemi di modesta dimensione per rappresentare un settore specifico (ad esempio la cardiologia).

Fra alcune metodologie per la rappresentazione della conoscenza ne vanno segnalate alcune semplicissime che impiegano un archivio di frasi in linguaggio naturale per caratterizzare un settore di conoscenza non formalizzato.

9. ANALISI DI UN CASO: LA MEDICINA

A scopo analitico e suggestivo vogliamo portare alle sue estreme conseguenze un caso: quello della medicina.
La discussione non vuole essere realistica sebbene tutti gli spunti contenuti siano realizzabili già ora (ed esistono sistemi che rispecchiano i punti indicati, anche se non globalmente).

D'altra parte l'analisi di qualsiasi aspetto della comunicazione in medicina coinvolge molte parti del sistema medico. Così occorrerà stabilire:

a) quale conoscenza si intende comunicare (esempio: la cardiologia)
b) chi la rappresenta in quale modo formale (i ricercatori?) e chi la diffonde
c) per quale utente viene rappresentata (paziente, medico,...)
d) che uso ne fa l'utente e con che mezzo di calcolo
e) che uso viene fatto dei dialoghi dell'utente (anamnesi, risultati dell'analisi, ecc.) da parte di chi?

Noi non risponderemo a nessuno dei punti indicati. Ci preme solo sottolineare che la disponibilità di mezzi modesti rende possibile (punto e)) un dialogo anche a più riprese fra utente finale (medico o paziente) ed organizzazione che ha predisposto il dialogo su supporto eseguibile.

Tutto ciò implica lo sviluppo di personale che abbia la capacità di rappresentare in una forma utilizzabile la conoscenza accanto ad organizzazioni capaci di gestire il tutto.

10. VERSO IL LIBRO ELETTRONICO

L'utopia a cui le osservazioni tendono è ora chiara: il libro elettronico. Un mezzo che qualcuno produce. Contiene informazione essenzialmente attiva destinata a certi utenti.

L'utente lo legge e lo può annotare. Le annotazioni possono essere distribuite. Le annotazioni possono essere lette. Decisioni possono essere quindi prese per una rilettura o la lettura di un altro libro elettronico e comunque per valutazione.

Chi ha proposto per primo molti aspetti di questa utopia con il nome di Dynabook è A. Kay della Xerox Research Center che ha indicato anche le tappe intermedie da realizzare con piccoli e potenti sistemi attuali.

Queste note vogliono sottolineare che l'utopia "Libro Elettronico" richiede tuttavia molti sforzi e non solo di informatica se le conoscenze da comunicare debbono essere di qualche valore.

11. CONCLUSIONI

La discussione di problemi della vastità indicata non poteva che essere condotta a livelli di astrazione molto elevati e non è certo completa neppure a quei livelli.

Il lettore potrà tuttavia convenire che la migliorata facilità di impiego degli elaboratori e le mutate possibilità di calcolo impongono uno sforzo rivolto a chiarire nessi causali fra gli eventi associati allo sviluppo di applicazioni.
Tali nessi dipendono da molti fattori e sono suggeriti da inferenze sulla realtà attuale e/o da analogie.

DAL LIBRO AI SUSSIDI ELETTRONICI

Franco Soresini
Honeywell Information Systems Italia

1. INTRODUZIONE

La necessità di ricordare le conoscenze per tramandarle nel tempo ha determinato la creazione di diversi metodi per codificare l'informazione e/o rappresentare figurativamente la stessa. Codificati e rappresentazioni necessitano di un supporto.

Attraverso una evoluzione dei supporti, si giunge alla realizzazione del libro, dapprima manuscritto, poi a stampa.

Dopo secoli di incontrastata predominanza del libro, l'evolversi delle tecnologie ha portato alla realizzazione di altri tipi di supporto delle informazioni che possiamo definire "surrogati del libro".

Per facilitare la lettura (anche nel caso di non vedenti), così come l'ascolto (anche nel caso di deboli d'udito) e lo studio, sono stati realizzati metodi e dispositivi atti semplificare e rendere rapida la lettura quanto facile l'ascolto sfruttando anche i riflessi condizionati.

Infine, le "macchine per insegnare", dal "laboratorio linguistico" ai vari metodi e macchine per la "istruzione programmata", si sono sostituite all'insegnante ed al libro di testo per conseguire l'apprendimento per autoistruzione.

La necessità di ricordare le conoscenze per tramandarle nel tempo ha determinato la creazione di diversi metodi per codificare l'informazione e/o rappresentare figurativamente la stessa.

registrandola sopra adatto supporto, atto a memorizzarla, per tramandarla nel tempo.

Codificati e rappresentazioni necessitano di un supporto

Ne sono un esempio:

- supporti solidi tridimensionali permanenti
  - incisioni rupestri
  - lapidi scolpite
  - tavolette di creta
  - tavolette cerate
- supporti solidi tridimensionali labili
  - muro
  - lavagna
- supporti in folio naturali permanenti
  - papiro
  - pergamena
  - seta
- supporti in folio artificiali permanenti (*)
  - carta di stracci
  - carta di cellulosa

Attraverso una evoluzione dei supporti, si giunge alla realizzazione del libro, prima manoscritto, poi a stampa.

Prima che le parole venissero scritte e le immagini rappresentate, il ricordo della informazione era unicamente affidato alla *memoria* e tramandato a voce o affidato a "evocatori".

Nelle masse, le notizie importanti erano il solo tipo di informazione che veniva diffuso tramite banditori.

Essi, dopo aver richiamata l'attenzione con voci e/o suoni, leggevano le grida (o bando) solitamente scritte su un lungo foglio arrotolato.

Con la diminuzione dell'analfabetismo, i bandi, scritti e/o stampati, furono affissi quali avvisi, manifesti e notificazioni su appositi albi, o tabelle, situati in determinati siti.

La raccolta ordinata di tale materiale informativo, ha finito con il determinare il concetto di *volume,* riferito ad un insieme di fogli, scritti o stampati, facenti *testo.*

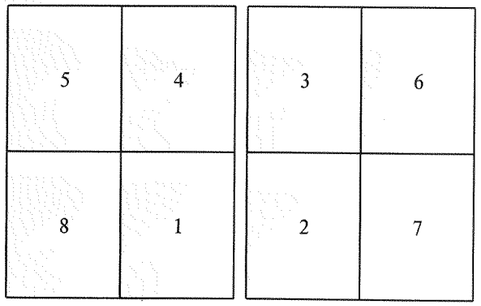
La riunione di tali fogli formanti volume ha costituito il *libro.*

2. IL LIBRO

- di testo
- atti
- vocabolari
- dizionari
- glossari
- enciclopedie
- collane e tematiche
- altri (abaco, abedecario, atlante, ecc.)

* Costituzione di un libro *
Il libro è costituito dalle pagine stampare in "quartino"

(*) Purtroppo, non tutti i tipi di carta, causa acidità (ptt) dei componenti



In relazione alle dimensioni del foglio utilizzato, si passa da 8 a 16 a 32 a 64 *facciate.*

Se il libro è di poche pagine assume il nome di: dispensa, estratto, fascicolo, monografia, opuscolo, quaderno.

* Legatura *
I quartini, cuciti, graffati, incollati di costa, formano un *volume.*
Il volume ha una *copertina* di carta spessa, cartoncino, cartone, plastica.
Anzichè la copertina il volume può avere la *legatura* di cartone ricoperto di pelle, plastica, pegamoide, tela, o altro.

* Assiemaggio *
Per raccogliere in volume, fogli, fascicoli, dispense, ecc., sciolti ed aggiornabili nel tempo, si usa una *cartella* con accessori di fissaggio: fermacampioni, vitoni, anelli, spirali o con accessori di contenimento a pressione: ganasce, costole elastiche, levette di compressione.
Un particolare tipo di assiemaggio di fogli mobili è l'*album.* Il sistema più semplice di assiemaggio di pagine o fascicoli sciolti è la *cartella con legacci.*

* Sovracopertina *
Sopra la copertina, la legatura o la cartella, può essere sovrapposta per protezione o indicazione (soprattutto cromatica) una *sovracopertina.*

* Astuccio *
Nel caso di volume di pregio, o per assiemare più volumi di una collana, si usa un *astuccio o custodia.*

* Costola *
Legatura, raccolta, sovracopertina (ed anche l'astuccio) hanno una *costola* il cui spessore dipende dal numero di pagine contenute.

* Risguardo *
È costituito da quelle pagine disposte fra la copertina ed il *frontespizio.* Nel caso di legatura, la prima di queste pagine vi è incollata.
Sul retro del *disguardo,* di faccia al frontespizio, vi può essere indicato il titolo della collana cui il libro appartiene.

* Frontespizio *
È la prima pagina del libro.

* Indicazioni *
Il frontespizio, la facciata anteriore di copertina, legatura, costola, portano le seguenti indicazioni:

- nominativo dell'autore o degli autori
- eventuali titoli personali o ente di lavoro
- titolo dell'opera ed eventualmente sottotitolo
- indicazione eventuale di tomo o parte
- edizione
- marchio editore e/o sua ragione
- luogo di pubblicazione e data.

* Parti di un libro *
Quando un'opera diviene ponderosa, o per ragioni di comodità d'uso, il testo viene suddiviso in *tomi* (detti anche volumi) e ciascuno di questi in *parti.* Tomi e parti si indicano con numerazione romana. In taluni casi, più volumi, anche indipendenti, costituiscono una *collana.* Le collane si evidenziano solitamente con particolari colori di copertina.

* Suddivisione del contenuto *
Si può suddividere nelle seguenti parti:
- presentazione
- prefazione
- introduzione

- testo vero e proprio
- conclusioni

* Suddivisione del testo *
Viene solitamente suddiviso in:
- capitoli
- sottocapitoli
- paragrafi
- sottoparagrafi

Il testo può venire compendiato da:
- figure
- tavole

La suddivisione si può evidenziare mediante la codificazione decimale.

Il testo può fare ricorso anche a:
- evidenziazioni, giocando sull'uso di caratteri diversi
- richiami, solitamente a margine
- note a piè pagina o fine capitolo
- bibliografia a piè pagina o fine capitolo
- appendici aggiuntive di complemento a fine volume

* Complementi *
A complemento di un testo si possono considerare:
- l'indice, quale elenco di tutti i titoli e sottotitoli, con il riferimento di pagina
- la tavola analitica di nomi e cose citate nel testo, con il riferimento a pagina
- il glossario di particolari termini usati nel testo
- il sommario succinto e sintetico del contenuto
- la scheda bibliografica, che è un sommario oltre che una bibliografia dell'autore

* Altri costituenti *
- Gli aggiornamenti, attuabili solo su libri non rilegati
- Gli inserti, eventuali brevi lavori a sè stanti (o a dispense) attinenti il libro cui sono allegati

* Mezzi di reperimento argomenti *
- la pagina, il suo numero progressivo in numeri arabi; in taluni casi in numeri arabi soltanto il testo, e in cifre romane le altre parti

* Immagini *
Se nel testo, sono *figure;* se fuori testo sono *tavole.* Sono costituite da:
- schizzi
- disegni
- incisioni
- illustrazioni in genere

Particolari tipi di immagini sono:
- tavole sinottiche
- flow
- diagrammi
- schemi
- monogrammi
- ecc.

Queste ultime hanno il compito soprattutto di sintetizzare gli argomenti.

Possono essere impiegati anche particolari tipi di tavole quali:
- carte topo-geografiche
- esplosioni
- sfilamenti
- sezioni
- ecc.

* Ausili *
Per dare risalto al testo, oltre alle immagini indicate, possono essere impiegate:
- stereoscopie
- stratificazioni
- tavole bugnate (in rilievo)
- tavole scomponibili
- tavole da ritagliare e/o ripiegare
- ecc.

Altri ausili al libro sono:
- griglie di evindeziazione
- sagome
- regoli
- rilevatori reattivi (di luce, di ph, ecc.)
- tavole campionarie (merceologiche)
- diapositive
- microfilm
- dischi

3. I SURROGATI DEL LIBRO

Dopo secoli di incontrastata predominanza del libro, l'evolversi delle tecnologie ha portato alla realizzazione di altri tipi di supporto per la informazione che possiamo definire "surrogati del libro".

Il surrogato più classico del libro stampato è il libro parlato, ossia il *fonografo* (1877).
Successivamente, dal "telegrafono" (1902) si è sviluppata la *registrazione magnetica.*
Parallelamente a questa, ha avuto pratica applicazione, come sonorizzazione cinematografica, la *registrazione fotoelettronica.*

Un altro surrogato del libro stampato è costituito dalla *proiezione:*

- fissa [epidiascopia, diapositive trasparenti, microfilm]
- in movimento [cinema, televisione] con o senza sonoro

Un metodo misto *audiovisivo* che utilizza diapositive e nastro magnetico, per la registrazione del suono e dei comandi, è costituito dal *film-strip.*

Per facilitare la lettura (anche nel caso di non vedenti), così come l'ascolto (anche nel caso di deboli di udito) e lo studio, sono stati realizzati metodi e dispositivi atti a semplificare e rendere rapida la lettura e a facilitare l'ascolto, sfruttando anche i riflessi condizionati.

Per permettere una lettura "per sommi capi" e non costringere il lettore a scorrere parola per parola, esistono ausili atti ad aumentare il potere di sintesi visiva (es. "reading rateometer", "reading accelerator", "lightening").

Un testo può anche essere letto con processi fotoelettrici. Il riconoscimento (cioè la lettura) di caratteri alfabetici e numerici stampati, scritti a mano in stampatello o in corsivo, è nato con l'"optophone". di Fournier D'Albe che traduceva per i ciechi i caratteri in suoni. Nel 1939 l'autore proponeva l'applicazione di un generatore di "voce artificiale" avvalendosi degli studi di analisi elettroacustica del linguaggio dovuti a Pastore e Gemelli.

Oggi perfezionati sistemi di analisi dei caratteri hanno permesso di realizzare le fronteggiatrici postali per la lettura del C.A.P.

Sintetizzatori di voce sono poi il mezzo di uscita idoneo per dar voce a queste macchine capaci così di leggere.

Un sistema intermedio, leggibile visivamente, ed in via magnetica, è costituito dai caratteri magnetici E138, CMC7, utilizzati sugli assegni, noti ormai da 20 anni.

Caratteri adatti ai non vedenti sono quelli dovuti a Braille (1829).

Utilizzante l'ascolto è un sistema per la ripetizione continua di un testo che, automaticamente, per riflesso condizionato, viene memorizzato durante il sonno e nel dormiveglia.

Non si può, fra gli altri, non menzionare l'"audifono stroboscopico" della SAFAR (1934) per rendere visibili i suoni ai deboli di udito.

A questa famiglia di oggetti, si può affiancare la "traduttrice" che, su display, o mediante sintetizzatore, si sostituisce al vocabolario; prima fra questa la Lexicon LK3000, e il "Parla tutto" adatto per i primi rudimenti linguistici.

Infine, le "macchine per insegnare", dal "laboratorio linguistico" ai vari tipi di metodi e di macchine per la "istruzione programmata", si sono sostituite all'insegnante e al libro di testo, per conseguire l'apprendimento in autoistruzione.

Alla base di queste macchine stanno le metodologie di "istruzione programmata".

4. L'ISTRUZIONE PROGRAMMATA

Il termine "ISTRUZIONE PROGRAMMATA" è forse il più usato per indicare un campo nuovo della metodologia dell'insegnamento.

Questa metodologia si basa su molti principi dell'insegnamento che sono ormai universalmente conosciuti ma che mai sono stati analizzati ed applicati secondo uno schema ricavato da vaste indagini psicologiche sul comportamento umano nell'apprendimento.

I primi studi, in questo senso, furono fatti verso il 1930 da alcuni educatori e da alcuni psicologi, ma soltanto dopo il secondo conflitto mondiale questi studi assunsero le proporzioni di una vasta ricerca organizzata.

Fra i primi ad occuparsene fu F.B. Skinner della Harvard University, il quale, assieme a numerosi collaboratori, preparò alcune sperimentazioni di questo nuovo tipo di insegnamento in scuole, università, istituti di ricerca, scuole aziendali e scuole militari.

La prima serie di documenti che descrivono l'uso pratico dei principi dell'I.P. è dovuta allo Skinner, che nel 1954, diede l'avvio ad una rivoluzione dell'insegnamento.

L'elemento essenziale negli studi fatti da Skinner, è il comportamento umano, "human behavior", inteso come insieme dei processi attraverso i quali l'uomo modifica il proprio comportamento (verbale e non verbale, ma comunque esteriormente osservabile e controllabile), sotto l'azione di stimoli esterni, anch'essi osservabili e misurabili.

Analizzando i risultati di alcuni esperimenti fatti sia sull'uomo che sugli animali, lo Skinner riuscì a definire il tipo di insegnamento nel quale il comportamento poteva essere portato ai risultati finali desiderati attraverso degli stadi intermedi.

In ogni stadio, il successo del metodo consiste nel creare uno stimolo al quale lo studente deve reagire in modo positivo, adattando il proprio comportamento alla nozione appresa.

L'effetto che lo stimolo produce trova riscontro nello stadio successivo.

Seguendo le indicazioni di Skinner, la United States Air Force, Office of Naval Research, Ford and Carnegie Foundation United States Office of Education fecero molti esperimenti. I risultati ottenuti furono incoraggianti per l'efficacia del metodo.
Da allora l'istruzione programmata ha "dilagato" sia per le innumerevoli applicazioni sia per l'elevato numero di specialisti che ad essa si sono dedicati per studiare nuovi metodi ed affinare quelli già esistenti.

L'istruzione programmata:

- permette all'allievo di lavorare da solo, al proprio ritmo, grazie ad un dispositivo appropriato.
- presenta all'allievo una materia in *sequenze ordinate.*
- ciascuna di queste sequenze ha lo scopo di insegnare all'allievo un *elemento semplice.*
- ogni sequenza termina con una *domanda* alla quale l'allievo deve rispondere.
- costui è *immediatamente avvertito* della validità della sua risposta
- permette il procedimento di *istruzione attiva*

Questa definizione è sufficientemente generale per potersi applicare a tutti i sistemi di istruzione programmata attuali.

- dai più semplici *quanto al metodo pedagogico* (programmi lineari di Skinner) ai più complessi, che presentano ad ogni allievo delle sequenze adatte al suo comportamento del momento (macchine di Gordon Pask, per esempio).
- dai più semplici, quanto al *dispositivo* (manuali lineari) ai più complessi (progetto Plato II per esempio, dove sono un calcolatore analogico ed un circuito chiuso di televisione che presentano le sequenze ad ogni allievo).
- da quelli che richiedono all'allievo *risposte scritte* (esempi di "Skip branching" del prof. Kay) a quelli che hanno dei sistemi di risposte a scelta forzata (esempio: programmi ramificati di Crowder).
- da quelli che si applicano ad *allievi isolati* (autotutor Mark II) a quelli che si applicano a *classi,* pur ponendo ad ogni allievo domande adatte (CLASS, indubbiamente il più ambizioso ed avanzato di tutti i sistemi attualmente allo studio).

L'istruzione programmata consente una *auto-istruzione,* nel senso che permette la sostituzione del professore con il suo dispositivo di presentazione.

Questa auto-istruzione ha dei punti in comune con l'istruzione che, da secoli, permette il *libro.* Ne differisce profondamente perchè si avvicina di più al maestro di quanto non faccia il libro tradizionale; questo, in realtà, è come un conferenziere *indifferente al suo uditorio.*

Nell'istruzione programmata, al contrario, l'allievo è *controllato dal dispositivo:* è interrogato, deve rispondere e procedere nel suo lavoro dopo aver capito gli argomenti precedenti.

Studiando su un libro tradizionale se ne possono girare le pagine. Nell'istruzione programmata no.

L'istruzione programmata non comprende unicamente "sequenze di insegnamento" ma anche:

- *sequenze di esercizi:* che permettono di applicare le nozioni acquisite
- *sequenze di prova:* che hanno lo scopo di controllare la acquisizione di una conoscenza chiave nel programma
- *sequenze di revisione:* che consistono in un richiamo delle nozioni acquisite.

La metodologia della "I.P." è la seguente.

a) *Programma lineare di SKINNER*

Lo Skinner osservò che se la trasformazione del ruolo del maestro è l'aspetto più spettacolare dell'istruzione programmata, i principi pedagogici che la stessa utilizza (e che potrebbero teoricamente essere utilizzati da parte di un precettore con un solo allievo, come accadeva per il "nobil signore" dei secoli scorsi) devono essere analoghi ai principi che Descartes enunciava nelle regole per la direzione dello spirito: "Tutto il metodo consiste nell'ordine e nella disposizione delle cose verso le quali è necessario puntare tutti gli sforzi del proprio spirito per scoprire qualche verità. *Lo seguiremo punto per punto, se riporteremo gradatamente le frasi oscure e complicate a frasi più semplici, e se partendo dall'intuizione delle cose più semplici, cercheremo di elevarci con gli stessi gradini alla conoscenza di tutte le altre"*

Indubbiamente l'americano Skinner fonda il principio delle "piccole tappe" (STEP BY STEP) sulla necessità di non dare mai all'allievo l'occasione di fare degli errori, dunque di poter essere "ricompensato" dall'annuncio di un comportamento corretto come il cane è ricompensato con lo zucchero" (fig. 1).

b) *Programma articolato di CROWDER*

Nel programma lineare le risposte non portano nulla di nuovo alle nozioni descritte nei vari "frames" e nei relativi "items", esse hanno uno scopo pedagogico di RINFORZO dell'item appena prescritto. Se un allievo sbaglia una risposta, è costretto a rivedere la relativa domanda, cioè deve modificare il proprio comportamento con la sola indicazione dell'errore commesso.

Questo non è accettato da Crowder, il quale sostiene che è bene orientare l'allievo, che ha commesso un er-

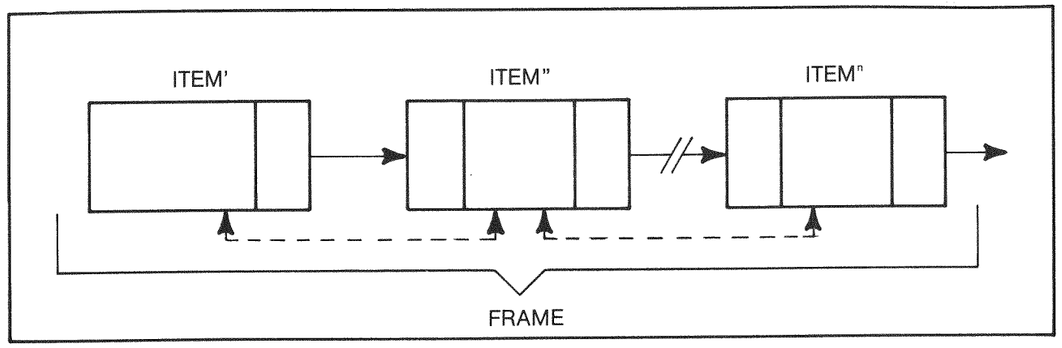


FIG. 1

rore, verso il rimedio specifico prima di condurlo all'item seguente.

Studiò quindi un nuovo programma diverso da quello di Skinner, strutturato come in fig. 2.

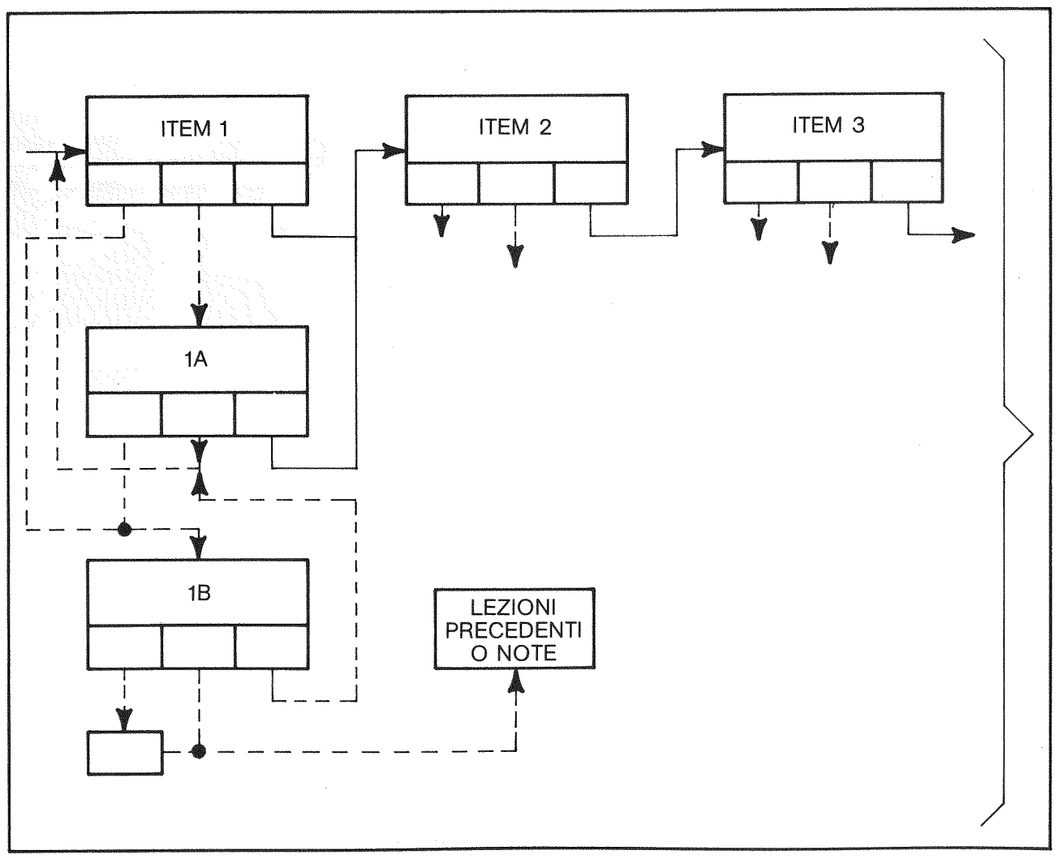


FIG. 2

La ragione della diversità fra i due metodi esaminati è la disputa che oppone "skinneristi" a "crowderisti" circa un "valore pedagogico dell'errore".

Per i primi, l'errore è odioso, bisogna impedirlo ad ogni costo suddividendo i frames in items molto brevi. Le loro ragioni sono quelle di psicologi del comportamento, che hanno l'abitudine di analizzare dei comportamenti umani "primari": ogni errore, segnalato immediatamente dal dispositivo, crea nell'allievo una "motivazione negativa" e lo scoraggia.

Crowder pensa al contrario che – in limiti ragionevoli (al max il 30%) l'errore sia utile: è bene che l'allievo faccia degli errori perchè questo permette a coloro che hanno soltanto una comprensione superficiale di una domanda, di approfondire la sequenza e di capire meglio.
Le domande che terminano le sequenze *hanno essenzialmente uno scopo diagnostico e la tecnica si basa sul fatto che la diagnosi così fatta può essere utilizzata immediatamente per fornire rimedi specifici all'allievo.*

Nei metodi *lineari* l'istruzione è concepita in modo che l'errore sia improbabile: è per questo che le sequenze sono molto corte e che si utilizzano degli "indicatori" (cues) che hanno l'effetto di favorire la scoperta della risposta giusta (si ricercheranno, per esempio, delle analogie, delle opposizioni, delle parole sottolineate, dei disegni, per aiutare l'allievo).

Nei metodi *ramificati,* le sequenze sono più lunghe, richiedono un maggior sforzo di attenzione, ma in genere la risposta giusta ad una domanda fatta, deve essere scelta fra un gruppo di risposte possibili *predeterminate* (risposte a scelta forzata).

Per scegliere la risposta giusta, l'allievo può scegliere fra più tattiche:

- può scegliere a caso;
- può esaminare una dopo l'altre le risposte possibili, provare il loro valore e quando ne ha trovata una che gli embra buona, sceglierla;
- può procedere come nel caso precedente, ma non scegliere la risposta buona prima di aver esaminato tutte le risposte possibili;
- può costruire la sua risposta senza guardare le soluzioni proposte, e cercare quella fra di loro che corrisponde alla sua risposta. Se nessuna corrisponde, ricomincia.

c) *Programma ramificato "SKIP-BRANCHING" di Kay*

Il Prof. Kay ha voluto una maggiore partecipazione attiva dello studente ed ha realizzato un programma denominato "SKIP BRANCHING".

Nello "skip branching" l'allievo legge delle sequenze della stessa lunghezza di quella del metodo articolato, ma costruisce la sua risposta senza essere aiutato dalla presentazione di possibili risposte. Una volta scritta la sua risposta, la paragona con la "risposta giusta" e spetta a lui decidere se la sua risposta è identica a quella che gli è stata presentata come buona.

In questo caso l'atteggiamento dell'allievo è più attivo (creativo) (fig. 3).

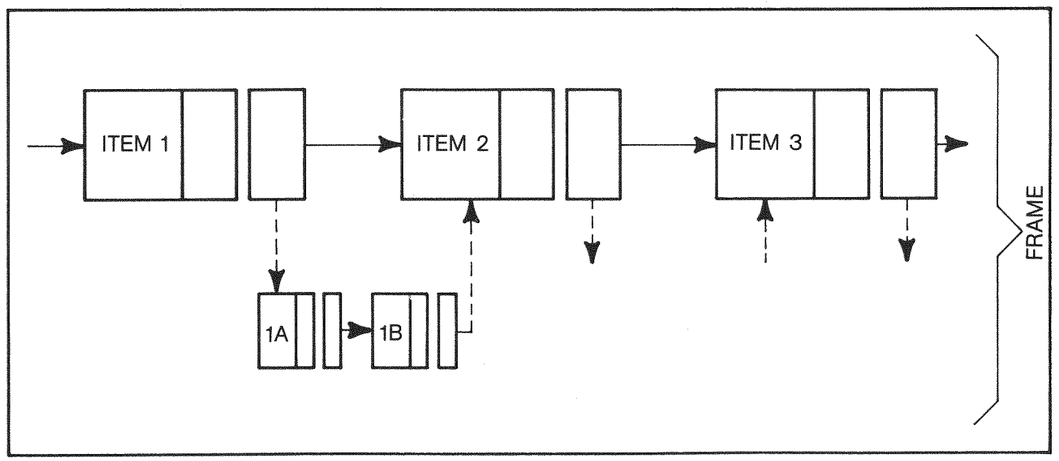


FIG. 3

Dalle osservazioni fatte sull'attività dello studente in istruzione programmata, si può trarre questa duplice conclusione:

- che la profondità dell'attività deve essere strettamente legata al livello "creativo" dell'allievo e che un errore frequente dei metodi tradizionali è di sperare troppo dalla capacità dell'allievo (è sempre il programma e mai l'allievo che è responsabile);

- meno l'attività è *profonda* (più è meccanica, del tipo "riflesso condizionato") meno l'esercizio di ricerca della riposta giusta costituirà un "apprendistato all'apprendere": più l'allievo immagina dei mezzi per trovare delle soluzioni, più si forma per apprendere nuove conoscenze.

Per quanto riguarda l'errore che l'allievo può commettere i creatori dello "skip branching" ragionano allo stesso modo di Crowder: ogni volta che l'insegnamento di una materia non è *meccanico* (come una procedura) "l'errore è utile".

In base a quanto accennato, sono stati realizzati:

- laboratori linguistici
- macchine per insegnare

facendo ricorso ai metodi su accennati e sfruttando le tecnologie audiovisive.

TECNICHE CAI

A. Andronico
Dipartimento di Matematica, Università degli Studi di Lecce

1. INTRODUZIONE

Le *tecniche CAI* (Computer Assisted Instruction) sono nate come naturale superamento del concetto di istruzione programmata legata soprattutto a metodologie di presentazione dei concetti oggetto dell'apprendimento e sono noti gli schemi classici di B.F. Skinner (schema lineare) e di N. Crowder (schema ramificato). Ma sono anche nate in contrapposizione a: maestro, voce, lavagna. Le prime applicazioni CAI risalgono agli anni 50 e tutte le applicazioni successive fino alla metà degli anni 70 sono state basate sugli schemi lineari prima, ramificato poi, conducendo a definire diverse modalità (S 4) di uso CAI man mano che lo sviluppo della informatica ha consentito tecniche di costruzione di sistemi di elaborazione e metodologie di programmazione evolute interessanti per le implicazioni dirette con i processi di comunicazione uomo-macchina.

I concetti di multiprogrammazione, time-sharing, parallelismo di elaborazione, interattività sono stati i più validi sostegni teorici alle tecniche CAI dal punto di vista tecnologico mentre cominciavano a farsi luce le teorie logiche connesse agli studi sulla cosiddetta intelligenza artificiale, reti semantiche, modelli per la rappresentazione di contesti limitati, teorie sulla rappresentazione della conoscenza, che costituiscono la base per la costruzione di sistemi didattici avanzati di CAI.

2. CAI E LIBRO

Una domanda che ci possiamo subito porre è: CAI e libro sono in contrapposizione? Certamente no. Tuttavia, appare evidente che le tecnologie emergenti e, in modo specifico quelle CAI, pur non togliendo valori al libro in sé, mettono in crisi la sua funzione in senso didattico. Non è una novità. Il libro stesso è stato creatore di crisi in tempi non più recenti, da dopo la rivoluzione della stampa alla fine del 18° secolo, epoca in cui il libro entra a far parte del sistema educativo. Se il libro ha provocato reazioni negative da parte degli educatori, come del resto anche la scrittura non era rimasta indenne da obiezioni e critiche, è evidente che solo l'idea che il libro, come oggetto-memoria collettiva, possa sparire o perdere alcune fra le sue funzioni proprie assegnategli fino ad oggi non solo a livello scolastico ma, e soprattutto, a livello culturale in senso lato, può essere preoccupante. A livello formazione certamente bisogna pensare che le attuali modificazioni sociali e di vita, indotte dalla tecnologia della comunicazione, hanno posto nuovi problemi per quanto riguarda la conservazione e la diffusione della informazione. Si può disquisire a lungo sul fatto che i processi di formazione sono sempre interattivi e che il tipo di interazione instaurato con l'utente del libro è multiforme, non in tempo reale, soffre di una certa staticità ed è a senso unico. Viceversa, il libro visto attraverso le tecniche CAI diviene un oggetto dinamico, animato su misura dal singolo utente e costruibile in tempo reale da ciascun utente. Questo aspetto dinamico fa sì che la fusione testo-immagine, il raccordo fra loro, l'integrazione conoscitiva dei due non siano legate al problema di produrre una, due, tre immagini accanto ad un testo, per poi immaginarsi quali altre possono o potrebbero essere necessarie (caso del libro) a chiarire o spiegare un concetto, un'asserzione, ecc. Le tecniche CAI consentono un capovolgimento totale di questo modo di vedere il libro: possono costituire un libro a cui si possono fare domande, che costruisce risposte, si possono chiedere immagini, integrazioni di immagini e testo. Non una, due, tre immagini, ma decine, centinaia, di più, se è necessario, non perchè queste restino lí ferme, ma perchè aiutino a trovare modelli, forme, strutture in senso conoscitivo. Certo non è possibile prevedere quali potranno essere i benefici di uso di tecniche CAI nei processi di formazione del futuro. Possiamo solo dire che, allo stato attuale, cominciano a mettere in crisi i benefici della scrittura e del materiale stampato, che tanto hanno impiegato a farsi sentire ed incidere sulla società. Nessuno, credo, si sente di rinunciare al piacere del libro, ma la sfera delle sue funzioni in certi campi sta diventando sempre più evanescente (in quello scolastico comincia a farsi sentire, anche se non conosco dati statistici su quanti libri si fanno comperare ai ragazzi che non vengono usati o sono usati una volta in un anno). Il vecchio dizionario quale fascino potrà avere in futuro se al proprio televisore-terminale ciascuno potrà consultarne una versione CAI, magari su videodisco, integrato con pezzi di testo (letterari, scientifici, artistici, ecc.) e immagini costruite da autori famosi, classici, moderni, ma con la possibilità di associare a un termine, a una voce-parola significati, contesti, analogie, traslati, metafore, ecc., in un rapporto diretto e coinvolgente.

Le difficoltà principali stanno nel costruire sistemi CAI e non nelle possibilità tecnologiche esistenti.

3. CAI E INSEGNANTE

Il problema di base è: l'insegnante sa (in generale) che cosa deve insegnare, ma spesso è assillato dal problema di come insegnarlo. Le tecniche CAI o le tecnologie didattiche si pongono per l'insegnante come associazione fra l'uso di criteri didattici in modo sistematico e strumenti tecnologici. L'insegnante esce dal concetto di lezione comune a tutti in senso tradizionale ed è obbligato a progettare (programmare) la lezione quasi su misura per il singolo. L'elaboratore come strumento diviene un collaboratore prezioso dell'insegnante, il quale interagisce con l'allievo senza diminuire il contatto personale, anzi aumentandolo.
Le tecniche CAI possono contribuire a valorizzare il ruolo dell'insegnante aiutandolo a creare quei cambiamenti di metodo e di esigenza che non producano effetti dannosi di fronte al mondo che cambia in fretta. Alle tecniche CAI l'insegnante accede mediante metodologie informatiche.

Poichè la programmazione elettronica costringe ad una scomposizione a livelli di un problema in sottoproblemi in modo operativo, per scopi didattici, la programmazione elettronica (CAI) costringe l'insegnante a scomporre e ricomporre il materiale didattico in modo da interagire con le possibili risposte dell'allievo. L'analisi quindi di un processo didattico gode della proprietà di essere univoca ed esplicita, anche se ciò può creare crisi nell'insegnante.

Le esperienze CAI sono comunicabili, adattabili e ripetibili anche su vasta scala, e ciò, senza per altro vincolare a schemi rigidi, può costituire per l'insegnante un controllo sulla loro validità ed adeguatezza.

4. CAI E ALLIEVO

Nella formazione scolastica è ormai noto che si considerano tre fini: partecipativo, conoscitivo e espressivo. Scrive N. Onesto che il fine conoscitivo è il mezzo, mentre il fine partecipativo e quello espressivo sono il messaggio. Ora, il fine conoscitivo è il processo di ricostruzione significativa della conoscenza, e quindi, per l'allievo, la costruzione del più alto livello conoscitivo e cioè quello formale. Ciò però non può essere disgiunto dal concetto di età, e quindi di livello scolare. Allora accade che, per ogni livello conoscitivo, può considerarsene sempre uno precedente. Se diciamo logico l'uno e pre-logico l'altro, nella scomposizione del materiale didattico, troviamo che i concetti sono sequenziali (dipendenti) per due o più livelli consecutivi e paralleli (indipendenti) per lo stesso livello.

Le tecniche CAI, per le caratteristiche costruttive proprie, interattive con l'insegnante e con l'allievo, consentono di tener conto delle esigenze dell'allievo ad ogni livello di conoscenza. Non solo, ma la relazione di dipendenza e indipendenza fra concetti può far sparire il concetto di materia, programma di quella materia, classe, ecc., in quanto i cammini didattici, propri di ciascun allievo, saranno, in generale, guidati dalle necessità dell'allievo nel rapporto fra lui, i problemi, i concetti e la serialità o il parallelismo del processo di apprendimento e della struttura della conoscenza. L'allievo è coinvolto in modo totale in tutto il suo processo di formazione divenendo egli stesso, con l'insegnante, gestore-controllore di tutto o parte del processo. A livello logico le tecniche CAI interattive con insegnante, allievo e mezzi audiovisivi giocano un ruolo importante in quanto costituiscono un sistema formale per operare trasformazioni simboliche su sequenze di simboli (di natura qualsiasi: alfabetica, numerica, iconica, ecc...) definite secondo le regole del sistema CAI utilizzato.

% di uso approssimativo	Modo	Settori	Obiettivo
20%	Esercitazione (Drill and practice)	Descrittivi	Acquisire abilità predefinite
50%	Istruzione (Tutor)	Scientifico-Tecnico	Costruzione di concetti nuovi
10%	Indagine (Inquiry)	Scientifico-Tecnico	Sviluppo e capacità critiche
8%	Giochi (Games)	Scientifico-Tecnico	Prova e verifica di decisioni
8%	Simulazione (Simulation)	Scientifico-Tecnico	Macchina per esperimenti
2%	Risoluzione di problemi (Problem Solving)	Scientifico-Tecnico	Costruzione e strutturazione di conoscenze
2%	Dialogo o Socratica	Scientifico-Tecnico	Formazione di abilità intellettuali particolari

A livello didattico poi le sequenze di simboli possono essere:

1) esercizi su un dato argomento e le trasformazioni che l'allievo deve effettuare sono quelle necessarie a risolvere interattivamente gli esercizi posti per acquisire certe abilità prefissate;

2) definizioni di problemi dalla cui risoluzione in modo interattivo si vogliono raggiungere obiettivi quali: la costruzione di nuovi concetti, lo sviluppo di capacità critiche, la prova e la verifica di decisioni, la costruzione di esperimenti, la strutturazione di conoscenze, la formazione di particolari abilità intellettuali.

A tali obiettivi sono associati modi di tecniche CAI la cui nomenclatura ormai standard è: Esercitazione per l'obiettivo del punto 1) e nell'ordine Tutoriale, Indagine, Giochi, Simulazione, Risoluzione di Problemi, Dialogo per gli obiettivi di 2). La tabella 1 riporta una sintesi di quanto detto con le percentuali approssimate di uso di tecniche CAI nelle varie modalità di cui è riportata la corrispondente notazione in lingua inglese.

5. PROSPETTIVE CAI E RIVOLUZIONE TECNOLOGICA

Osserviamo subito che le tecniche CAI non sostituiscono l'insegnante ma lo amplificano in modo naturale. Esse richiedono strumenti di comunicazione flessibili e soprattutto l'integrazione di sistemi di elaborazione con sistemi audio e video. Lo sviluppo tecnologico determinatosi con la diffusione dei micro e personal computers ha consentito di sviluppare sistemi (e lo consentirà sempre più in futuro) di presentazione di informazione nelle forme grafica e audiovisiva a basso costo. Sistemi come TELETEXT, VIDEOTEL, VIEWDATA sono già realtà che consentono la ricezione su normale televisore di informazione grafica e alfanumerica in alternativa a quella video o integrazione dei diversi tipi sullo stesso supporto. Questi sistemi cominciano ad essere interattivi con l'utente e possono già configurarsi come valido appoggio e supporto a sistemi CAI. Richiamo qui il videodisco che sicuramente si imporrà nella messa a punto di sistemi CAI, potendo esso contenere oltre che informazioni di varia natura, anche programmi per elaboratore accessibili a chiunque se integrato in sistemi del tipo detto.

Le tecniche CAI potranno creare una certa rivoluzione su due fasce:
1) in campo didattico come l'unica via, forse, per superare il concetto di didattica "trasmissiva" ed equilibrare o eliminare gli squilibri fra i due atteggiamenti pedagogici dell'insegnamento per regole e per problemi valorizzando il processo di apprendimento per problemi in un processo ricorsivo di interazione con l'insegnamento per regole;
2) in campo editoriale potranno diventare dei "libri elettronici" di larga anzi larghissima utilizzazione senza nulla togliere al libro stampato di altissimo valore.

La velocità con cui si susseguono i cambiamenti tecnologici non credo autorizzi a fare previsioni oltre quelle citate.

6. BIBLIOGRAFIA

A. Andronico, Informatica: linguaggio, metodo, strumento, *III Convegno Nazionale CIDI Roma,* Aprile 1980.

A. Andronico, Le tecnologie didattiche nel processo insegnamento-apprendimento, *Atti Convegno "Insegnamento Scientifico e Ricerca didattica,* Firenze, Settembre 1980.

A. Andronico, Applicazione degli elaboratori alla didattica nella scuola secondaria in Italia e all'estero, *Atti del Convegno AICA,* Pavia, Settembre 1981.

N. Onesto, La cibernetica per l'evoluzione culturale, *Atti II Convegno di Cibernetica,* Montecatini Terme, 1975.

McLUHAN, L'ELETTRONICA, IL LIBRO

Bruna Zonta
Consiglio Nazionale delle Ricerche - Milano

Osannato, discusso, criticato, un po' dimenticato, dopo circa 20 anni dall'uscita dei suoi più noti volumi, McLuhan torna a far parlare di sè e per motivi forse diversi da quelli che lo hanno reso famoso. A riproporlo oggi in primo piano sono soprattutto le sue previsioni su un futuro prossimo guidato dalla elettronica, la quale – sono parole sue – è destinata a porre presto fine alle "antiche dicotomie fra tecnologia e cultura, fra arte e commercio, e fra lavoro e tempo libero".

Ad un recente convegno su "Il linguaggio della divulgazione" (Milano, 11-12 febbraio 1982) più di una volta si è sentito richiamare il celebre "villaggio planetario", in cui la comunicazione avverrebbe istantaneamente e in maniera globale, grazie appunto alla cultura elettronica. Sí, perchè per McLuhan una cultura altro non è che l'"interiorizzazione" di tecnologie, e vedremo come.

Anche il nome del convegno che ospita questa mostra, "Editoria e comunicazione totale", non può non essere stato ispirato alle sue ripetute affermazioni.

Del resto, tutti o quasi i suoi slogan possono venire rivisti oggi in una luce leggermente cambiata. Vediamone alcuni.

Il medium è il messaggio. Si tratta dello slogan più diffuso ed anche di quello più severamente analizzato da detrattori e non. Nella nozione di medium McLuhan confonde – è stato detto – la forma del messaggio, il codice in cui viene espresso, e il canale o veicolo della trasmissione. Anche la parola "contenuto" (del messaggio) oscillerebbe fra le due accezioni, quella corrente, del "ciò che viene detto", e quella della teoria dell'informazione, del "numero di scelte binarie, ecc. ecc.".

Può darsi. Tuttavia è difficile sottrarsi alla suggestione dei suoi esempi. La televisione – sostiene McLuhan – è medium-messaggio non per ciò che ha trasmesso, trasmette o trasmetterà, ma per avere cambiato, con il suo solo ingresso nelle nostre case, il modo stesso di passare le serate ed i rapporti fra famigliari ed amici. Così il treno a suo tempo aveva modificato di non poco le abitudini sociali, non perchè portava a Londra o a Istanbul, ma per avere alterato i ritmi e le proporzioni del lavoro come dello svago. Il telefono poi ha creato un personaggio inimmaginabile prima, la ragazza-squillo. Ecc. Ecc. Che non sia il tradizionale "contenuto" l'elemento costitutivo del messaggio è ancor meglio dimostrato dall'esempio della luce elettrica, medium-messaggio per eccellenza, in quanto di per sè priva di contenuti particolari, quella luce che ha rivoluzionato le società in cui è penetrata, con il suo astratto messaggio di diffusione e unificazione. L'elettronica sta amplificando questo messaggio, mettendo a disposizione di tutti, o quasi, strumenti universali quali i calcolatori, producendo istantaneità di trasmissione e coralità di comunicazione. E se il tutto viene esposto in una atmosfera più da Ballo Excelsior che da saggistica accademica, non importa poi molto: il messaggio McLuhan è in ogni caso andato a segno. Sarebbe difficile prescinderne oggi.

Media caldi e media freddi. Anche a questo proposito dubbi e riserve non mancano. "C'è un principio base – scrive McLuhan – che distingue un medium caldo ...da un medium freddo... È caldo il medium che estende un unico senso fino ad una "alta definizione" ...". Ma l'elencazione non convince:

media caldi	**media freddi**
fotografia	cartoon
radio/grammofono	telefono
cinema	televisione
alfabeto fonetico	geroglifico/ideogramma
stampa	parola orale
carta	pietra
sillogisma	aforisma
meccanica	elettronica
...	...

Proprietà dei media caldi: "dicono tutto", non richiedono partecipazione (anzi, escludono), accelerano, sono uniformi-ripetitivi, esplodono, sono distributivi, generano specializzazione e frammentazione. I media freddi, all'opposto: implicano un alto grado di completamento e partecipazione (includono), rallentano, sono variati e variabili, implodono, sono conservativi, riducono una vera e propria ritribalizzazione delle società.

Inutile aggiungere che le simpatie di MacLuhan sono per i secondi (anche se può sconcertare la scelta di chiamare "freddi" i media favoriti, quelli fra l'altro più legati al mondo delle emozioni!). Insieme a Francesco Bacone oppone ad una prosa calda, cioè allo scrivere "secondo metodi", diremmo noi "strutturato", una prosa fredda, cioè uno scrivere più discontinuo, incompleto ed allusivo.

La Galassia Gutenberg – dice McLuhan in una lettera – vuole presentarsi più come un mosaico di ideogrammi che come una esposizione lineare. Gli stessi titoli-glosse-indici che delimitano le tessere di questo mosaico appaiono spesso riferirsi a qualcosa che si trova non nel testo immediatamente precedente o seguente, ma da qualche altra parte, da scoprire per associazioni. Citazioni, anticipazioni, rimandi, riprese, ripetizioni, ridondanze si alternano in una prosa che cerca ostinatamente di sottrarsi alla legge della sequenza, che egli considerava tipica, fra l'altro, della stampa, tecnica ormai obsoleta, inadeguata comunque ad esprimere e trasmettere idee, almeno se si vuole rispettare la natura simultanea e polifonica del sistema nervoso stesso.
Il suo ultimo libro, *Il medium è il massaggio* (sic), in cui tutti i mezzi offerti dalla stampa, da quelli tipografici a quelli fotografici, sono composti in configurazioni da collage, rappresenta in questo senso il tentativo paradossale di servirsi di un medium per negarlo. E se questo tentativo non si può considerare certo riuscito, il discorso di McLuhan apre ugualmente nella direzione di ciò che chiamiamo "libro elettronico", dove non è rilevante come il testo è stato scritto, cioè quale è l'ordine e la disposizione delle sue parti, ma come esso viene letto e dove ogni lettura potrebbe essere diversa da tutte le altre. In alcuni casi si potrebbe anzi dire che il libro elettronico non esiste come libro se non nel momento e modo di lettura: cosa tanto più vera quando si tratti di testi ideati appositamente per lo strumento elettronico, ma plausibile anche per testi tradizionali usati per studio e consultazione. Ne potrebbero derivare nuovi concetti per la stilistica e – perchè no? – persino una nuova retorica.

La cultura come interiorizzazione di tecnologie. L'introduzione di una tecnologia genera sempre, secondo McLuhan, uno stato di stress generale esattamente come un intervento chirurgico: la "zona d'urto" si intorpidisce, mentre tutto il sistema entra in crisi e l'equilibrio fra le componenti (nel caso dei media, i sensi) viene spostato con ripercussioni nello psichico come nel sociale. Poi, poco alla volta, l'assimilazione e l'assuefazione producono i loro effetti: altri equilibri vengono cercati e trovati. La nuova tecnologia e le sue conseguenze vengono interiorizzate e si fanno cultura.

L'invenzione della stampa, una volta divenuta cultura tipografica, ha creato l'autore nel senso moderno e il suo pubblico, nonchè ovviamente l'editore, ma produsse anche, attraverso i meccanismi della ripetizione, la conoscenza applicata, l'individualismo e il nazionalismo (non è qui possibile nè ricordare nè motivare tutti gli effetti, plausibili e non, attribuiti da McLuhan alla rivoluzione tipografica).

Come i media elettronici agiranno, o stanno agendo, sino a dove giungeranno i loro effetti, questo McLuhan lo ha solo intravisto e non era forse nelle sue possibilità analizzarlo sino alle ultime conseguenze.
Del resto, l'"interiorizzazione" della tecnologia elettronica nel suo complesso non è compiuta, i traumi e le tensioni sono probabilmente ancora in atto, per lo meno nella popolazione adulta.

BIBLIOGRAFIA

M. McLuhan,
The Gutenberg Galaxy: The Making of Typographic Man, University of Toronto Press, 1962 (Trad. it. *La Galassia Gutenberg,* Armando Armando, Roma, 1976)

M. McLuhan,
Understanding Media, McGraw-Hill Book Co., New York, 1964 (Trad. it. *Gli strumenti del comunicare,* Il Saggiatore, Milano, 1967)

M. McLuhan e Q. Fiore,
The Medium is the Massage, Bentam Books, New York, 1967 (Trad. it. *Il medium è il massaggio,* Feltrinelli, Milano, 1968)

G. Gamaleri,
La galassia McLuhan, Armando Armando, Roma, 1976

UNIBOOK

M. Casazza
Istituto di Cibernetica, Università degli Studi di Milano

1. INTRODUZIONE

Viene qui descritto UNIBOOK, una delle realizzazioni del libro elettronico sviluppate all'Istituto di Cibernetica della Università degli Studi di Milano.

UNIBOOK è stato sviluppato con il linguaggio "C" ambientato in un sistema operativo UNIX° con impiego di terminali alfanumerici ASCII.

La presente descrizione è suddivisa in 4 parti:
- Analisi dei requisiti
- Struttura logica del libro
- Comandi disponibili
- Implementazioni

2. ANALISI DEI REQUISITI

Nella fase di disegno della struttura di UNIBOOK ci si è basati sulle seguenti considerazioni:

- In un libro ci sono almeno due tipi di informazioni: quelle concernenti strettamente l'argomento trattato e quelle di supporto, cioè esempi, approfondimenti di alcune parti, prerequisiti, esercizi, ecc.

- Il lettore ha spesso necessità (specialmente quando il libro tratta di argomenti tecnici) di accedere a una particolare parte di testo.

- Strumenti come l'indice analitico e il glossario sono molto utili ma, sul supporto cartaceo, scomodi da usare.

In base a queste considerazioni si sono prese le seguenti decisioni:

- Tutte le informazioni definite "di supporto" non devono essere frammischiate alle altre, ma devono comparire separatamente e l'utente deve poter accedervi solo su richiesta esplicita.

- Occorre mettere a disposizione uno strumento che permetta all'utente di accedere velocemente alla parte del libro che lo interessa. Il solo indice generale è stato giudicato insufficiente.

- L'indice analitico e il glossario devono essere molto ricchi ed efficienti e consultabili in ogni momento. L'indice analitico deve permettere una veloce visualizzazione di tutti i rimandi. Inoltre, in fase di preparazione del libro, occorre fornire uno strumento per costruire gli indici automaticamente.

Nell'attuale versione di UNIBOOK, la stesura dei testi viene effettuata impiegando gli editor disponibili sotto UNIX.

° UNIX è un Trade Mark dei Bell Laboratories.

3. STRUTTURA LOGICA DEL LIBRO

Il libro visto da UNIBOOK è composto da:
titolo
indice generale
indice analitico
glossario
paragrafi
capitoli
parti
dettagli
abstract di parte
abstract di capitolo

Il "titolo" è sia il nome del libro che quello del file che contiene la prima schermata mostrata da UNIBOOK.

L'"indice generale" è l'elenco delle "parti", "capitoli", "paragrafi" presenti nel libro. Viene costruito manualmente. È consultabile in qualsiasi momento. L'"indice analitico" contiene i termini più significativi presenti nel testo e per ciascuno una lista con il nome dei paragrafi e il numero della linea in cui compare. I termini che devono essere compresi nell'"indice analitico" vengono segnalati da chi scrive il libro facendo precedere il termine da un carattere particolare. Questi termini vengono raccolti automaticamente nell'analitico, consultabile da qualsiasi contesto.

Il "glossario" è costruito manualmente. È consultabile

anch'esso in qualsiasi momento. Il testo contenente le informazioni più strettamente legate all'argomento trattato è contenuto nei "paragrafi".

Le altre informazioni di supporto sono contenute nei "dettagli". Nel testo dei "paragrafi" possono comparire segnalazioni del tipo (D5) che indicano che esiste un "dettaglio" logicamente legato a quel punto del testo. I "capitoli" non contengono testo; hanno il solo scopo di raggruppare logicamente i "paragrafi". Analogamente, le "parti", che hanno il solo scopo di raggruppare logicamente i "capitoli". Gli "abstract di parti" e "di capitoli" sono brevi riassunti del contenuto dei paragrafi relativi, volti a guidare il lettore nella ricerca di argomenti particolari.

4. COMANDI DISPONIBILI

Nella scelta dei comandi da fornire all'utente si è cercato di trovare il giusto mezzo fra sofisticazione e semplicità. Si è in particolare cercato di limitare al massimo il numero dei comandi e di fare in modo che lo stesso comando, in situazioni diverse, facesse ciò che pareva più sensato.

UNIBOOK viene richiamato scrivendo:
unibook <nome-libro>.

La prima schermata costituisce il "titolo". Subito dopo si hanno a disposizione tutti i comandi:

indice generale	(i)
indice analitico	(a)
glossario	(g)
paragrafo successivo	(s)
paragrafo precedente	(r)
salto ad altro paragrafo	(j)
salto all'ultimo visto	(u)
visualizzazione abstract	(b)
apertura finestra	(c)
spiegazioni	(h)
verso visualizzazione: avanti	(>)
: indietro	(<)
pagina successiva o precedente	(p)
linea successiva o precedente	(l)

I comandi (s) e (r) permettono di leggere sequenzialmente il testo dei "paragrafi". Il comando (j) chiede il nome di un "paragrafo" e lo visualizza. Per ritornare al "paragrafo" di partenza si usa (u). Il comando (d) chiede un identificatore, numero o lettera, e mostra il "dettaglio" relativo. Il lettore può poi tornare al "paragrafo" di provenienza. Il comando (b) chiede se si vuole il precedente o il successivo "abstract" e lo mostra. In questo modo il lettore può visualizzare in sequenza tutti gli "abstract". Con i comandi (g) e (p) si puo procedere sequenzialmente nella visualizzazione dei "paragrafi" a partire dal "capitolo" cui l'"abstract" raggiunto si riferisce oppure con il comando (u) si può tornare al "paragrafo" di provenienza. Il comando (g) chiede la parola da cercare e mostra la definizione contenuta nel "glossario". Il comando (a) chiede la parola da cercare nell'"indice analitico" e mostra tutti i rimandi. A questo punto il lettore può chiedere la visualizzazione dei "paragrafi" corrispondenti. Una volta mostrato un "paragrafo", il lettore può scegliere se continuare la lettura sequenziale da quel punto o se farsi rivisualizzare i rimandi per scegliere un altro "paragrafo". In ogni momento può tornare al "paragrafo" di provenienza con il comando (u).

Esistono due modalità di visualizzazione: a schermo pieno o a metà schermo. Nel secondo caso la metà superiore dello schermo rimane "congelata" e mostra sempre lo stesso testo, mentre le operazioni proseguono nella metà inferiore. Il comando (c) permette di passare dall'una all'altra di queste modalità. Se ci si trova nella modalità a schermo pieno, il comando congela la metà superiore dello schermo così come esso si trova in quel momento ed entra nella modalità a metà schermo. Se ci si trova nella modalità a metà schermo, il comando chiede se si desidera continuare dal "paragrafo" mostrato nella metà superiore o da quello mostrato nella metà inferiore e passa alla modalità a schermo pieno. La modalità a metà schermo ha ovviamente il vantaggio di poter mostrare due "paragrafi" diversi contemporaneamente. Poichè non si può limitare la lunghezza di un "paragrafo" a quella del video, occorrono comandi per mostrare tutte le parti del testo. I comandi (p) e (l) mostrano rispettivamente la pagina e la linea successiva rispetto al verso corrente di lettura. Questo può essere modificato con i comandi (>) e (<), avanti e indietro rispettivamente.

5. IMPLEMENTAZIONE

Il programma UNIBOOK è diviso in tre parti:
- controllo della memoria (memory dispatcher)
- controllo del video
- logica del libro

Gli obiettivi fondamentali che ci si è proposti sono stati:
- alta velocità di risposta e di rappresentazione sullo schermo
- isolamento di tutte le caratteristiche dipendenti dallo schermo
- imposizioni minime a chi scrive i testi.

Poichè si è riscontrato che i maggiori responsabili della scarsa velocità sono le strutture dei "paragrafi" da file, si è deciso di tenere in memoria centrale la maggior quantità possibile di testo.

Il memory dispatcher provvede appunto alla distribuzione della memoria (in pezzature da 512 byte) per la registrazione dei "paragrafi" letti. Quando la memoria è satura, per ottenere lo spazio necessario per altro testo, vengono scaricati i "paragrafi" letti meno recentemente.

Si è resa la gestione del video sostanzialmente indipendente da tutte le caratteristiche (dimensioni e sequenze di controllo) del tipo di terminale usato. Ciò permette di usare più tipi di terminale senza alcun problema e favorisce tutte le migliorie grafiche e funzionali di UNIBOOK (le due modalità di schermo pieno e metà schermo non erano state previste all'inizio e sono state aggiunte in 1 ora).

Il formato dei testi dei "paragrafi" è totalmente libero; solo il nome dei file contenenti i "paragrafi" e la loro disposizione all'interno del file system sono rigidi. Si è infatti pensato di sfruttare la struttura ad albero del file system di UNIX. Così il nome del libro coincide con il nome di una directory che contiene altre directory che costituiscono le "parti"; queste a loro volta, contengono solo directory che coincidono con i "capitoli"; e questi, finalmente, contengono i "paragrafi", cioè i file veri e propri. Il formato dell'"indice analitico" non coinvolge l'utente perchè esso viene costruito automaticamente. Il "glossario", invece, deve essere compilato a mano seguendo una sola regola: la parole di cui si dà definizione devono comparire da sole su una riga ed essere accostate a sinistra, mentre il corpo della definizione non ha limiti di lunghezza o di struttura se non quello che le sue linee non devono iniziare in prima colonna.

VERSO IL MANUALE ELETTRONICO

E. Bertolotti
Istituto di Cibernetica, Università degli Studi di Milano

La presente nota ha lo scopo di illustrate i risultati di una esperienza di documentazione di programma mediante "libro elettronico". Il programma scelto è una procedura software disegnata-documentata con la metodologia Wellmade.

Tale metodologia fornisce un modo strutturato di procedere nell'analisi e implementazione di programmi basata sul raffinamento per livelli successivi delle specifiche iniziali. In breve, per prima cosa, viene individuata una macchina astratta di primo livello che riassuma nella sua generalità il problema che deve essere risolto, formalizzando ciò che è contenuto nelle specifiche di procedura. Lo sviluppo procede nella definizione di una o più macchine di secondo livello che costituiscono il raffinamento e la scomposizione in sottoproblemi della macchina di primo livello. Si procede quindi in modo top-down fino a che il raffinamento ha raggiunto un punto tale da non richiedere altro che una codifica nel linguaggio di programmazione usato. La documentazione, prodotta in modo concorrente nella fase di disegno, risulta strutturata anch'essa in diversi livelli di dettaglio ordinati secondo un rapporto gerarchico che per sua natura ben si adatta ad un supporto quale il libro elettronico.

Il libro elettronico proposto, infatti, struttura il testo in "parti", "capitoli", e "paragrafi" costruendo una alberatura che l'utente lettore, da terminale, può esplorare mediante un insieme di semplici comandi per cercare e visualizzare sul video informazioni al livello di dettaglio desiderato. Ricordiamo che i comandi sono del tipo: leggi prossimo capitolo, torna al paragrafo precedente, cerca nel glossario la parola "...", visualizza il dettaglio "...", e simili.

Le seguenti figure sono tratte dalla documentazione ottenuta nell'esperienza citata che riguarda una procedura software di aggiornamento conti correnti in ambiente bancario.

A 1.1 (comando che visualizza il paragrafo 1.1)

```
1.1.  LA MACCHINA DI PRIMO LIVELLO M1
      ------------------------------------
      Contenente il numero di conto corrente ed altri dati ad esso specifici (D1) _____
      ------------------------------------
```

A 1.1.2 (comando che visualizza il paragrafo 1.1.2)

```
1.1.2 M1 DESIGN DESCRIPTION

      PREDICATES
      LET n1, n2, n3: num;
      n1<n2 MEANS ABSTRACT _____
      ------------------------------------
```

U (comando che visualizza l'ultimo paragrafo visto)

```
1.1   LA MACCHINA DI PRIMO LIVELLO M1
      ------------------------------------
      Si tratta quindi nei tre casi di una successione
      di elementi omogenei (records) ciascuno contenente il numero di conto corrente ed altri dati ad esso specifici (D1)
```

D1 (comando che visualizza il dettaglio 1)

```
D1    DEFINIZIONE DEI TRE TIPI FONDAMENTALI

      TYPES    account-File: account ARRAY
               amount: (anum:num;ainfo:aaltro)
      ------------------------------------
```

L'esperimento di sostituzione del libeo cartaceo con il libro elettronico nella documentazione di programmi è giustificato, fra l'altro, per i seguenti motivi.

La diversità dei tipi di utente lettore di una documentazione software nel corso del suo ciclo di esistenza è tale da far nascere diversi modi di accesso a seconda delle differenti esigenze. Le metodologie di disegno, o comunque i modi più diffusi di procedere nello sviluppo di software, sono già tali da produrre una documentazione quasi sempre strutturata in vari livelli di dettaglio, via via crescente, a partire dai "requirements" primitivi. La stesura sequenziale su supporto cartaceo di tali diversi livelli all'interno di uno stesso documento costituisce spesso un ostacolo nella consultazione, almeno per quel tipo di lettore che non voglia leggere tut-

to il testo, pagina dopo pagina, argomento dopo argomento, ma che, per esempio, sia interessato soltanto alla lettura di passi di carattere generale. Il libro elettronico costituisce una soluzione al problema delle diverse esigenze di consultazione di una unica documentazione. Nella procedura presa ad esempio l'utente che fosse interessato solamente alle specifiche generali del programma potrà richiamare sul video le descrizioni funzionali delle diverse macchine astratte ottenendo una lettura sequenziale delle sole informazioni di minore dettaglio. Data la grande quantità e capillarità delle informazioni contenute in un documento di supporto ad un software che non sia di banali dimensioni, sorgono soprattutto problemi di accesso e ricerca da parte dell'utente lettore.

Più precisamente, questi problemi sono di due tipi. Per primo quello di reperibilità delle informazioni: nell'esempio precedente, si vede come, scorrendo i paragrafi di livello più generale contenenti le descrizioni funzionali, si possa, trovata la funzionalità voluta, scendere direttamente nei paragrafi di maggior dettaglio per cogliere gli aspetti implementativi. Un secondo tipo di problema riguarda la velocità di accesso. Sempre con riferimento alle figure, si noti come con semplicità il tecnico progettista, che spesso e volentieri è costretto a saltare da un paragrafo all'altro per poi tornare di nuovo al precedente, possa con un solo comando visualizzare l'ultimo paragrafo visto, invece di scartabellare numerose pagine come avverrebbe su un testo cartaceo.

L'esperienza di documentazione con libro elettronico (manuale elettronico) è stata rivolta alla valutazione della possibilità di definire uno standard di libro elettronico adatto anche a problemi di documentazione del software. L'esperienza ha messo in evidenza che è opportuno associare i comandi del libro elettronico alla struttura degli oggetti in esso rappresentati. È risultato comunque evidente che la documentazione elettronica certamente facilita la consultazione.
Del resto la specificità dei problemi di documentazione del software impone che, accanto a semplici comandi di lettura, siano presenti anche comandi orientati alla manutenzione sia nel momento della stesura sia in quello della lettura.

L'esperienza è stata condotta usando una versione di libro elettronico sviluppata da M. Casazza in ambito UCSD Pascal.

BIBLIOGRAFIA

A. Pizzarello, "Data Representation in Wellmade", *Note di Software,* No. 13-14, 1980

Wellmade: a Rational Design Methodology, Honeywell I.S., 1980

UN ESPERIMENTO DI TRASPORTO DA UN LIBRO ORDINARIO A UNO ELETTRONICO

Luisa Agliati
Istituto di Cibernetica, Università degli Studi di Milano

L'esempio che presentiamo consiste nella "trascrizione" da libro tradizionale a libro elettronico limitatamente al capitolo intitolato "Processi concorrenti", tratto da *Operating System Principles,* di Brinch Hansen (1973, Prentice-Hall, Inc., Engle Wood Cliffs, N.J.).

Si tratta di un testo tecnico con interesse didattico. Il capitolo riguardante i processi concorrenti permette di verificare l'impiego di un libro elettronico per assolvere sia le esigenze di colui che, già esperto in materia, desidera individuare rapidamente determinati concetti, sia quelle dello studente che necessita invece di studiare il testo. Il primo sarà interessato ad una lettura veloce degli "abstract", al fine di individuare facilmente quale sarà il "paragrafo" che desidera; il secondo, cioè lo studente, seguirà presumibilmente un altro tipo di lettura, e cioè: "abstract" di "parte", "abstract" di "capitolo", "paragrafo", "dettagli", con consultazione frequente dell'"indice analitico" e del "glossario".

La struttura del testo contenuto nel libro elettronico, ottenuta cercando di rispettare il più possibile la filosofia imposta da quest'ultimo, è visibile nell'indice generale, ed è costituita da sei "parti", ciascuna formata da due o tre "capitoli", a loro volta suddivisi in più "paragrafi", contenenti o no "dettagli". Per quanto riguarda la visualizzazione delle figure relative al testo in esame, questa è resa possibile dal collegamento con un proiettore di diapositive e le figure vengono segnalate all'interno dei "paragrafi" o "dettagli".

La lettura dei testi caricati su questo libro elettronico avviene per mezzo di una serie di comandi a menu, che vengono di volta in volta presentati all'utente lettore. Consideriamo, per esempio, la figura 1, nella quale è schematizzata la struttura a livelli della prima "parte", che ha per titolo: "Concorrenza". Tenendo presente i due tipi di utenti cui il libro è indirizzato, possiamo osservare che il lettore già esperto farà uso più frequente dei comandi "successivo", "precedente" che riguardano gli "abstract", per poi usare il "salto a paragrafo" o a "dettaglio", una volta individuato quello che cerca. Lo studente, invece, seguendo la lettura sequenziale, tenderà a ricorrere ai comandi "successivo" e "precedente" che riguardano i "paragrafi" e richiamando spesso l'"analitico" o il "glossario", allo scopo di non perdere informazione alcuna sull'argomento trattato; a costui sarà di grande aiuto il comando "ultimo visto", che permette di tornare all'ultimo paragrafo visto, compiendo "salti" senza per questo perdere il segno (figure 2 e 3).

Più che entrare nei particolari strutturali o contenutistici del risultato ottenuto, vogliamo accennare qui ad alcune difficoltà che inevitabilmente si riscontrano nel trascrivere un libro tradizionale sul supporto elettronico.

Esse sono dovute soprattutto al fatto che con questa operazione si cerca di trasformare un oggetto statico in qualcosa di dinamico, nel senso che, mentre il libro ordinario si presta egregiamente alla lettura sequenziale, ma meno bene a "salti" ed "escursioni" (tutte a carico del lettore, spesso con perdita del contenuto o indebite interpretazioni), il libro elettronico è concepito per un uso dinamico, e suggerisce, oltre che permettere, agili esplorazioni e letture di differente livello, con la conseguenza di migliorare qualità e tempi di apprendimento.

A questo proposito, il primo problema che il "trascrittore" incontra consiste nel conciliare, da una parte, la doverosa fedeltà all'autore anche per quanto riguarda l'impostazione formale assegnata al libro originario (che quanto meno rispecchia i diversi valori e pesi dei contenuti), e, dall'altra parte, i requisiti studiati appositamente per il libro elettronico. Questo, naturalmente, può venire ottenuto concentrando sinteticamente nei "paragrafi" le parti riconosciute come "portanti", e nei "dettagli" le spiegazioni, gli esempi, le applicazioni, le illustrazioni, le varianti; ma la decisione su ciò che è portante e ciò che è di arricchimento, come è noto, è un compito non banale.

Un altro problema è rappresentato dalla presenza degli "abstract", che nei testi tradizionali spesso non esistono o se esistono hanno talvolta finalità diverse. Nel caso del libro elettronico gli "abstract", sia delle "parti" che dei "capitoli" devono essere costruiti in modo da comporre nel loro insieme delle "miniature" del libro originario, rispettandone la completezza pur nei diversi livelli di astrazione.

Chi svolge questo lavoro è dunque chiamato a fare molto di più che non una "trascrizione". Da una parte gli viene richiesta una conoscenza professionale dell'ar-

gomento e dall'altra una capacità di vero e proprio rifacimento di testi nel rispetto della struttura generale prevista per il libro elettronico: un mestiere forse tutto da inventare.

FIG. 1

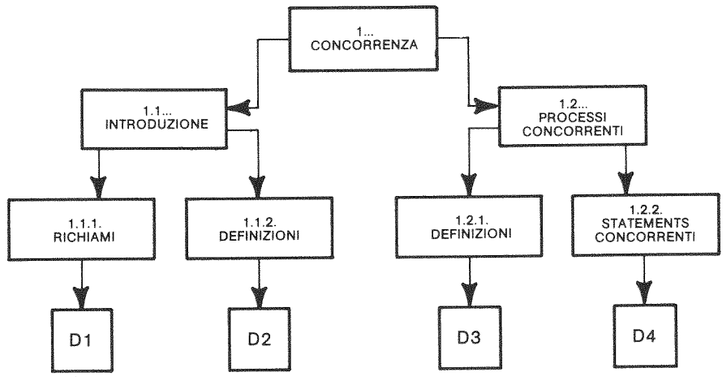


FIG. 2

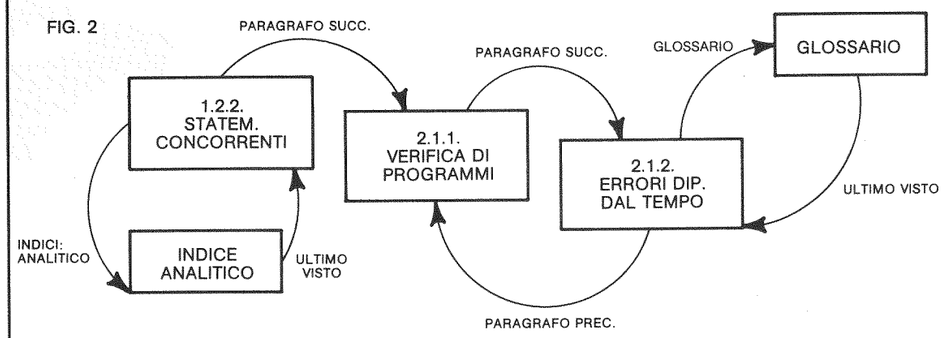
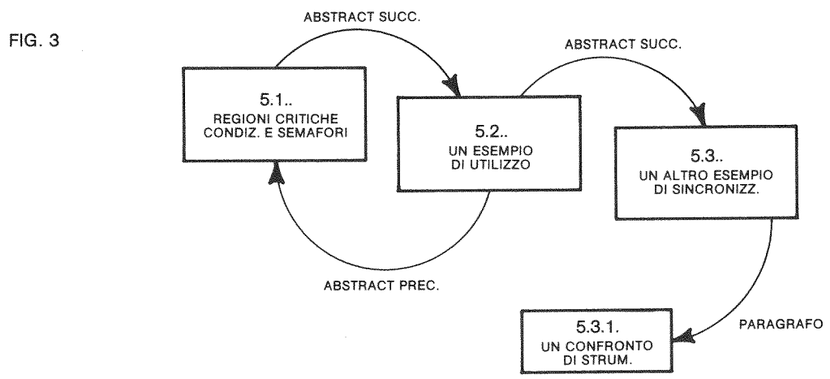


FIG. 3



TEX: UN SISTEMA DI COMPOSIZIONE PER LA STAMPA DI TESTI SCIENTIFICI E TECNICI

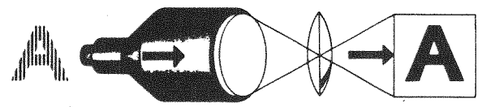
G. Canzii, M. Gualzetti
Istituto di Cibernetica, Università degli Studi di Milano

Riassunto

Primo di due lavori che descrivono due sistemi per stampare testi e generare caratteri, l'articolo esamina un sistema di composizione per la stampa di testi tecnici e scientifici, ricchi di formule e simboli matematici.
La creazione e lo sviluppo di tale sistema, denominato TEX, è opera di Donald E. Knuth dell'Università di Stanford; TEX è l'acronimo di Tau Epsilon Chi dall'iniziale della parola greca techne che in greco significa arte.
Scopo principale di TEX è di produrre, a differenza delle fotocompositrici tradizionali, direttamente su stampante ad alta risoluzione il testo finale, senza passaggi intermedi di tipo fotografico, salvo che per successive riproduzioni in grande tiratura.
Nell'articolo si illustrano con esempi le possibilità che offre TEX anche per chi non ha particolare esperienza di elaborazione elettronica dei dati.

1. INTRODUZIONE

Nell'ultimo decennio vi è stata una vera e propria rivoluzione in campo tipografico con l'introduzione delle fotocompositrici, in sostituzione delle tradizionali linotype. Le fotocompositrici permettono di effettuare l'impaginazione direttamente sul video con ottimi risultati anche qualitativi oltre che con risparmio di tempo. Senza voler entrare nel dettaglio delle caratteristiche di queste macchine, è interessante sottolineare quali sono i limiti che hanno portato allo sviluppo di sistemi più sofisticati, come ad esempio il TEX.
Tra le maggiori restrizioni vi sono la lentezza con cui queste macchine compongono il negativo di una pagina (dell'ordine dei 20 - 30 minuti) e il numero ristretto di fonti di caratteri utilizzabili, se non costringendo l'utente a cambiare continuamente la pellicola o il disco matrice, a seconda del supporto utilizzato, su cui sono incisi i caratteri di stampa. Nuove fotocompositrici recentemente immesse sul mercato si sono particolarmente avvantaggiate dal punto di vista tecnologico dei progressi dell'elettronica. Un esempio è dato dall'APS - Micro 5 controllata da un microcomputer. Il dato più interessante di questo modello è comunque determinato dal fatto che sfrutta l'alta risoluzione di un tubo catodico per trasformare un carattere digitale (cioè definito come un insieme di punti) in uno continuo:



Questo sistema ha una elevata velocità di fotoimpressione (circa 8.000 caratteri/secondo, circa 4.000 righe/minuto), può gestire nello stesso lavoro fino a 40 fonti di 128 caratteri ciascuna, ed avere una libreria di fonti che arriva a contenerne fino a 200. È evidente che il costo di un tale sistema è abbastanza elevato e quindi destinato a produzioni con elevata tiratura (giornali, riviste, ecc.). L'innovazione più rilevante è senz'altro quella di considerare un carattere di stampa come formato da una configurazione discreta di punti.
Questa evoluzione è stata seguita anche dalle stampanti dei calcolatori, data l'esigenza sempre crescente di ottenere una stampa migliore di quella precedentemente ottenibile con le stampanti ad impatto o meccaniche e con in più una certa libertà nella scelta dei caratteri.

Ciò è stato reso possibile dallo sviluppo tecnologico delle stampanti elettrostatiche e a laser. In questi modelli la pagina viene vista come un'unica matrice formata da milioni di piccoli rettangolini (pixel): per stampare un carattere viene annerita la configurazione di punti corrispondente. La qualità della stampa dipende unicamente dalla risoluzione della stampante, cioè dal numero di punti in cui è suddivisa la pagina.

Uno dei motivi che hanno limitato la possibilità di sfruttare in pieno questi sistemi è che si è sempre pensato di creare le fonti dei caratteri tramite piastre di firmware o con package software piuttosto modesti (al massimo 5 o 6 fonti).

Per questo motivo si è sviluppato lo studio di un adeguato sistema software per permettere di utilizzare qualsiasi insieme di caratteri, simboli speciali, decorazioni e figure.

Il presente articolo è stato pubblicato sulla rivista Pixel, n. 1, 1981

TEX

Alla base dello sviluppo del sistema TEX è lo studio fatto da D.E. Knuth sull'evoluzione dei caratteri di stampa nelle pubblicazioni dell'American Mathematical Society dal 1900 ad oggi e, in seguito, fino ai testi della fine del'400 e inizio del '500 degli italiani Feliciano, Torniello, Pacioli e Palatino; questi ultimi avevano analizzato le regole della calligrafia e, soprattutto Pacioli, studiato come costruire le regole dell'Alfabeto mediante regole geometriche (fig. 1).

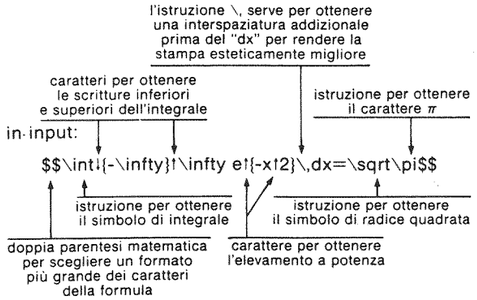
Basandosi sulla traduzione in forma algoritmica delle regole geometriche di costruzione dei caratteri, Knuth ha prodotto un sistema software (METAFONT) per la generazione delle fonti. Accanto a questo, in collaborazione con un gruppo di ricercatori, ha sviluppato un sistema software e hardware per la composizione di testi che si avvale, oltre che delle fonti generate algoritmicamente con METAFONT, di tecniche di impaginazione automatica.

TEX offre la possibilità di poter utilizzare qualsiasi tastiera di calcolatore, poichè fornisce un insieme di istruzioni che permettono di ottenere la stampa dei caratteri grafici e scientifici desiderati.

Ad esempio per ottenere il simbolo di sommatoria la sequenza di controllo è:
- input : $\sum $
- output: Σ

I due caratteri "$" agli estermi dell'istruzione sono parentesi matematiche che indicano al sistema che il carattere da stampare è di tipo matematico; il carattere "\ " è riconosciuto come speciale perchè serve ad avvisare il sistema che l'insieme di caratteri (un massimo di 8) che lo seguono, fino al primo spazio vuoto, costituisce un'istruzione da eseguire e non appartiene al testo originale.

Un altro esempio più complesso, ma che può far capire come sia abbastanza semplice per l'utente ottenere la stampa di un'espressione è il seguente:



È importante notare come siano stati usati soltanto i simboli normalmente disponibili su ogni tastiera di calcolatore; qualora un utente utilizzi una tastiera mancante di uno di questi simboli o nel caso di non funzionamento di un tasto, allora è sufficiente scrivere al posto del carattere mancante l'istruzione: \char <number> dove <number> va da 0 a 127 e corrisponde al carattere della fonte che si sta utilizzando. Ad esempio, se non dovesse funzionare la lettera "s" e si dovesse far stampare la parola "sasso", si scrive:

\char 116 a \char 115\char 1150

dove 115 è il numero corripondente alla "s" nella tabella utilizzata.

IL LINGUAGGIO DI INPUT DEL TEX

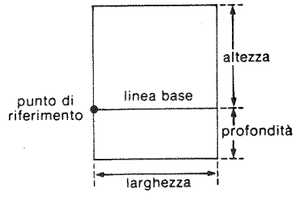
TEX è strutturato come un sistema unificato in cui le configurazioni matematiche vengono trattate come il resto del testo, a differenza di quanto avviene normalmente in fotocomposizione dove testo e parte matematica sono trattati in due modi differenti.

L'idea principale su cui si basa TEX per costruire una riga stampa è di supporre che essa sia composta di atomi, chiamato scatole, corrispondenti ai singoli caratteri.

Una formula matematica è costruita in modo analogo; per esempio: il numeratore e il denominatore di una frazione sono scatole e la linea di frazione è una scatola sottile, rettangolare e nera.

Le scatole hanno forma rettangolare a due dimensioni con associate tre grandezze: altezza, larghezza e profondità.

La figura mostra una tipica scatola:



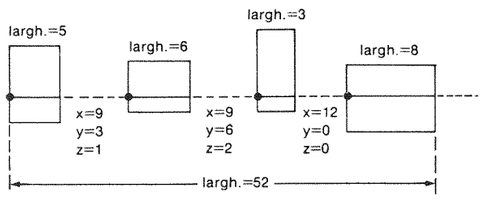
Il disegnatore della fonte decide l'altezza, la larghezza e la profondità del carattere e quale sarà il simbolo in essa raffigurato. TEX usa queste dimensioni per incollare insieme le scatole e formare le righe di stampa e per determinare sulla pagina la localizzazione del punto di riferimento di tutti i caratteri, in modo da averli tutti allineati sulla medesima riga.

Un esempio di come è inscatolato un carattere è il seguente:

I singoli elementi di una lista orizzontale o verticale sono separati da uno speciale tipo di componente elastica chiamata colla; ad essa vengono associate tre componenti (x, y, z) espresse in unità di lunghezza:

1) la componente *spazio* (x) è lo spazio tipico tra ogni carattere della fonte;
2) la componente *estensione* (y) è l'ammontare di spazio in più tollerabile tra i caratteri;
3) la componente *compressione* (z) è l'ammontare di spazio tra i caratteri che può essere rimosso, se necessario.

Per capire come agiscono queste tre grandezze si può osservare il seguente esempio di quattro scatole in una lista orizzontale, separata da tre gruppi di colla:

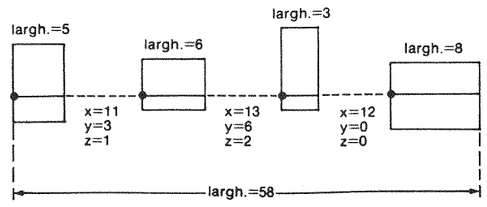


Il primo elemento di colla ha 9 unità di spazio, 3 di estensione e 1 di compressione; l'ultimo ha solo 12 unità di spazio e zero per le altre due componenti, il che significa che non è possibile estendere o restringere lo spazio (e quindi la lunghezza della riga).

La larghezza totale di scatole e di colla in questo esempio, considerando la sola componente x, è:

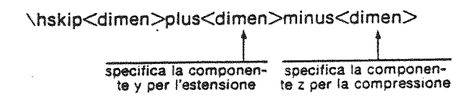
5+9+6+9+3+12+8=52 unità.

Questa è chiamata *larghezza naturale* di una lista orizzontale e costituisce il modo preferenziale per incollare insieme i caratteri. Supponendo che TEX debba costruire la stessa lista in una riga larga 58 unità, in questo caso la colla deve estendersi di 6. La massima estendibilità è di 9 unità e pertanto TEX moltiplica ciascuna di queste grandezze per 6/9 per ottenere le 6 unità necessarie. Il primo gruppo di colla diventerà 9+6/9x3=11, il secondo 9+6/9x6=13, mentre l'ultimo rimarrà 12 avendo la y=0; si ottiene così la riga desiderata:

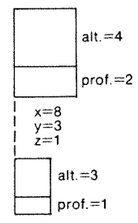


Se un elemento di colla è posto alla sinistra di una lista di scatole, l'effetto sarà di rigiustificarle, cioè di spostarle verso il limite più a destra.

Se si pone una colla infinita sia alla destra che alla sinistra della lista, si ottiene l'effetto di centrare la stringa di caratteri all'interno del campo di stampa.
Per definire la colla in una lista orizzontale si utilizza l'istruzione seguente:



Lo stesso procedimento viene eseguito per incollare tra loro le scatole di una lista verticale. In questo caso la colla sarà l'interlineatura sempre con associate le tre componenti, es.:



È evidente che per eseguire un'estensione o una compressione in una lista verticale di scatole le operazioni matematiche che TEX esegue al suo interno sono le stesse analizzate nell'esempio precedente per una lista orizzontale.

Per definire la colla in questo caso si utilizza l'istruzione seguente:

\vskip<dimen>plus<dimen>minus<dimen>

La colla *infinità* applicata sia a destra che a sinistra di una lista verticale di scatole produce l'effetto di centrare le righe di stampa nella pagina del testo.

Formattamento della pagina
Suddivisione sillabica automatica
Interruzione di un paragrafo su più pagine

TEX permette all'utente il formattamento della pagina, avendo a disposizione un insieme di istruzioni che consentono di stabilire le seguenti funzionalità:
- si possono ottenere i margini laterali fissi o variabili;
- numerare e titolare i paragrafi;
- numerare e titolare in cima alla pagina;
- inserire postille in fondo alla pagina;
- fissare la lunghezza della riga di stampa;
- fissare l'interlineatura e l'interspaziatura (cioè le componenti della colla);
- centrare i titoli dei testi o dei capitoli;
- formattare tabelle;
- incorniciare frasi o formule particolari.

Il sistema è in grado di suddividere automaticamente le parole per andare a capo in modo corretto rispettando il limite fissato dal margine o dalla lunghezza della riga di stampa (aggiustamento).

Questo evita all'utente di preoccuparsi dell'interruzione corretta della riga durante la battitura del testo. Tutto questo è possibile in quanto TEX è stato predisposto in modo da rispettare le regole grammaticali della lingua (attualmente solo l'inglese). Per rispettare i margini fissati, TEX agisce sulle tre componenti della colla tra i singoli caratteri della riga, in modo da estenderla o comprimerla della stessa quantità per ciascuno di essi, con un risultato che non altera l'esteticità e la qualità della stampa.

I lettori che hanno esperienza in questo settore potrebbero obiettare che tutte queste operazioni vengono normalmente eseguite anche dalle fotocompositrici più evolute. Infatti è importante sottolineare che tra le caratteristiche del sistema sono comprese anche quelle comuni alle fotocompositrici.

Ma tutto questo serve inoltre per introdurre una delle più interessanti prestazioni di TEX. I sistemi di fotocomposizione e i formattatori predefiniti, presenti finora sul mercato, quando devono interrompere una riga per rispettare il margine destro, eseguono i passi descritti in precedenza per l'aggiustamento e quindi, eseguita la suddivisione, passano alla riga successiva "dimenticando" ciò che è avvenuto prima. Questo modo di procedere porta a volte a risultati esteticamente non gradevoli o non accettabili come ad esempio l'ultima riga di un paragrafo che termina su una nuova pagina, oppure il titolo di un paragrafo sull'ultima riga di una pagina e il testo relativo stampato su quelle successive.

Di conseguenza l'utente è spesso costretto a ripetere il lavoro, ridefinendo i parametri di formattamento pagina (naturalmente questa procedura è più veloce per chi ha a disposizione le fotocompositrici, che permettono di visualizzare il testo esattamente come verrà stampato, in modo da poter eseguire subito le varie correzioni).

Il sistema TEX introduce un nuovo approccio al problema della divisione della riga: la fine del paragrafo può influenzare il modo con cui vengono interrotte le prime righe, con il criterio di avere il minor numero di righe interrotte all'interno del paragrafo. Con questo modo di procedere si salvano le esigenze estetiche e si evitano gli inconvenienti sopra descritti.

Tale approccio è basato su proprietà matematiche; utilizzando la colla si introduce il concetto di *badness* (cattiva qualità).

Quando una lista orizzontale di scatole ha una certa larghezza normale w - composta dalla larghezza delle sue scatole e dalle componenti spazio (x) della colla - una certa estensione y - somma delle componenti estensioni - ed una certa compressione z - somma delle compressioni - il badness della composizione della colla per costruire una scatola (riga) di larghezza W è definito dalla:

$$\text{badness} = 1 + 100\ t^3;$$

dove t è definito in modo che:

badness=1	se W=w,
badness=1+100(W-w/y)^3,	se W>w,
badness=1+100(w-W/z)^3,	se (w-z)≤W<w,
badness=infinito,	se W<(w-z).

Così il badness sarà molto piccolo se W si avvicina a w e avrà invece un valore molto elevato se W è più grande di w e la componente totale dell'estensione è limitata. Trattando il problema della suddivisione della riga in termini strettamente matematici occorre trovare una soluzione che renda minimo il badness, nel caso ottimale badness=1.
Per l'intera pagina si tratta di risolvere il seguente problema:

dato un testo di un paragrafo e l'insieme di tutti i possibili punti di suddivisione delle righe, trovare i punti di suddivisione che rendano minima la somma dei quadrati del badness delle righe risultanti.

La soluzione computazionale a questo problema viene ottenuta con tecniche di programmazione dinamica, riuscendo in tal modo a ridurre i passi di calcolo da 2^n a n^2, dove n è il numero dei possibili punti di suddivisione. Il concentto di badness è applicato con gli stessi criteri anche alle dimensioni verticali, in modo da evitare una cattiva interruzione di colonne o pagine.

LE FONTI

TEX è in grado di gestire, durante la composizione di un testo, 64 fonti di 128 caratteri ciascuna. Esse rappresentano i vari stili di stampa (ad esempio: "Roman", "Italico", "Cirillico", "Germanico", ecc.) in diversi corpi, i simboli e gli operatori matematici, i simboli misti e di parentesi ed altri.

Il corpo del carattere di una fonte è la grandezza che ne definisce il formato; la sua unità di misura è espressa in punti tipografici (un millimetro corrisponde a circa tre punti tipografici).

TEX al suo interno tratta tutte le grandezze in punti, ma permette all'utente di usarne anche altre, come i pollici, i centimetri e i millimetri, le unità in pica e punti Didot. Il sistema normalmente inizia a lavorare selezionando i caratteri della fonte "Roman" di corpo 10.

Quindi se l'utente deve iniziare a scrivere il testo con quella fonte non è obbligato a specificarla, mentre se desidera un altro stile deve specificare quella corrispondente con l'istruzione:

\<nome fonte><corpo>.

TEX consente all'utente di cambiare lo stile di stampa e l'impaginazione in qualsiasi punto del testo.

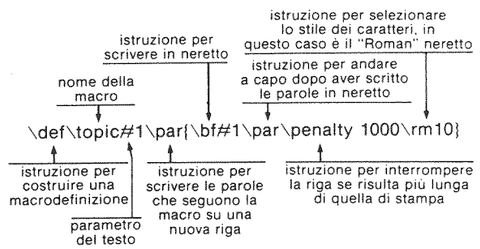
MACRODEFINIZIONI

Le macrodefinizioni sono delle istruzioni che contengono le definizioni delle sequenze di controllo fondamentali che servono al sistema per interpretare i comandi che riceve dall'utente. Queste macro sono residenti in una libreria chiamata "Basic TEX".

Le macro possono essere costruite dall'utente e, inserite nella libreria, vengono richiamate solo col nome.

Seguono due esempi che possono chiarire come si costruiscono e come si richiamano le macrodefinizioni.

Nel primo si suppone di dover scrivere un articolo per una rivista dove ricorrentemente una riga o una parola devono essere scritte in neretto (ad esempio: argomenti di un paragrafo, parole o frasi particolari). Invece di specificare ogni volta i passi che TEX deve compiere per ottenere quel risultato, si può costruire la seguente macrodefinizione all'inizio dall'articolo:



quindi \topic sarà l'abbreviazione di:

"\par{\bf#1\par\penalty 1000\rm}"

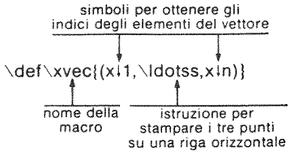
Ogni volta che sarà necessario scrivere nel testo qualcosa in neretto, invece di riscrivere l'istruzione nella sua forma estesa, sarà richiamata con \topic la macro corrispondente; come esempio se nel nostro articolo si deve scrivere in neretto il seguente titolo di una sezione di un paragrafo:

"la struttura della versione di Pascal"

in input è sufficiente scrivere:

\topic La struttura della versione di Pascal

Nel secondo esempio, prendendo il caso in cui nelle espressioni matematiche di un manoscritto ricorre frequentemente il vettore "(x1, ..., xn)", allora si può costruire la seguente macrodefinizione:

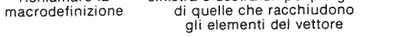


quindi \xvec è l'abbreviazione di "(x\1,\ldotss,x\n)" e ogni volta che si troverà nel testo questo vettore sarà sufficiente scrivere "\xvec" per richiamare la macro corrispondente; come esempio, se si vuole ottenere la stampa della seguente espressione:

si deve scrivere:

$$\sum_{(x_1, \ldots, x_n)\neq(0, \ldots, 0)} \left(f_{(x_1, \ldots, x_n)} + g_{(x_1, \ldots, x_n)} \right)$$

```
$$\sum\1{\xvec≠(0,\ldotss,0)}\bigIp f\xvec +
                    +g\xvec\bigrp$$
```



COME SI USA IL TEX

L'utente che vuole utilizzare il sistema TEX dovrà osservare una sequenza di lavoro per ottenere una stampa che soddisfi le sue richieste.

I vari passi da compiere sono i seguenti:
- caricare il "Basic TEX"
- caricare eventuali macro costruite dall'utente;
- scegliere le tabelle iniziali (si ricorda che TEX permette il cambiamento degli stili e dei formati in qualsiasi puno del testo);
- dare tutte le istruzioni che riguardano il formattamento della pagina;
- inizio scrittura del testo;

- inserimento nel testo delle istruzioni per ottenere tutte le rifiniture desiderate nella stampa (ad esempio: inserimento di postille, numerazione e titolazione dei paragrafi, ecc.);
- fare elaborare la pagina dal sistema TEX;
- attendere eventuali messaggi di errore e correggerli;
- far stampare la pagina e controllare se il risultato ottenuto è quello desiderato;
- correggere le eventuali imperfezioni nel formattamento della pagina;
- memorizzare la pagina stampata per passare a quella successiva.

Una completa esemplificazione di come può essere ottenuta una pagina di stampa è riportata nelle pagine precedenti.

Le macrodefinizioni utilizzate sono state costruite dall'American Mathematical Society, e la pagina riportata appartiene ad un primo manuale diffuso da tale associazione.

BIBLIOGRAFIA

D.E. Knuth *"Mathematical typography"*, American Mathematical Society-Digital Press, 1979.

D.E. Knuth *"TAU EPSILON CHI - a sistem for technical text"*, Computer Science Department - Report No. STAN-CS-78-675, Stanford Artificial Intelligence Laboratory, September 1978.

D.E. Knuth *"METAFONT - a system for alphabet design"*, A.M.S. - Digital Press, 1979.

Robert A. Morris, *"An overview of TEX"*, UMASS/Boston, Boston MA 02125.

Il presente lavoro è stato svolto nell'ambito del CP Project tra l'Università di Milano, Istituto di Cibernetica e la Honeywell Information Systems Italia.

R. Palais S., *"A. TEX STATUS REPORT"*. TEX User Group, Nov. 1979.

METAFONT: UN LINGUAGGIO PER LA DEFINIZIONE SINTETICA DI ELEMENTI DI COMPOSIZIONE GRAFICA

G. Canzii, A. Pilenga
Istituto di Cibernetica, Università degli Studi di Milano

Riassunto

In un articolo precedente (Pixel n. 1-81) era stato descritto un sistema di composizione per la stampa di testi scientifici e tecnici chiamato TEX; proseguendo su questo tema in questo lavoro viene descritto un sistema per generare caratteri con computer.
La creazione e lo sviluppo di tale sistema indicato come METAFONT, si deve sempre, come nel caso di TEX, a Donald E. Knuth dell'università di Stanford.
Il META del nome sta a indicare che una generale spiegazione di come disegnare una fonte di caratteri trascende da ogni particolare insieme di disegni per quei caratteri.
Scopo principale di METAFONT è di generare la mappa di bit che riproducono esattamente il carattere disegnato. Tale mappa tiene conto del dispositivo di uscita impiegato, sia esso una stampante (elettrostatica o a laser) o un video. Questo significa che METAFONT è in grado di far avere diverse mappe di bit per lo stesso carattere; l'utente è quindi in grado di ottenere quella desiderata in considerazione della risoluzione della stampante o del video di uscita (per risoluzione di una stampante o di un video si intende la densità di punti che ha in un pollice lineare).
Nell'articolo sono illustrate, anche con esempi, le caratteristiche tipiche di METAFONT e le possibilità che offre all'utente.

INTRODUZIONE

Lo studio della costruzione dei caratteri, mediante regole geometriche, risale al 1509, quando l'italiano Luca Pacioli, nel "De Divina Proportione", incluse un'appendice sugli alfabeti.

Prendendo lo spunto da quello studio e da altri successivi, D. E. Knuth ha realizzato un sistema software (METAFONT), che permette di generare caratteri tramite computer.

In pratica un utente METAFONT scrive un programma per ogni carattere desiderato; tale programma viene scritto in termini di parametri variabili, in modo da ottenere una varietà di tipi dello stesso carattere cambiando semplicemente i valori dei parametri stessi (è evidente che può essere usato per definire un singolo carattere o una sola fonte).

Il presente articolo è stato pubblicato sulla rivista Pixel, n. 3, 1981

I programmi METAFONT sono espressi in un linguaggio dichiarativo algebrico, sviluppato essenzialmente per la soluzione di problemi grafici. In questo linguaggio l'utente specifica la locazione dei punti principali che compongono la figura, e specifica come deve essere disegnato il carattere voluto usando due entità software indicate come "pennini" e "gomme" (il significato di questi termini verrà spiegato nel seguito).

Uno dei vantaggi principali di METAFONT è che permette di ottenere ogni carattere in una forma estremamente precisa; infatti l'utente può apportare tutte le modifiche che ritiene necessarie per mezzo di successive approssimazioni semplicemente aumentando il numero dei punti per cui deve passare la curva (questo soprattutto nei punti critici della figura).

In matematica esiste un metodo pratico di approssimazione delle curve passanti per punti stabiliti. Tale metodo è quello degli "splines" e, in particolare per questo sistema, quello degli "splines cubici", ovvero polinomi di terzo grado della forma:

$$z(t)=a_0+a_1t+a_2t^2+a_3t^3$$

dove a_0, a_1, a_2, a_3 sono numeri complessi e t è un parametro reale. Vi sono vari metodi per determinare i coefficienti di tali polinomi. In quello seguito da Knuth per costruire le routine di METAFONT, stabiliti i punti z_1 e z_2 attraverso i quali passa la curva, l'equazione cubica risulta determinata completamente da z_1 e z_2 e dalla direzione delle tangenti in z_1 e z_2; il metodo di computazione per ottenere qualsiasi curva è così una sequenza di punti con le relative tangenti.

Nel caso in cui si desidera collegare con un arco di circonferenza un insieme di punti – cosa impossibile con un polinomio di 3° ordine – la soluzione è quella di forzare la relazione dell'equazione che esprime la forma della linea che connette i punti. È evidente che questi calcoli non sono semplici e in questa sede vengono omessi, poichè ciò che qui interessa mettere in evidenza è come l'utente di METAFONT vede il sistema. L'utente non deve infatti eseguire nessuna di queste operazioni, nè prendere in considerazione la relativa equazione.
Tutto questo procedimento è codificato in routine che vengono attivate nel momento in cui l'utente, ad esempio, cambia il formato del pennino, definisce i punti at-

traverso i quali deve passare la curva desiderata e le eventuali tangenti direzionali.

Utilizzando questo sistema Knuth e i suoi collaboratori hanno costruito le tabelle delle fonti di caratteri che vengono messe a disposizione degli utenti di TEX.

Il carattere che ha procurato maggiori problemi è stato la "S" (e il numero 8 che utilizza la stessa procedura), questo a causa del cambiamento di concavità.

È interessante notare come, variando soltanto un parametro, si può ottenere un insieme di "S" differenti tra loro. (fig. 1).

SSSSSSS

Fig. 1 - Differenti "S" ottenute variando le curvature in mezzo.

PENNINI E GOMME

Come precedentemente anticipato, METAFONT si serve di due entità di software, chiamate "pennini e gomme".

Tali entità costituiscono le caratteristiche principali del sistema: i pennini permettono all'utente di ottenere qualsiasi carattere e qualsiasi figura con la larghezza di inchiostro desiderato; le gomme permettono all'utente di cancellare le parti annerite con inchiostro all'interno delle figure. (fig. 2).

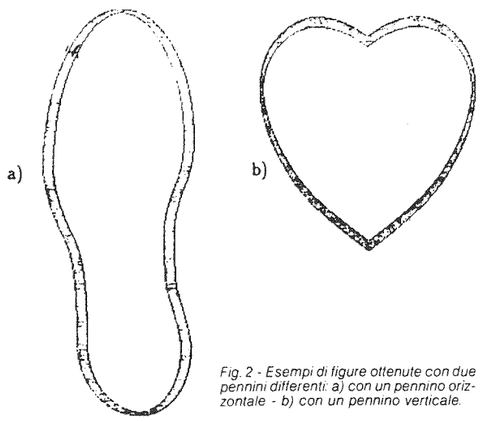


Fig. 2 - Esempi di figure ottenute con due pennini differenti: a) con un pennino orizzontale - b) con un pennino verticale.

Il sistema consente la scelta di sette diversi tipi di pennini e cioè:

cpen: "pennino circolare"; ha la sezione circolare infatti un punto disegnato con tale pennino è cerchio;
hpen: "pennino orizzontale"; ha la sezione di un'ellisse con la lunghezza dell'asse verticale fissa e quella dell'asse orizzontale stabilita dal valore del parametro "w" (larghezza);
vpen: "pennino verticale"; ha la sezione di un'elisse con la lunghezza dell'asse orizzontale fissa e quella dell'asse verticale variabile; (fig. 3)

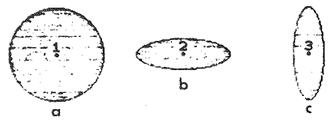


Fig. 3 - a) pennino circolare - b) pennino orizzontale - c) pennino verticale.

lpen: "pennino sinistro"; lascia il punto sulla sinistra di un rettangolo;
rpen: "pennino destro"; lascia il punto sulla desra di un rettangolo;
spen: "pennino speciale"; pennino di sezione ellittica;
epen: "pennino esplicito"; pennino di forma completamente arbitraria, stabilita a partire da parametri scelti dal disegnatore (fig. 4).

Fig. 4 - a) pennino sinistro - b) pennino destro - c) figura di un cuore ottenuta con un pennino esplicito (in questo caso di forma obliqua)

L'utente di METAFONT, una volta scelto il tipo di pennino, deve specificarne la dimensione; scegliere quella per la "cpen" è molto semplice; è sufficiente dare un valore w prima dell'istruzione di disegnare una curva tra i vari punti:

cpen; 5 draw 1 . . 4

In seguito verrà spiegato con esempi il significato di tali istruzioni. Importante è piuttosto notare la loro semplicità (in questo caso il cinque che precede l'istruzione draw è la larghezza in pixel, che è una unità di misura rivolta alla risoluzione della stampante).

Per le "hpen" e "vpen" l'utente dovrà assegnare subito il valore costante dell'altezza per la prima e della larghezza per la seconda, e soltanto quando utilizzerà tali pennini assegnerà loro il valore corrispondente alla grandezza variabile. Per le "lpén" e "rpen" si procede nello stesso modo delle "cpen". Per gli ultimi due pennini, che servono per ottenere curve particolari, la scelta delle grandezze per specificare la loro forma risulta più complicata, ma comunque sempre facile una volta acquisito un minimo di esperienza.

Il concetto di "gomma" è stato introdotto per permettere all'utente di poter cancellare l'inchiostro in parte di una curva precedentemente disegnata. Anche in questo caso è molto semplice scegliere il tipo desiderato (fig. 5) poichè ogni pennino può essere convertito in gomma semplicemente mettendo il simbolo "#" dopo il suo nome (per esempio, "cpen #" specifica una gomma circolare).

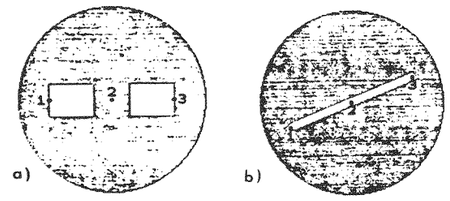
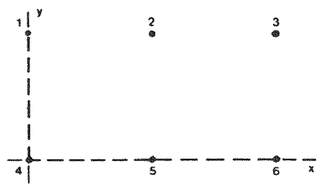


Fig. 5 - a) gomma sinistra e destra nel mezzo di un pennino circolare grande - b) una linea obliqua disegnata con una gomma triangolare

FONDAMENTI E CARATTERISTICHE DI METAFONT

Definizione della forma di un carattere

Per definire una forma o un carattere usando METAFONT, non occorre disegnarlo; è sufficiente spiegare come disegnarlo in modo del tutto generale, poichè la stessa spiegazione servirà per i relativi modelli, in circostanze differenti, variando opportunamente i parametri variabili. A questo scopo è necessario trovare un modo preciso per specificare i punti principali della figura. METAFONT usa le coordinate Cartesiane standard e quindi la locazione di un punto è definita specificando le sue coordinate x e y in numero di unità di punti rispetto al punto di riferimento. Per esempio i sei punti mostrati in figura:



hanno le seguenti coordinate:

$(x_1, y_1)=(0,100)$; $(x_2, y_2)=(100,100)$; $(x_3, y_3)=(200,100)$

$(x_4, y_4)=(0,0)$; $(x_5, y_5)=(100,0)$; $(x_6, y_6)=(200,0)$.

In una applicazione tipica di METAFONT viene preparata una bozza della figura che si vuole progettare su carta da disegno, si etichettano i punti principali della bozza con dei numeri, e si scrive un programma in cui si spiega come calcolare le coordinate dei punti e come disegnare le linee e le curve desiderate che passano attraverso questi punti.

Un programma METAFONT per un singolo carattere (più avanti sarà fornito un esempio di programma completo per definire un carattere) consiste di un insieme di istruzioni separate da punti e virgola e concluse con un punto.

La forma più comune di un'istruzione è un'equazione che esprime una o più relazioni algebriche tra le variabili.

Per esempio, per definire i sei punti della figura precedente è sufficiente scrivere le seguenti equazioni;

$$x_1=x_4=y_4=y_5=y_6=0;$$

$$x_2=x_5=y_1=y_2=y_3=100;$$

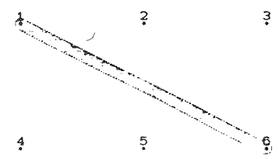
$$x_3=x_6=200;$$

Il passo successivo è di dare a METAFONT le informazioni sulle linee che collegando i punti formano il simbolo voluto. Per esempio si scrive l'istruzione

cpen; 9 draw 1 . . 6;

dove
- cpen si riferisce alla scelta del pennino più adatto per disegnare la figura (in questo caso un pennino circolare);
- 9 è la larghezza del pennino espressa in unità di raster (quindi dipende dalla risoluzione della stampante ottenere una linea più vicina al continuo o al discreto);
- draw 1 . . 6 è il comando che indica a METAFONT che nella figura una linea retta congiunge i punti 1 e 6.

Il risultato che si ottiene è il seguente:

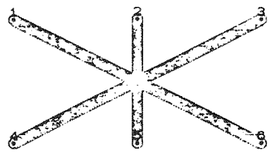


Un piccolo programma completo che congiunge in altro modo i sei punti definiti precedentemente è il seguente:

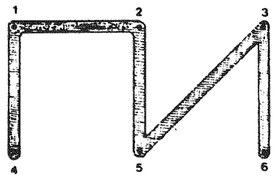
α
```
x1=x4=y4=y5=y6=0;
x1=x5=y1=y2=y3=100;
x3=x6=200;
```

β
```
cpen; 9 draw 1 . . 6;
       draw 2 . . 5;
       draw 3 . . 4;
```

Se non si cambia il tipo di pennino o il suo spessore non occorre ridefinirlo dopo la prima istruzione e si ottiene la seguente figura:



È evidente che si può ottenere un'altra figura con una riga di larghezza diversa cambiando lo spessore del pennino e le istruzioni della parte ß del programma. Se si vuole una riga di larghezza 4 e come risultato la seguente figura:



la parte ß del programma si riscrive così:

```
cpen; 4 draw 1 . . 4;
       draw 1 . . 2;
       draw 2 . . 5;
       draw 3 . . 5;
       draw 3 . . 6;
```

Ciò che è importante notare è che il centro del pennino va da punto a punto quando si disegna una linea, cioè quando è stata disegnata la linea da 1 a 6, tali punti non rimangono sul bordo della linea, ma al centro della posizione di partenza e arrivo.

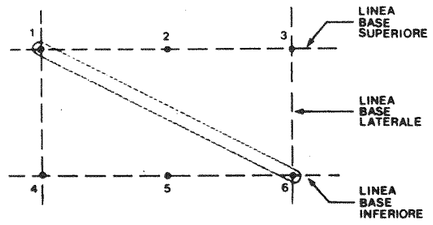
In poche parole, non si descrive tanto il contorno del carattere, quanto il movimento del pennino. Questo permette di ottenere figure dello stesso tipo, ma di larghezza di inchiostro diversa semplicemente cambiando la larghezza del pennino nell'istruzione relativa (come è stato fatto nell'ultimo esempio).

Dopo questa osservazione un disegnatore esperto potrebbe obiettare che la figura esce dalla linea base e le sue dimensioni sono leggermente diverse da quelle stabilite.

Se si prende dall'esempio precedente l'istruzione:

```
cpen; 9 draw 1 . . 6;
```

il risultato che si ottiene secondo il ragionamento precedente è il seguente:

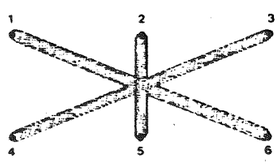


Come si può osservare il centro del pennino è esattamente sui punti 1 e 6 ma la riga disegnata supera di 4 unità questi due estremi (sempre nel caso di larghezza del pennino di 9 unità). Questo potrebbe far supporre che i risultati ottenibili non possano essere assolutamente precisi, come era stato precedentemente affermato.

METAFONT è però in grado di eliminare questo inconveniente perchè permette all'utente di specificare, con le due notazioni "bot" e "top", che il fondo del pennino nel punto 6 sarà 0 mentre il suo centro (se ad esempio è di larghezza 9) si troverà nel punto 5, e che la cima del pennino nel punto 1 sarà 100 mentre il suo centro si troverà nel punto 95. Utilizzando queste istruzioni il primo programma riportato come esempio sarà così modificato:

```
x1=x4=0;   x2=x5=d;   x3=x6=2d;
y1=y2=y3;  y4=y5=y6;
top1 y1=h;  bot1 y4=0;
cpen;   w1 draw 1 . . 6;   draw 2 . . 5;
draw 3 . . 4;
```

e il risultato sarà il seguente:



La figura ottenuta ha così esattamente le dimensioni assegnate indipendentemente dalla larghezza del pennino usato (è da osservare che nel programma non sono stati assegnati dei valori fissi ma dei parametri variabili, ciò vuol dire che cambiando il loro valore si ottiene un insieme delle figure senza dover modificare il programma che le genera).

Per risolvere il problema delle sbavature laterali si hanno altre due istruzioni: "lft" (per la parte sinistra), "rt" (per quella destra) che agiscono nello stesso modo di "top" e "bot".

Curve

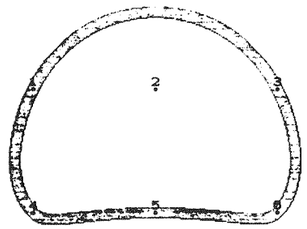
La definizione delle curve costituisce il maggiore problema nella costruzione del carattere.

Come è stato illustrato nell'esempio precedente, il comando "draw" produce la retta che congiunge due punti fissati. Ma, rispettando le regole geometriche, se si fissano almeno tre punti distinti nel piano lo stesso comando genererà una curva.

Riprendendo l'esempio precedente con gli stessi punti e le stesse coordinate, scrivendo l'istruzione:

```
cpen;   9 draw 5 . . 4 . . 1 . . 3 . . 6 . . 5
```

si ottiene la seguente figura:



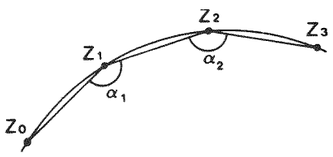
Le regole di METAFONT sono molto semplici e la curva attraverso due punti z_1 e z_2 dipende solo da

a) le coordinate di $z_1=(x_1, y_1)$;
b) le coordinate di $z_2=(x_2, y_2)$;
c) l'angolo della curva in z_1
d) l'angolo della curva in z_2.

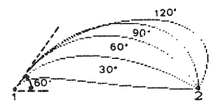
Con queste quattro informazioni, METAFONT è in grado di disegnare la miglior curva passante per questi punti.

Una proprietà importante è la seguente:
Ogni segmento di una curva disegnata da METAFONT dipende soltanto dalle coordinate dei due punti estremi e da quelle dei punti più vicini esterni all'intervallo di definizione della linea.

Per esempio se il segmento di una curva va da z_1 a z_2 e proviene da z_0 e prosegue in z_3, l'angolo in z_1 è determinato da z_0, z_1 e z_2 mentre in z_2 è determinato da z_1, z_2 e z_3.

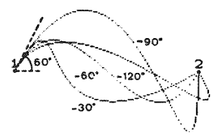


Per rendere più comprensibile quanto detto, si può prendere in considerazione la seguente curva:



Questa mostra come si può ottenere un insieme di curve che passano per gli stessi punti z_1 e z_2 ma partono tutte con lo stesso angolo di 60° in z_1 e arrivano in z_2 con angoli di 0°, 30°, 60°, 90° e 120° nell'ordine.

Nella seguente figura:



si può invece osservare come è possibile, con gli stessi punti dell'esempio precedente e lo stesso angolo di partenza in Z_1 di 60°, forzare le curve ad arrivare in Z_4 passando sotto l'asse orizzontale, dando valori negativi agli angoli in Z_2 (in questo caso di -30°, -60°, -90°, -120°).

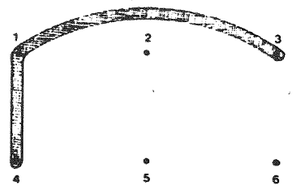
Verranno ora illustrati due metodi per specificare le pendenze nei punti limite.

Un metodo è di definire dei "punti nascosti" per il comando di "draw", come nel seguente esempio:

```
draw (5 . .) 4 . . 1 . . 3 (. . 6)
```

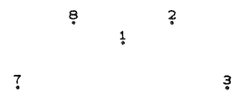
Il "(5 . .)" significa che METAFONT immagina che la curva passi per il punto 5 vicino a quello da dove inizia il comando di "draw" e che non si fermi al punto 3 ma

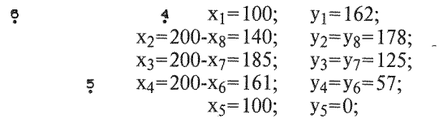
prosegua fino ad uno vicino (il 6). Il risultato che si ottiene è il seguente:



Il secondo metodo per specificare la pendenza di una curva è più flessibile rispetto al primo, in quanto si deve dichiarare la pendenza desiderata.

Per illustrare questo metodo si prenda ad esempio in considerazione la costruzione di un cuore. Il primo passo da compiere è quello di fare uno schizzo su carta millimetrata per poter fissare le coordinate dei punti principali attraverso i quali dovrà passare il contorno della figura. Quindi si considerino i seguenti punti con le relative coordinate:



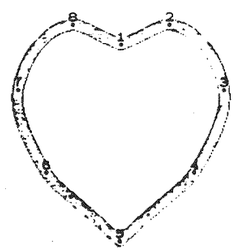


x_1=100; y_1=162;
x_2=200-x_8=140; y_2=y_8=178;
x_3=200-x_7=185; y_3=y_7=125;
x_4=200-x_6=161; y_4=y_6=57;
x_5=100; y_5=0;

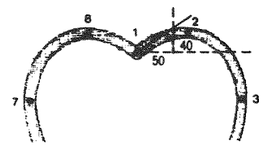
Un modo banale di comandare METAFONT per disegnare il cuore sarebbe il seguente:

```
cpen; 9 draw 1 . . 2 . . 3 . . 4 . . 5;
       draw 5 . . 6 . . 7 . . 8 . . 1;
```

ma non avendo specificato i punti terminali e le direzioni non si può ottenere un gran risultato, come è evidente:

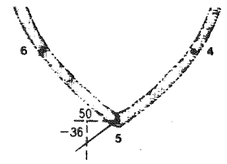


Un metodo più corretto per ottenere un buon risultato è invece il seguente. Si prende una riga e si traccia una linea sulla bozza su carta millimetrata nella direzione che la forma del cuore prende nel punto 1:



La pendenza desiderata può essere specificata semplicemente contando i quadretti. Il disegnatore trova che la linea si sposta di quaranta unità in alto e di 50 unità a destra, e la pendenza nel punto 1 è così determinata dalla coppia di numeri 50 e 40.

Nel punto 5, invece, la linea tracciata si sposta di 36 unità in basso e 50 a sinistra:



La pendenza in questo caso è specificata dai numeri -50 e -36 che indicano che è rivolta verso il basso a sinistra.

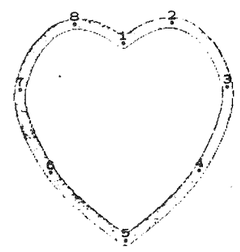
Nel programma METAFONT, per indicare queste pendenze, si scrive la coppia di numeri in parentesi graffe precedute dal numero del punto a cui si riferisce; le istruzioni dell'esempio precedente dovranno allora essere corrette in:

```
draw 1{50, 40}. . 2 . . 3 . . 4 . . 5{-50, -36}

draw 5{-50, 36}. . 6 . . 7 . . 8 . . 1{50, -40}
```

Rispettivamente per la parte destra e sinistra del cuore.

Il risultato che si ottiene con i comandi modificati è il seguente:



Come si può notare, il disegno ottenuto è senz'altro migliore rispetto alla prima esecuzione, tuttavia esistono ancora delle imperfezioni nella parte di curva che congiunge i punti 2, 3 e 4 a destra e 8, 7 e 6 a sinistra.

Un primo rimedio consiste nell'aumentare il numero dei punti nella parte di curva da correggere. In questo caso sarebbe sufficiente inserire il punto 9 (con relative coordinate) tra il 3 e 4, e il punto 10 tra il 6 e 7.

Ma, se è possibile, l'utente di METAFONT dovrebbe evitare di introdurre nuovi punti, poichè il sistema consente di utilizzare un'altra soluzione.

Osservando il cuore, si nota che i punti 2 e 8 sono quelli più in alto, che hanno cioè la coordinata y più grande, mentre i punti 3 e 7 sono quelli più esterni.

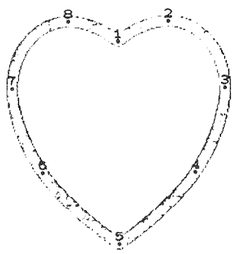
Quindi è possibile conoscere la loro corretta direzione: la curva è orizzontale nei punti 2 e 8 e verticale in 3 e 7.

METAFONT permette di specificare le direzioni della curva in tutti i punti fissati ed è perciò sufficiente modificare i comandi come segue:

```
draw 1{50, 40}. . 2{1, 0}. . 3{0, -1}. . 4 ... 5{-50, -36};

draw 1{-50, 40}. . 8{-1, 0}. . 7{0, -1}. . 6 . . 5{50, -36};
```

ed ottenere la figura finale del cuore nella forma ottimale desiderata:



UN ESEMPIO DI PROGRAMMA PER GENERARE CARATTERI

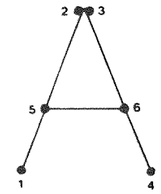
Analizzato, anche se brevemente, il funzionamento di METAFONT e le sue caratteristiche per generare qualsiasi carattere o disegno, si può esaminare l'esempio di programma completo per generare la lettera maiuscola "A" illustrato nel riquadro.

Verificando punto per punto il programma "The letter A" è il titolo del programma che identifica sinteticamente cosa si vuole ottenere.

"call charbegin ('A, 13, 2, 2, ph, 0, 0)" è la chiamata di una routine di inizializzazione. In essa si indica: il carattere, la sua larghezza in pixel, le eventuali correzioni rispetto al carattere serif, la sua altezza in pixel, la profondità rispetto alla linea base (come spiegato precedentemente) ed infine la correzione rispetto alla profondità se il carattere appartiene al tipo italico. I valori elaborati da questa routine permettono di tracciare il carattere nella propria "scatola", nella posizione desiderata e in modo più che corretto da un punto di vista tipografico. Con questa dichiarazione, METAFONT sa quale carattere ottenere, come centrarlo nella propria scatola, come disegnarlo.

Con "hpen" viene dichiarato quale tipo di pennino usare, ma non è stata ancora tracciata alcuna linea, nè potrebbe farlo poichè non conosce i punti per cui questa deve passare.

Nel programma segue infatti la definizione della posizione dei sei punti chiave individuanti la lettera (vedere nella figura i punti 1, 2, 3, 4, 5, 6).



I punti chiave sono gli estremi dei segmenti che raccordati in modo opportuno danno la forma desiderata al carattere.

Infine il programma utilizza routine particolari chiamate "serif routine", il cui scopo è quello di aggiungere alcuni dettagli al carattere migliorando l'effetto complessivo.

Il risultato che si ottiene usando queste routine è il seguente:

COME COSTRUIRE CON METAFONT LE TABELLE DELLE FONTI DI CARATTERE PER TEX

Il disegno di una fonte completa di caratteri è un sistema complesso; la procedura generale è di lanciare METAFONT con un comando tipo:

mode=<mode number; input font name>

che consente di scegliere automaticamente particolari routine che permettono di ottenere il carattere pronto per essere inviato ai diversi tipi di stampanti.

Il "modo 0" genera lettere che devono essere disegnate da stampanti con una risoluzione di 36 pixel per punto tipografico (~2600 pixel/pollice).

Il "modo 1" genera una fonte per stampante XGP con una risoluzione di 3,6 pixel per punto, memorizzando su un disco le mappe di bit delle lettere da stampare.

Il "modo 2" produce una fonte per stampante CRS con una risoluzione di 73, 7973 pixel per punto, (5300 pixel/pollice).

Attualmente il "modo 0" è poco usato per ottenere una fonte completa ed è invece impiegato nella creazione di nuovi caratteri.

Selezionando il "modo" più opportuno si può costruire un file che contiene le mappe dei bit dei caratteri compatibili con la risoluzione della stampante usata.

Affinchè TEX possa utilizzare i nuovi caratteri creati con METAFONT è necessario costruire una TFX, tabella usata da TEX per ogni fonte, che contiene tutte le informazioni necessarie al sistema per gestire le fonti.

Queste informazioni non sono altro che alcune istruzioni ovunque inseribili in un programma METAFONT dopo aver scelto il "tfxmode"; in questo modo si possono memorizzare tutte le informazioni necessarie al TEX per gestire sia le tabelle di caratteri che quelle matematiche.

BIBLIOGRAFIA

D.E. Knuth *"Mathematical typography"*, American Mathematical Society-Digital Press, 1979.

D.E. Knuth *"METAFONT - a system for alphabet design"*, American Mathematical Society - Digital Press, 1979.

D.E. Knuth *"The computer modern family of typefaces"*, Computer Science Department - Report No. STAN-CS-80-780, Stanford artificial Intelligence Laboratory, January 1980.

Il presente lavoro è stato svolto nell'ambito del CP Project tra l'Università di Milano, Istituto di Cibernetica e la Honeywell Information Systems Italia.

TEXLIDE: UN SISTEMA GRAFICO PER LA GENERAZIONE, LA MANIPOLAZIONE E L'ARCHIVIAZIONE DI TRASPARENTI PER LAVAGNA LUMINOSA

A. Avogadro di Vigliano, D. Marini, E. Paizis
Istituto di Cibernetica, Università degli Studi di Milano

1. INTRODUZIONE

Tra le applicazioni della "Computer Graphics" non è da tralasciare la costruzione di audiovisivi da utilizzare sia per l'istruzione e l'aggiornamento nelle scuole, sia per la formazione del personale e la presentazione di nuovi prodotti nell'industria. Attraverso sussidi didattici adeguati è possibile, infatti, migliorare in modo considerevole l'apprendimento, minimizzando lo sforzo dell'utente, sia esso il docente o l'allievo stesso.

In questo ambito il ruolo dell'immagine è fondamentale; del resto la pubblicità da anni usa l'immagine come catalizzatore dell'attenzione e ne sfrutta le proprietà evocative. Tuttavia, essa può venir usata in modi differenti; pertanto chi la costruisce dovrà esaminarne il significato nel contesto in cui verrà usata, tenendo presenti le esigenze sia di chi trasmette il messaggio, sia di chi lo riceve.

La costruzione di un'immagine associata ad un messaggio è, dunque, una attività complessa, non solo per le intrinseche difficoltà tecniche, ma anche perchè le esigenze dell'utenza possono essere diversificate e non sempre preconfigurabili con facilità. L'immagine dovrà essere realizzata ad un costo proporzionale a ciò che produce, tenendo conto del fatto che potrà essere usata non solo, come spesso avviene, a corredo di una esposizione verbale, ma anche come veicolo primario dell'informazione.

Tutte le tecniche comunicative, come l'uso del colore, di caratteri speciali, di spessore di tratto differenti, di disegni evocativi, etc., dovranno essere studiate perchè il messaggio risulti semplice ma efficace.

Tra i vari supporti tecnici disponibili per visualizzare un'immagine, la lavagna luminosa è risultata una dei più vantaggiosi nel contesto didattico analizzato (corsi e seminari universitari), in quanto mezzo economico e facilmente reperibile in molti ambienti. I trasparenti usati con tale mezzo possono venir proiettati in piena luce e permettono l'uso di varie tecniche di visualizzazione quali la rivelazione graduale degli argomenti, l'uso delle maschere, la sovrapposizione, ecc., elementi non trascurabili per la buona riuscita di una comunicazione. Al contrario, la produzione di filmati, diapositive e film strips richiede molto tempo e personale specializzato e ad un costo molto elevato, mentre l'uso di lavagne a gesso o a fogli mobili è molto economico ma senz'altro limitato perchè non permette accurata preparazione, nè conservazione delle immagini da proiettare.

Tuttavia, anche l'uso della lavagna luminosa presenta degli svantaggi quando si analizzano le tecniche di produzione dell'immagine da proiettare. La preparazione manuale dei trasparenti, infatti, richiede abilità grafiche da parte di chi li realizza o l'impiego di personale specializzato e quindi la produzione è, per forza di cose, limitata a causa della bassa qualità del risultato nel primo caso e dell'alto costo nel secondo. Le tecniche fotografiche, invece, producono sempre risultati di ottima qualità, ma hanno un costo molto elevato e non producono immagini adattabili alle esigenze dell'utente ma semplici copie, a meno di preparare l'originale a mano con tutti gli svantaggi che può comportare.

Dall'analisi di questa situazione è nata all'Istituto di Cibernetica dell'Università Statale di Milano, nell'ambito dei progetti di ricerca sulle tecnologie didattiche l'idea di realizzare, per mezzo di un microprocessore dotato di video grafico e di un plotter, un sistema per la produzione automatica interattiva di trasparenti per lavagna luminosa.

Il presente lavoro è stato svolto nell'ambito del progetto CP di collaborazione tra HISI e Istituto di Cibernetica. La strumentazione grafica adottata è stata acquisita nell'ambito del progetto di ricerca su Modelli Matematici e Strumenti Informatici per il Controllo delle Acque Potabili, finanziato dal Comune di Milano, Ripartizione Ecologica. È stato pubblicato negli atti del Convegno AICOGRAPHIC 81.

2. REQUISITI DEL SISTEMA

Nell'attuazione del progetto si sono tenuti presenti alcuni requisiti ritenuti fondamentali per il successo del prodotto, tra i quali principale è la facilità d'uso. L'utente, cioè, deve essere in grado di creare per mezzo della tastiera del microelaboratore il trasparente, senza un modello precedente o al limite con uno schizzo impreciso, e di modificare a piacimento il risultato, osservandolo sul video, senza possedere cognizioni di programmazione. Tutto ciò viene realizzato per mezzo dell'interattività e di un manuale d'uso, compreso nel sistema stesso, che può essere richiamato durante una qualunque delle fasi di produzione. L'uso del microprocessore e del plotter garantisce, inoltre, la qualità del risultato e la sua velocità di realizzazione; tuttavia, si è pensato di differenziare la produzione in due diversi livelli: realizzazione molto veloce con caratteristiche standard e realizzazione meno veloce con caratteristiche scelte dall'utente.

La memoria ausiliaria può, inoltre, venir usata per la creazione di un archivio momentaneo o permanente dei trasparenti generati.

In sintesi il sistema mette a disposizione dell'utente programmi che gli permettono di creare un testo o un disegno inserendo i dati per mezzo della tastiera, di rivedere una pagina già generata per controllarla o per apportarvi delle modifiche, di archiviare una pagina, di visualizzare una pagina presente in archivio e di tracciare un trasparente su carta normale a titolo di prova o direttamente su carta lucida adatta alla proiezione. L'implementazione della procedura ha previsto in un primo tempo la sola generazione di testi a cui, in una fase successiva è stata aggiunta la possibilità di realizzare diagrammi di flusso, reti di Petri e disegni liberi ottenuti mediante concatenazione di segmenti.

3. DESCRIZIONE DELL'HARDWARE

Per la realizzazione di questo progetto è stato utilizzato il microelaboratore 4052 Tektronix, nella configurazione a 56K di memoria utente, dotato di video grafico di tipo "storage", particolarmente adatto per il tracciamento di disegni anche complicati, di tastiera alfanumerica con chiavi definibili dall'utente e di unità a nastro magnetico incorporata. A tale microelaboratore è stato collegato, con interfaccia IEEE 488, il plotter 9872B Hewlett Packard a quattro penne, con possibilità di montare pennini per carta da proiezione. A tale fine è stato sviluppato anche un semplice "driver" per tradurre i comandi grafici del linguaggio Tektronix in quelli del processore del plotter HP.

Il linguaggio con cui è stato realizzato il programma è il Basic del Sistema Grafico Tektronix.

4. REALIZZAZIONE DEL SOFTWARE

Per realizzare un sistema dotato dei requisiti sopra elencati, sono stati esaminati gli elementi che compongono un trasparente, suddivisi nelle seguenti categorie: testo, disegno, spazi vuoti, evidenziature ed impaginazione. Ognuno di questi elementi contribuisce alla formazione della pagina in modo tale che le informazioni in essa contenute risultino chiare, concise ed essenziali.

Dal punto di vista fisico, il trasparente viene costruito su fogli di acetato le cui dimensioni rispettano, più o meno, il formato standard A4. Chiameremo questa dimensione "area fisica". La variabilità del formato non rappresenta un problema dal momento che le dimensioni del piano delle lavagne luminose più diffuse sono standard e quindi costringono a definire, all'interno dell'area fisica un'"area utile" di dimensioni fisse.

Ogni pagina è composta a sua volta da elementi base o primitive, componenti un menù da cui l'utente può estrarre gli elementi a lui utili. A tali primitive vengono associati attributi che ne definiscono le proprietà e che possono venir modificati per mezzo di funzioni. Ad esempio, sono primitive i caratteri, le stringhe di caratteri, i segmenti, le figure, etc... Sono invece attributi la dimensione, il colore, lo spessore del tratto, etc... Sono funzioni le modifiche dei valori degli attributi, le trasformazioni delle primitive che permettono impaginature particolari, etc...

Una pagina può venir considerata essa stessa una primitiva alla quale vengono associati i seguenti attributi: definizione della spaziatura tra una riga e l'altra, ovvero l'interlinea, centratura delle singole righe o impaginazione a bandiera, incorniciatura, numerazione pagine, allineamento dei disegni.

L'interlinea deve essere definita come funzione della dimensione dei caratteri. La centratura del testo permette di riempire con densità differenti una pagina senza che rimangano antiestetici spazi vuoti. Tra tutti questi attributi della pagina sono assegnati per difetto: l'impaginazione a bandiera, l'interlinea automatica; tutti gli altri sono opzionali.

4.1. AREA UTENTE E FASI DI PRODUZIONE

Per poter localizzare esattamente una primitiva all'interno dell'area utile è necessario ricoprire tale area con una griglia immaginaria che permetta l'indirizzamento ad una posizione precisa. In questo modo, si può associare ad ogni primitiva una coppia di coordinate che ne definisce univocamente l'ubicazione (ad es.: ad ogni primitiva si può associare una determinata coordinata iniziale).

Stabilito in questo modo un sistema di riferimento, è possibile suddividere la produzione di un trasparente in quattro fasi: progettazione, realizzazione su video,

modifica e rifinitura, esecuzione su plotter (Fig. 1). In fase di progettazione si crea uno schizzo, appena abbozzato su un foglio di carta, della pagina da costruire. In fase di realizzazione si inseriscono i dati da tastiera affinchè vengano elaborati dal sistema che presenta la sua "versione" sul video. Questa versione può essere modificata a piacimento mediante funzioni richiamabili con tasti chiave. Una volta completato, il contenuto di una pagina può essere inviato al plotter per la stampa e/o archiviato in modo permanente su cassetta magnetica. In fase di tracciamento vengono definiti gli attributi della pagina. Questo modo di procedere comporta il vantaggio di realizzare l'immagine molto rapidamente e di modificarla anche molte volte senza spreco di carta o acetati. Inoltre, la possibilità di creare un archivio rende molto semplice e veloce sia la riproduzione, sia la creazione di una vera e propria biblioteca di trasparenti.

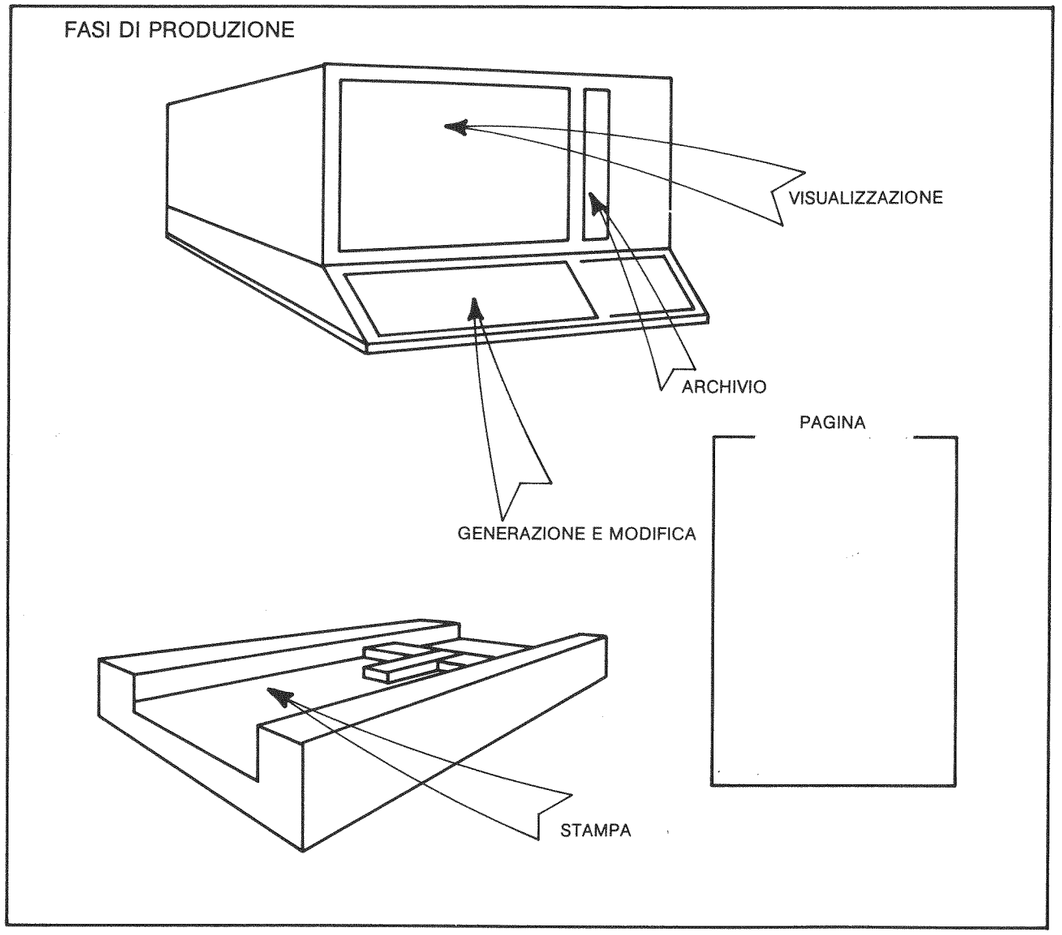
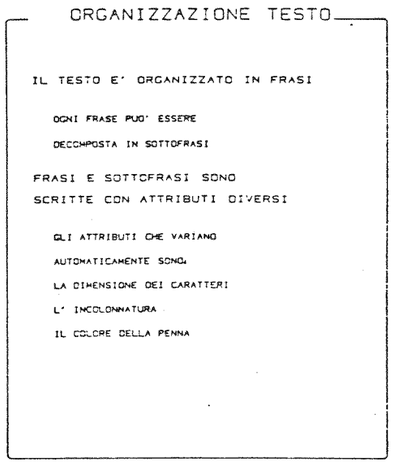


FIG. 1 La realizzazione di una pagina

4.2. GENERAZIONE DI UN TESTO

Sono elementi fondamentali di una pagina le parole, raggruppate in tre categorie principali: titolo, frase, sottofrase. Alla prima categoria appartengono insiemi di parole che definiscono sinteticamente il contenuto della pagina. Alla seconda, insiemi di parole che definiscono gli argomenti principali del discorso. Alla terza, infine, appartengono insiemi di parole che rappresentano argomenti secondari o liste di informazioni (Fig. 2). La definizione di tali elementi permette all'utente l'inserimento del testo indipendentemente dall'impaginatura; basterà individuare la categoria di appartenenza di un elemento perchè a questo venga automaticamente associata una determinata posizione e determinate caratteristiche.

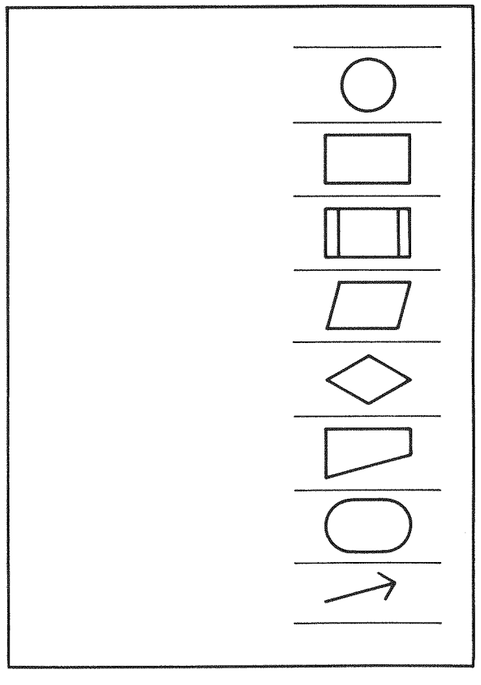


Ogni elemento del testo è poi dotato di differenti proprietà, che ne definiscono le caratteristiche. Si possono associare molti attributi alle primitive al fine di aumentare le possibilità creative dell'utente; tuttavia è necessario tener presente che il tempo di esecuzione e l'occupazione di memoria incrementano notevolmente il costo di produzione. Una soluzione equilibrata è quella di definire, per ogni primitiva i seguenti attributi: colore della penna, dimensione dei caratteri (orizzontale e verticale), sottolineatura e tipo di sottolineatura, inclinazione dei caratteri, posizione (in coordinate X e Y). Ogni attributo può venir associato automaticamente alle primitive secondo valori per difetto con molti vantaggi nella realizzazione; infatti può accadere che la pagina generata in questo modo soddisfi le esigenze di chi l'ha prodotta o che le modifiche da attuare non siano molte e che, di conseguenza, tutta la procedura diventi molto veloce. I parametri per difetto possono venir definiti dal sistema a priori o decisi dall'utente stesso.

Ogni attributo deve poter essere modificato in modo da soddisfare le esigenze degli utenti. Per questo si possono definire funzioni di trasformazione che operino sugli attributi o sulle primitive stesse. Le funzioni inserite in questo lavoro sono: definizione e tipo di sottolineatura, modifica dell'inclinazione dei caratteri, variazione della posizione di un elemento all'interno della pagina, modifica dell'altezza e della larghezza dei caratteri, centratura di una riga, modifica del colore e correzione di eventuali errori. Tutte queste operazioni producono la reimpaginazione automatica del testo stesso.

4.3. GENERAZIONE DI UN DISEGNO

Nell'ambito della generazione di disegni sono elementi fondamentali di una pagina: il titolo, le primitive grafiche, eventuali scritte libere, o vincolate e i segmenti con cui sono composti disegni liberi. Le primitive grafiche sono elementi geometrici di base, che compongono un menù a cui l'utente fa riferimento ogni volta che definisce un elemento del diagramma che vuole generare.



I segmenti, invece, possono essere di lunghezza e direzione variabile e vengono realizzati per mezzo del movimento del cursore sul video associato ad opportuni comandi di tracciamento. Alcune funzioni facilitano il tracciamento di un disegno preciso. Alla categoria delle scritte libere, appartengono insiemi di parole posizionabili in qualunque punto della pagina al di fuori delle aree occupate dai disegni. Alla categoria delle scritte vincolate, invece, appartengono insiemi di parole posizionate all'interno delle primitive grafiche e che quindi sono vincolate a non uscire dai confini dei disegni geometrici.

Ad ogni primitiva grafica sono associati punti particolari, classificabili in tre categorie: centroide, punto di riferimento, punti di connessione. Il centroide è un punto nelle vicinanze del baricentro di ogni primitiva e serve per il suo riconoscimento all'interno di una pagina. Il punto di riferimento è il punto da dove inizia il tracciamento del disegno, mentre i punti di connessione, in generale più di uno, servono per collegare in modo preciso i disegni mediante frecce.

A ciascun tipo di primitiva sono associati, come nella generazione di testi vista sopra, attributi che ne definiscono le caratteristiche, tra cui il tipo di linea, il colore, e, nel caso dei caratteri, l'inclinazione, la larghezza e l'altezza, che possono venir attribuiti automaticamente secondo valori per difetto, oppure ridefiniti dall'utente. È inoltre possibile, dopo aver costruito una pagina, cancellarne degli elementi, aggiungere disegni o scritte, allineare in senso orizzontale o verticale le primitive. Per quanto riguarda la cancellazione di elementi va sottolineato il fatto che può essere cancellato un solo elemento per volta oppure una serie di elementi corrispondenti a tutti quelli che seguono il disegno da cui si inizia la cancellazione. Per quanto riguarda il disegno libero si hanno a disposizione funzioni che permettono il posizionamento preciso del cursore agli estremi delle figure disegnate.

5. ORGANIZZAZIONE DELLE PROCEDURE E SCRITTURA DATI

La figura 4 mostra l'organizzazione delle procedure del sistema, cui è stato dato il nome di TEXLIDE.

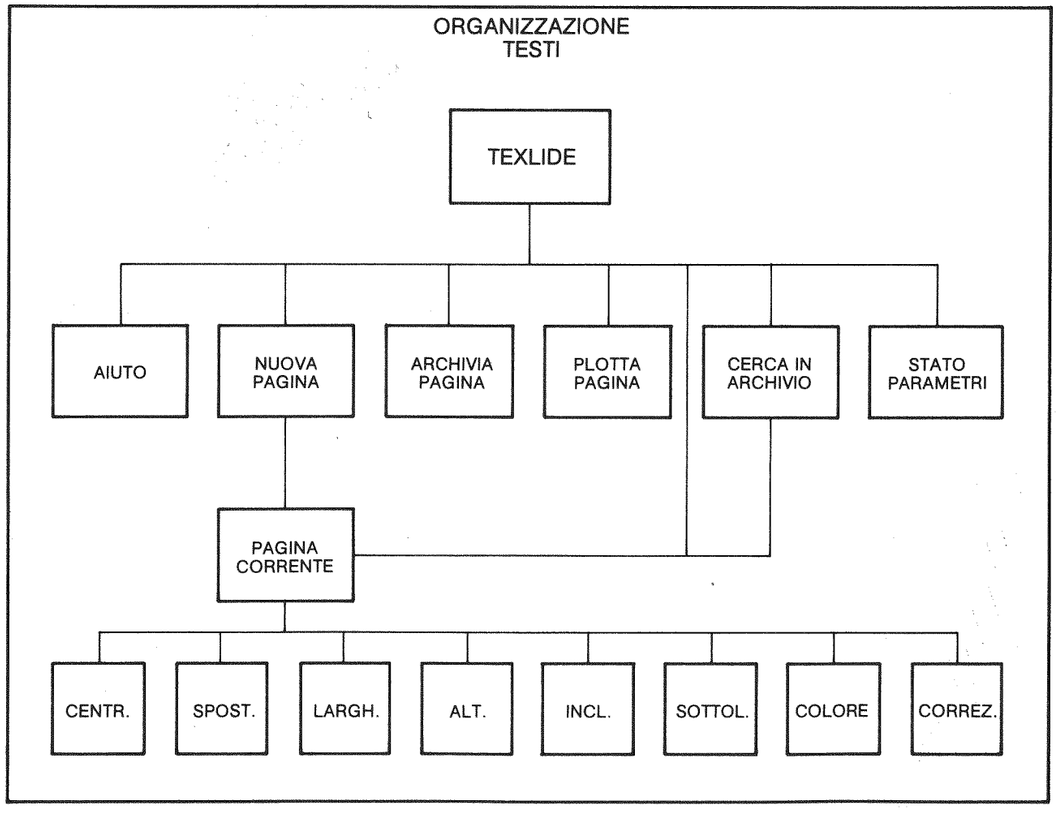


FIG. 4 La struttura di TEXLIDE

5.1. GENERAZIONE TESTI

I dati che l'utente inserisce da tastiera in fase di generazione di un testo vengono archiviati in due tabelle chiamate FRASI e PAROLE. La tabella FRASI contiene informazioni sulle coordinate iniziali e finali di ogni frase e due puntatori che indicano la prima e l'ultima parola della frase nella tabella PAROLE. La tabella PAROLE contiene l'elenco delle parole che compongono il testo seguite dagli attributi (colore, larghezza caratteri, etc.). Tali attributi vengono inizialmente assegnati in modo automatico alle parole, secondo valori standard archiviati in un'altra tabella. Durante la modifica del testo e dei suoi attributi le tabelle FRASI e PAROLE vengono continuamente aggiornate.

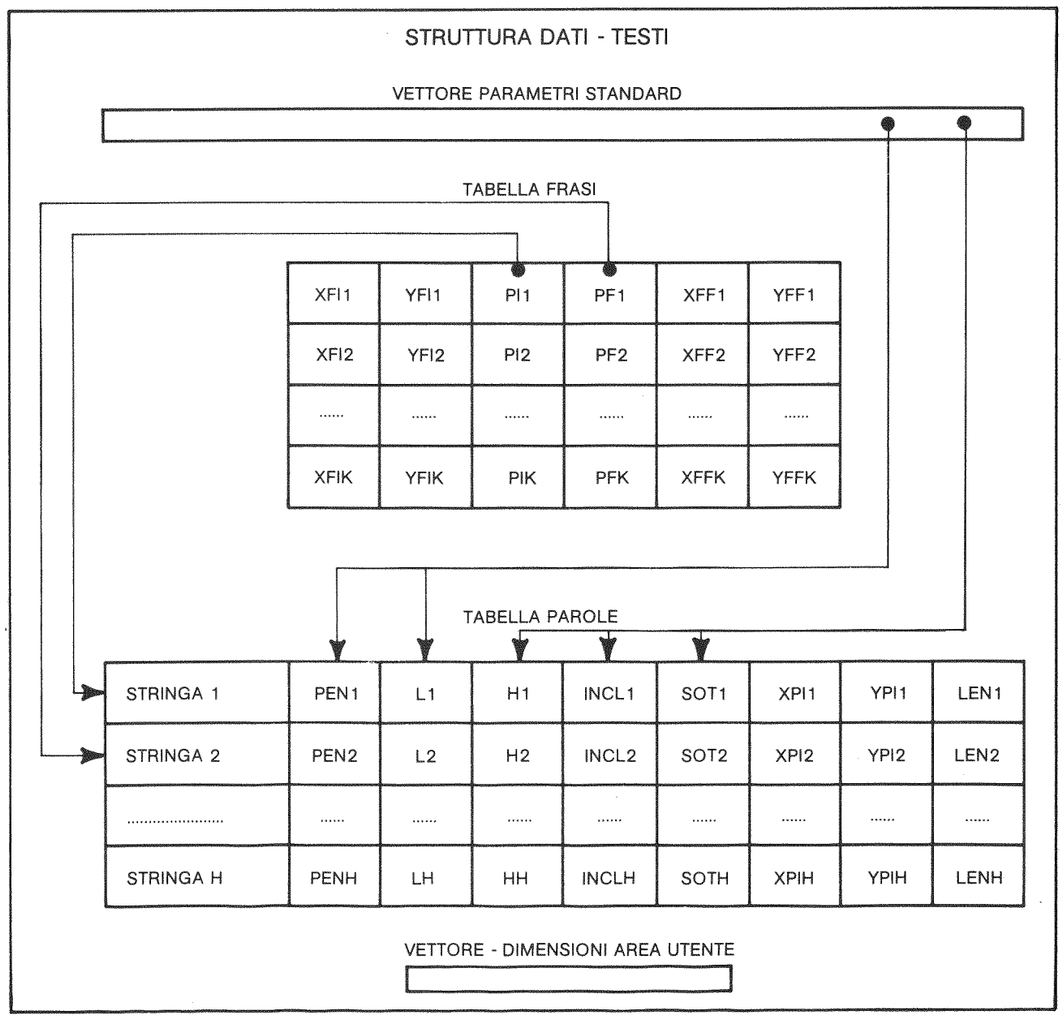


FIG. 5 La struttura di un testo

Il vettore Parametri Standard contiene i valori per difetto degli attributi delle primitive: le coordinate iniziale e finale del titolo, delle frasi, delle sottofrasi, il colore, la dimensione e l'inclinazione dei caratteri.

5.2. GENERAZIONE DISEGNI

Per quanto riguarda i disegni, il programma è suddiviso logicamente nelle seguenti fasi:

- generazione di un disegno
- editing di un disegno già generato
- gestione archivio dei disegni
- programma di stampa con il plotter

Nel programma di editing del disegno sono compresi sottoprogrammi che permettono le modifiche, come l'aggiunta, la cancellazione e l'allineamento di primitive, il cambiamento dei parametri, l'inserimento di scritte, ed inoltre quelli che governano il movimento del cursore all'interno della pagina per tracciare disegni liberi.

La struttura concettuale dei dati è costituita da due liste concatenate.

I nodi della prima lista sono gli elementi del disegno: o le primitive disponibili a menu o le poligonali costruite nel corso del disegno libero. La seconda lista ha come nodi le coordinate degli elementi costruiti col disegno libero. I parametri delle primitive sono conservati in una lista separata.

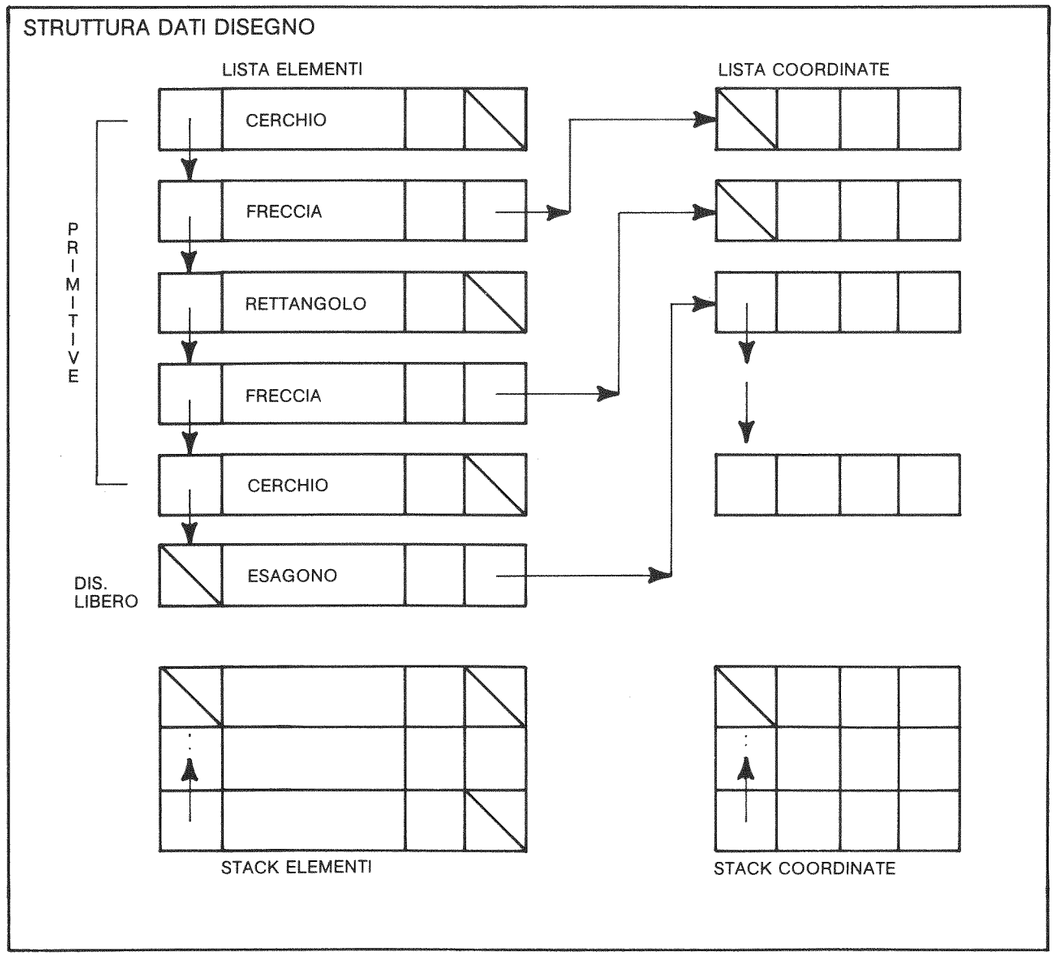


FIG. 6 La struttura dei dati del disegno di Fig. 7

L'operazione di ricerca di un elemento viene effettuata percorrendo la lista 1, seguendone le concatenazioni. Le operazioni di inserzione e cancellazione si avvalgono di 2 stack, uno per ciascuna delle 2 liste da cui vengono prelevati o rimessi i nodi. Pertanto una operazione di modifica di un nodo comporta la ricerca, la cancellazione del vecchio nodo e la immissione di un nuovo nodo prelevato dallo stack (cfr. Fig. 6).

Le scritte introdotte nel disegno sono considerate attributi degli elementi.

Fisicamente queste strutture sono realizzate con arrays residenti in memoria.

La creazione di un disegno più complesso può comportare la creazione di più liste la cui gestione è governata da un altro stack. La possibilità di gestire disegni complessi con insiemi di liste, consente di manipolare separatamente porzioni differenti del disegno.

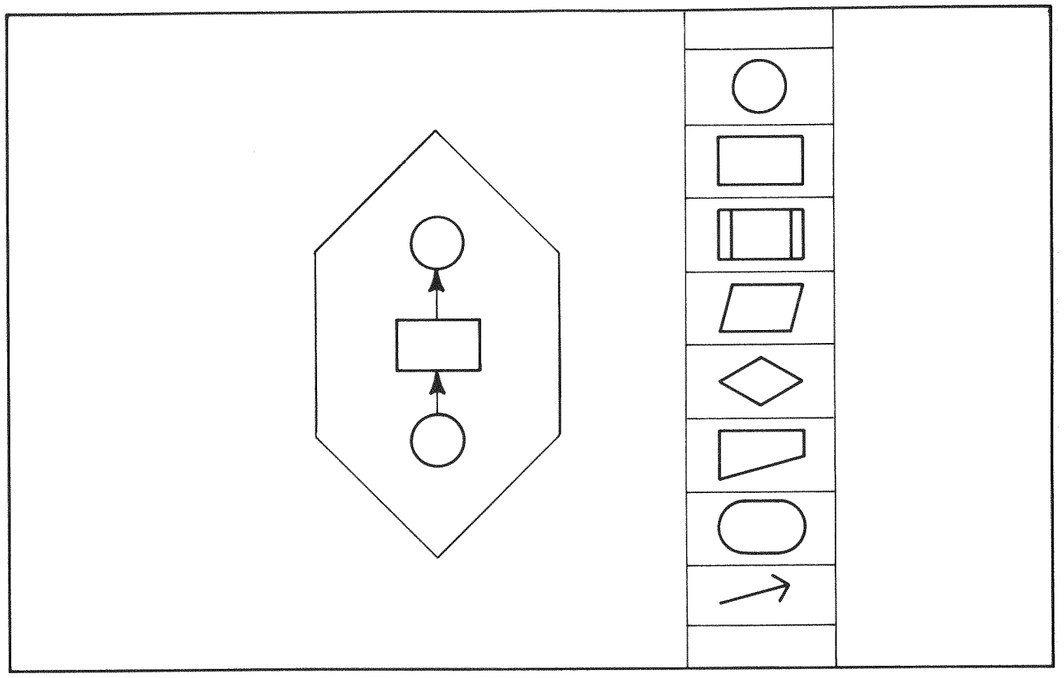


FIG. 7 Un semplice esempio

6. CONCLUSIONE

Lo scopo fondamentale di questo lavoro è quello di creare un sistema che metta in grado qualunque docente o una qualunque unità didattica di creare materiale audiovisivo personalizzato.

Le prime esperienze condotte per collaudare il sistema hanno mostrato che, dopo un breve periodo di addestramento (circa una giornata), il tempo medio di produzione di un trasparente costituito soltanto da testi è dell'ordine di 15 minuti, mentre per un trasparente contenente disegni complessi il tempo di produzione non è superiore ai 30 minuti.

Il sistema illustrato soddisfa alle esigenze fondamentali, tuttavia può, senza dubbio, essere ampliato con l'introduzione di molte altre procedure per la realizzazione di grafici e diagrammi particolati o per la gestione di tabelle.

All'Istituto di Cibernetica è già in fase di studio l'inserimento di funzioni che permettano di operare trasformazioni sulle primitive, come ad esempio, la rotazione, la traslazione, il cambiamento di scala.

7. BIBLIOGRAFIA

(1) Morvan, P., Lucas, H., *"Images et ordinateur: Introdution à l'infographie interactive"*, Larousse, Paris, 1976

(2) Newman, W.N., Sproull, R.S., *"Principles of interactive computer graphics"*, McGrow-Hill Book Company, U.S.A., 1973.

(3) Klinger, A., Fu, K.S., Kunii, T.L., *"Data Structures, Computer Graphics and Patter Recognition"*, Academic Press, Inc., U.S.A., 1977

(4) Rumelhart, D.E., *"Introduction to human information processing"*, John Wiley & Sons, U.S.A., 1977

PAGINAZIONE PER UN SISTEMA DI CONSULTAZIONE DI TESTI

Piercarlo Grandi
Istituto di Cibernetica - Università degli Studi di Milano

Riassunto

Un sistema per la consultazione guidata di testi (Libro Elettronico) deve assumere che i testi sono in genere troppo lunghi per poter essere memorizzati interamente nella memoria centrale del calcolatore utilizzato; deve quindi memorizzarli per la maggior parte su memoria di massa a basso costo, e adottare una opportuna strategia per minimizzare i tempi medi di accesso ad essa, utilizzando al meglio la memoria centrale. Il problema è notevolmente influenzato dalla natura particolare della applicazione, il che consente di inglobare nella strategia una considerevole quantità di informazione a priori.

INTRODUZIONE

Un sistema per la consultazione di testi deve assumere che la quantità di testo sia superiore alla capacità della memoria centrale disponibile, e che quindi solo una parte del testo potrà essere tenuta in una memoria ad accesso immediato; si pone quindi il problema di cercare di minimizzare il tempo medio di accesso a una pagina di testo, tenendo conto del fatto che inevitabilmente alcune pagine dovranno essere ottenute da una memoria di massa il cui tempo d'accesso è assai più lento della memoria centrale.

Oltre a minimizzare il tempo medio d'accesso è anche utile minimizzare il tempo massimo d'accesso, poichè da studi di psicologia effettuati, se il sistema risponde alle richieste dell'utente erraticamente e peggio talvolta troppo lentamente, l'utente perde rapidamente in concentrazione e attenzione.

Avendo presenti i tre fattori limitanti (lentezza della memoria di massa, minimo tempo medio, minimo tempo massimo), la natura dell'applicazione ci consente di sfruttare le sue peculiari e ben prevedibili caratteristiche per tener conto di alcuni fattori favorevoli.

Ci si può aspettare che la gran parte degli accessi a pagine del testo avvengano più frequentemente in una ben specifica sequenza con occasionali cambiamenti di sequenza e/o periodi di salti arbitrari fra le pagine.

Infatti i testi consultati hanno in genere una qualche struttura logica che si riflette nella sequenza numerica delle pagine benchè esistano esempi contrari quali i dizionari ad esempio), con eventuali riferimenti ad altri testi, a loro volta in genere strutturati nella stessa maniera. Vi sono anche momenti in cui la consultazione del testo è circa casuale, ad esempio quando si sta cercando un riferimento.

È inoltre possibile che più utenti consultimo in multiprogrammazione uno stesso sistema di testi; questo complica in qualche misura il problema.

LA PAGINAZIONE

Si potrebbe pensare che questi problemi siano in qualche misura analoghi a quelli della paginazione della memoria virtuale; benchè esista una qualche somiglianza, di cui trarremo vantaggio, sono evidenti alcune importanti differenze:

* L'accesso al testo segue andamenti ben più limitati di quelli di un programma generico ai suoi dati.

  moria centrale è ben superiore a quello fra memoria virtuale e memoria reale.
* Il rapporto fra quantità di testo e capacità della me-
* Un sistema di consultazione di testi ha tempi di attesa per la computazione relativa a una pagina (consultazione da parte dell'utente) superiori di diversi ordini di grandezza al tempo in cui un tipico programma elabora i dati contenuti in una pagina di memoria virtuale; in altre parole, il colo di bottiglia del sistema è la velocità dell'utente nel leggere il testo, mentre per la memoria virtuale è la velocità nel portare la pagina in memoria.

* Difficilmente si può far conto per i testi della località di riferimento (ripetuto uso delle stesse pagine in un certo intervallo di tempo) comune invece per la memoria virtuale. Tutto questo ci induce a pensare che benchè nella struttura generale la soluzione al nostro problema userà dei meccanismi simili a quelli di una

memoria virtuale, la politica di decisione di quali pagine portare a tenere in memoria centrale sarà alquanto diversa.

LA POLITICA DI PAGINAZIONE

In virtù dei punti precedentemente esposti e dei nostri obiettivi è ovviemente inadeguata una politica "demand driven" (demand driven è l'atteggiamento di paginazione della memoria virtuale per il quale non si fa alcun tentativo di prevedere quali pagine verranno richieste, bensì si aspetta che una pagina venga usata da un programma per caricarla in memoria, eventualmente rimpiazzando una di quelle che ci sono già e sono poco usate) collegata all'idea di "working set" (working set è l'insieme di pagine che ad un dato istante sono così frequentemente usate da dover essere in memoria a pena di doverle andare a prendere da memoria di massa in continuazione).

Infatti non solo una politica demand driven ci porterebbe in alcuni casi a tempi d'attesa inutili, visto che non fa uso del fatto che possediamo informazioni a priori sul probabile andamento degli accessi al testo, il che ci consente invece di utilizzare tecniche predittive e quindi prepaginare in memoria centrale quelle pagine di testo che in base alle nostre informazioni a priori verranno probabilmente accedute dopo quella correntemente consultata.

Nasce ora il problema di scegliere in quale maniera incorporare le nostre conoscenze a priori nell'algoritmo di paginazione. Abbiamo almeno tre scelte:

stocastica
il sistema dinamicamente mantiene un indice delle N pagine più frequentemente accedute in successione associato a ogni pagina di testo (magari con le relative frequenze), e ogni volta che una pagine viene consultata, pre pagina le K pagine ($K < N$) più opportune;

guidata
opportune direttive associate a ogni pagina o blocco di pagine o date direttamente dall'utente durante la consultazione guidano la scelta delle pagine da tenere in memoria e da pre paginare; tali direttive possono essere del tipo "accesso sequenziale", "accesso arbitrario", "accesso a blocchi".....

predeterminata
si assume che prevarrà una specifica modalità di accesso e la si incorpora staticamente, essa sola, nell'algoritmo di decisione.

Osserviamo subito che una politica predeterminata, mentre verosimilmente è sensata ed efficiente nel senso di produrre un buon tempo medio per gli accessi alle pagine, non soddisfa il nostro criterio di avere anche un buon tempo massimo, dato che nel caso peggiore, in cui l'accesso effettivo sia diverso da quello prevalente come codificato nell'algoritmo, avremmo una effettiva pessimizzazione dell'efficienza.

Una scelta guidata da direttive evita questo problema, ma incorre in un altro spesso sottovalutato, cioè che molto spesso l'uso inteso di un testo è assai lontano da quello che poi si stabilisce nella pratica, col che le direttive dovrebbero essere periodicamente riviste a mano, in maniera euristica.

Ci rimane quello che sembra essere il più complesso sistema, quello di mantenere delle statistiche di accesso in modo da poter fare una scelta basata su criteri continuamente variabili. Potrebbe sembrare che un simile livello di generalità sia eccessivo, e in effetti, fosse solo per guidare l'algoritmo di paginazione, mantenere statistiche sugli accessi è probabilmente non molto utile; ma se a questo si aggiunge che tali statistiche sono interessanti di per sè, a scopo analitico e magari anche di contabilizzazione, la prospettiva cambia.

Possibili perfezionamenti

Rimane da osservare che in realtà la distribuzione dei modi accesso a un testo non è funzione della solo struttura del testo, come abbiamo assunto implicitamente finora, ma anche della logica dei programmi che l'utente utilizza per consultare il testo, e della logica con cui l'utente consulta il testo. Ad esempio un dizionario verrà normalmente impiegato per cercare parole sparse, ma nulla vieta che un utente lo legga tutto a partire dalla lettera A o che un programma di ricerca lessicale lo scansioni per intero. In tali casi, è probabilmente utile darre all'utente e ai programmi la possibilità di far pesare nell'algoritmo di paginazione anche direttive o suggerimenti per dare al sistema di consultazione informazioni per una migliore predizione dei futuri accessi alle pagine.

LA MULTIPROGRAMMAZIONE

La soluzione proposta (statistiche dinamiche) ha anche il pregio di adattarsi meglio delle altre al caso di multiprogrammazione, in quanto tenderà ad assegnare statistiche di alto utilizzo a quelle pagine di testo più frequentemente consultate dalla *maggior parte* degli utenti, e quindi ad esempio tenderà a tenere in memoria le parti di più frequente consultazione a livello globale.

CONCLUSIONI

Risulta inappropriato usare le tecniche usuali di paginazione adottate per la memoria virtuale, poichè nel caso di accesso a testi ci aspettiamo scarsa località di riferimento e disponiamo di abbondanti informazioni a priori, il che ci consente e suggerisce di adottare politiche predittive (in senso euristico) di paginazione. Fra queste la più flessibile, basata su statistiche di accesso

mantenute dinamicamente, è anche la più interessante e probabilmente la più efficiente e si adatta bene anche al caso di multiprogrammazione.

Rimane da osservare che verosimilmente il miglior uso delle statistiche di accesso fra le pagine non è solo quello di determinare quali portare in memoria, ma anche di tenerne conto al momento di disporre o ridisporre le pagine di testo sulla memoria di massa, per la quale in genere non è del tutto indifferente ai fini del più breve tempo di accesso l'ordine in cui le pagine sono memorizzate.

CONTRIBUTI TECNICI

ASPETTI DI ARCHITETTURA SOFTWARE NELLA SIMULAZIONE DISCRETA FUNZIONALE DI COMPLESSI SISTEMI REAL-TIME

S. Mussi
(Cilea, Segrate)

Riassunto

Vengono esposte le linee fondamentali dell'architettura di un programma interattivo per la simulazione funzionale discreta di un sistema real-time i cui componenti sono task che dialogano funzionando in parallelo ed in modo asincrono e quindi realizzando l'interfogliamento delle evoluzioni dei processi contemporaneamente in vita nel sistema.
Lo studio si appoggia ad un esempio che pur non essendo in se stesso un caso reale tuttavia si ispira a sistemi reali e possiede quelle caratteristiche di semplicità e di rappresentatività che lo rendono un supporto significativo e necessario allo sviluppo di considerazioni più generali.
Nell'ultima parte si mostra come misurare sul modello di simulazione alcuni indici di prestazione.

Abstract

This paper shows the fundamental aspects of the architecture of an interactive program for functional discrete simulation of real time systems, whose components are tasks which communicate and work in a parallel and asinchronous way carrying out an interleaving processes system.
This study is supported by an example which, even if not a specific real case, however refers to real systems and it is simple and meaningful enough to be considered a useful tool to develop more general considerations.

1. INTRODUZIONE

Nello sviluppo di sistemi real-time, anche molto complessi come possono essere per esempio i sistemi distribuiti o i sistemi di commutazione telefonica basati su comando computerizzato, spesso è utile modellare i meccanismi di dialogo tra le macchine del sistema e la logica di comportamento delle stesse, cioè costruire un simulatore funzionale discreto del sistema onde poter verificare la consistenza del sistema nella sua interezza integrando dinamicamente le logiche delle varie macchine componenti (e quindi realizzando l'interfogliamento di processi paralleli che evolvono nel sistema) ed inoltre ottenere delle misure di performance sul modello. (1) (2) (3)

Essenziale a questo scopo è l'utilizzo del concetto di pseudoparallelismo: le fasi attive dei differenti processi vengono simulate sequenzialmente, ma l'avanzamento del tempo simulato è fatto in modo tale da creare l'illusione del parallelismo (4). Vedremo più avanti come verrà implementato questo concetto.

Vi sono diversi linguaggi di simulazione discreta anche estremamente potenti dal punto di vista concettuale, basti ricordare il Simula 67 nel quale lo pseudoparallelismo è meravigliosamente realizzato (5). Purtroppo, spesso, poichè non si ha disponibile un Simula 67 (o comunque un linguaggio speciale), o perchè si deve lavorare su un mini o comunque per ragioni di trasportabilità, si deve implementare un complesso modello di simulazione discreta usando linguaggi più diffusi, come il Fortran o il Pascal (6).

Orbene, queste note si propongono di mostrare una traccia per disegnare l'architettura di un simulatore discreto interattivo di sistemi paralleli.
È stato utilizzato, come linguaggio di disegno, un "Pascal concettuale" senza allocazione dinamica di variabili.
Per poter fissare le idee è stato scelto un esempio di sistema realtime che fosse il più semplice possibile, ma al tempo stesso significativo.

Un secondo obiettivo che queste note si prefiggono è di tipo didattico. Si vuole, infatti, mettere in luce come viene realizzata l'implementazione di alcuni concetti di base come la sequenza degli eventi, lo pseudoparallelismo, ecc. Infatti, i linguaggi di simulazione discreta, sia per ragioni di efficienza che di sicurezza, tendono a negare all'utente la visibilità della implementazione di questi meccanismi.

2. DESCRIZIONE DEL SISTEMA CONSIDERATO

Viene considerato un sistema i cui componenti sono tasks (entità modellabili come macchine sequenziali a

stati finiti).

Vi sono due tipi di task: task A e task B (vedi fig. 1). Il sistema è composto da una istanza del task A e due istanze del task B (vedi fig. 2). Ogni istanza di task ha una sua coda di ingresso in cui vengono raccolti i "messaggi" che devono essere processati.

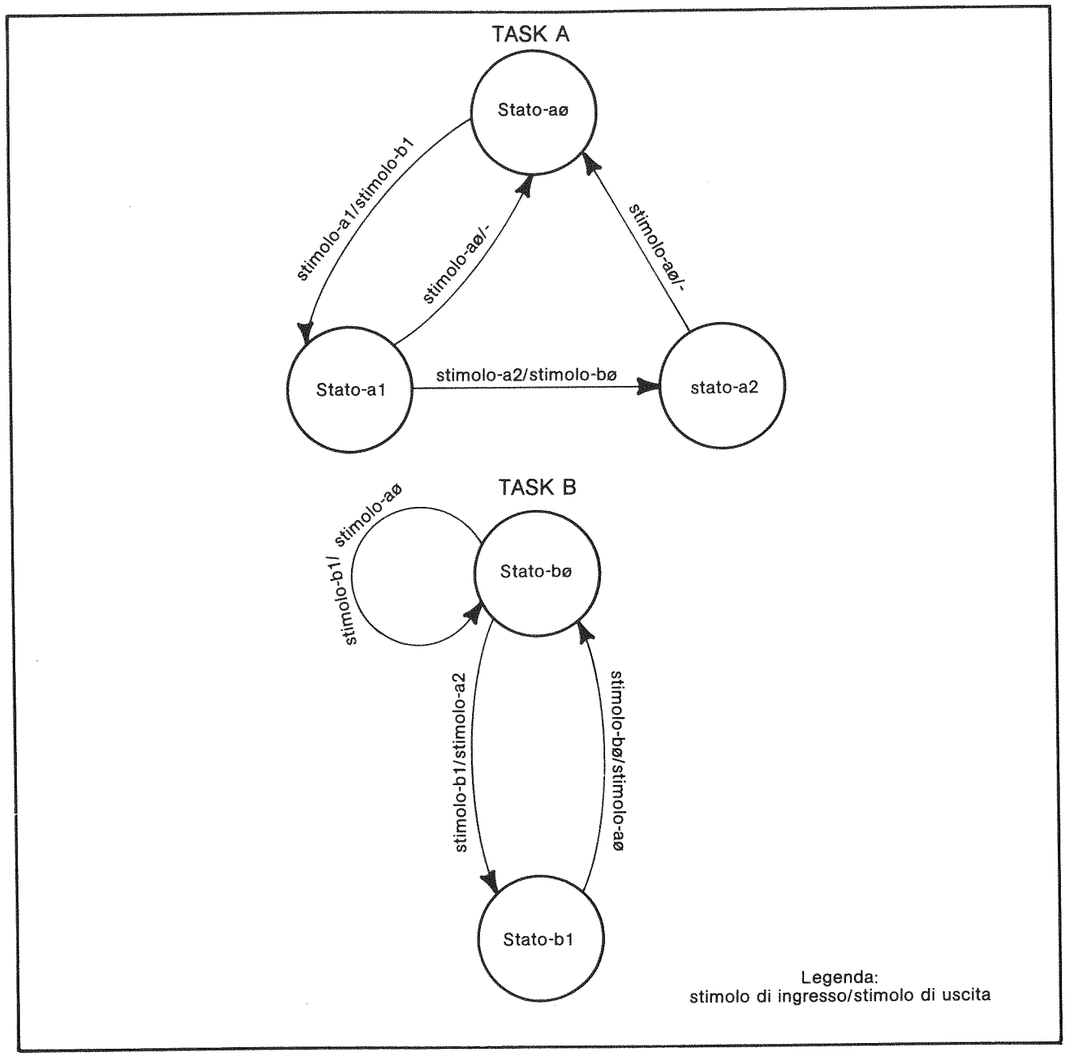
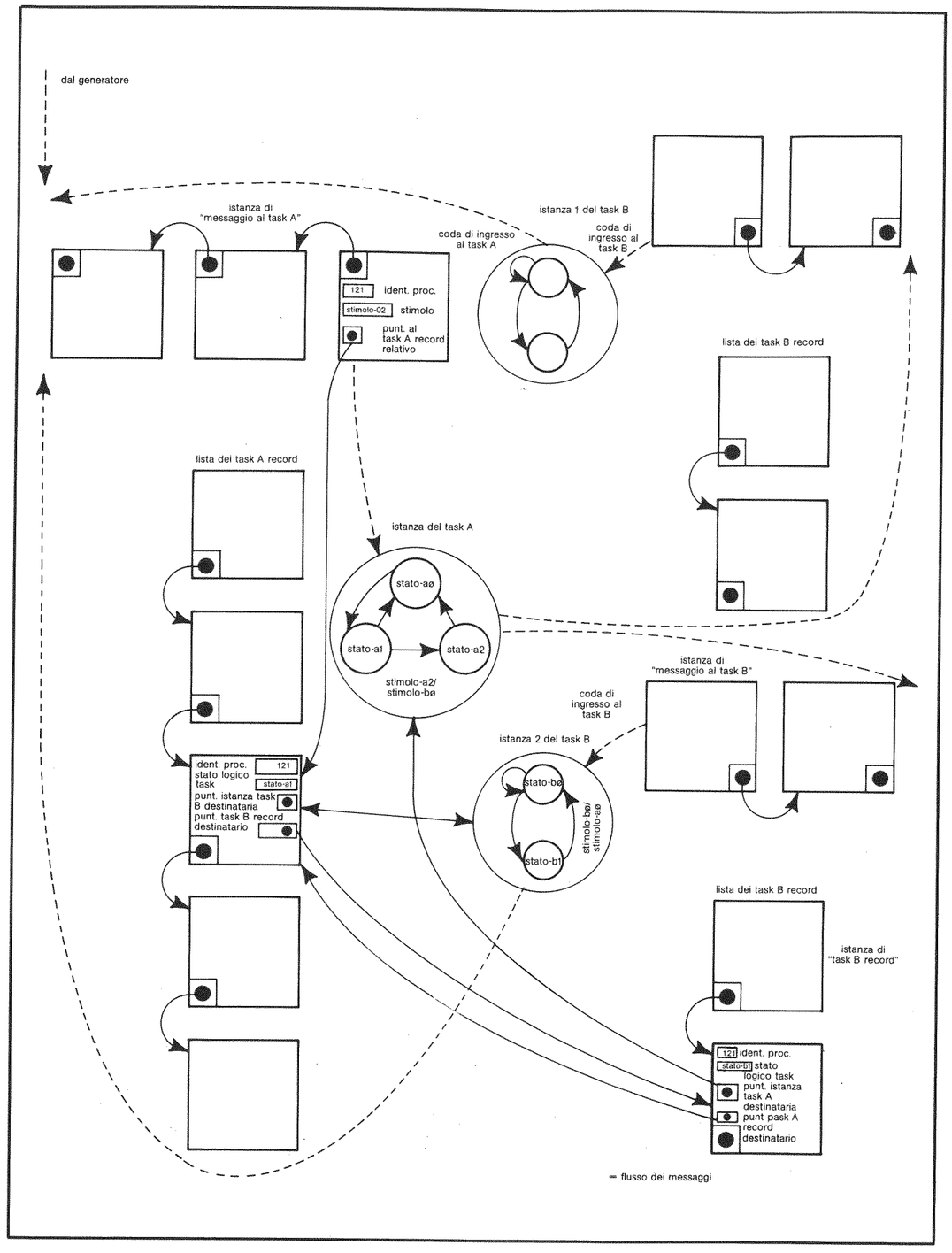


FIG. 1

Le istanze del task B dialogano con l'istanza del task A. Due interlocutori dialogano inviandosi dei messaggi da processare: per esempio se l'istanza del task A vuole comunicare con l'istanza 1 del task B, metterà un opportuno messaggio nella coda di ingresso all'istanza 1 del task B.

Un messaggio è un pacchetto di informazioni contenente, tra l'altro, lo stimolo di ingresso per l'entità destinataria. Quando l'entità destinataria processerà quel messaggio compirà la transizione di stato corrispondente alla coppia: stato corrente dell'entità, stimolo corrente in ingresso; inoltre verrà generalmente inviato un nuovo messaggio di risposta all'interlocutore.



Ogni istanza di task procede nel tempo in maniera asincrona rispetto alle altre.
Il funzionamento di una istanza di task consiste nel processare il primo messaggio in coda, estrarlo dalla coda, di nuovo processare il primo messaggio in coda, e così via fino a che la coda è vuota.
La processazione di un messaggio richiede una certa quantità di tempo. Quando il task trova la coda vuota si passiva.

Il sistema viene alimentato generando nuovi messaggi che vengono inseriti nella "coda di ingresso al task A". Il tempo di interattivo tra una generazione e la successiva segna l'andamento di una distribuzione di probabilità esponenziale negativa.

La generazione di un messaggio rappresenta la nascita di un nuovo processo. Insieme al messaggio viene generata anche una istanza di "task A record", cioè un pacchetto di informazioni riguardanti il processo nato. Ogni istanza di task ha la lista delle proprie istanze di task record.

Ogni istanza di task record contiene il campo "identificatore di processo" il cui valore, definito all'atto della nascita del processo, identifica un processo in vita nel sistema.
Nel pacchetto di informazioni costituenti il messaggio ad un task, troviamo oltre al campo "stimolo", anche il campo "identificatore di processo" e il campo "puntatore al relativo task record". Infatti ogni messaggio presente nella coda di ingresso di una istanza di task punta, nella lista di task record di quel task, a quel task record il cui campo "identificatore di processo" ha lo stesso valore dell'omonimo campo del messaggio puntante.

Nel messaggio, come dicevamo, è contenuto il campo "stimolo", mentre nel relativo task record è contenuto il campo "stato logico del task". Il valore di questo ultimo campo rappresenta lo stato in cui si trova quel task relativamente a quel processo che è appunto identificato dal valore del campo "identificatore di processo" di cui sopra.

Questa coppia di valori stimolo-stato permette quindi al task di selezionare il pacchetto di azioni opportune la cui esecuzione è stata chiamata processazione di messaggio.

Poichè, come già detto, ogni istanza di task procede nel tempo in maniera asincrona rispetto alle altre abbiamo contemporaneamente in vita nel sistema un certo insieme di processi che si interfogliano nel tempo ed evolvono parallelamente ed asincronamente.
Quando l'istanza del task A processa un messaggio relativo ad un processo appena nato, esamina la lunghezza delle rispettive code di ingresso delle due istanze del task B e quindi seleziona come interlocutore l'istanza del task B avente la coda più corta.

Ogni istanza di task B funziona con un certo grado di indeterminismo, infatti quando si trova nello stato - bø se riceve lo stimolo - b1 può ritornare nello stesso stato inviando al task A stimolo - aø oppure andare nello stato - b1 inviando alla task A stimolo - a2. Ognuna di queste alternative è decisa estraendo un numero casuale.

3. DESCRIZIONE DEL MODELLO

Il modello del sistema è composto da 4 entità tipo: il generatore (di nuovi processi), il task A, il task B, il timeout (che chiude la simulazione).

3.1. Entità del modello

Avremo quindi il tipo scalare:

entità del modello = (taskA,taskB, generatore,timeout)

Ogni entità è composta da due parti: una parte (parte dati) che descrive la struttura dati dell'entità e una parte (parte azioni) che descrive come funziona quella entità.

La parte dati è implementata nei tipi:

struttura dati del.....= **record**

end

Avremo quindi i quattro tipi strutturati: struttura dati del task A, struttura dati del task B, struttura dati del generatore, struttura dati del timeout.

La parte azione è implementata nelle procedure:

procedure azioni del .....

Avremo quindi le quattro procedure: azioni del task A, azioni del task B, azione del generatore, azioni del timeout.

Una istanza di entità è rappresentata da una istanza della struttura dati di quell'entità.
La parte azioni di una entità è comune a tutte le istanze della struttura dati di quella entità.

3.2. Il Tempo simulato

L'avanzamento del tempo simulato è ottenuto manipolando con opportune primitive una "lista eventi" ossia una lista di istanze del tipo "evento".

Abbiamo infatti il tipo strutturato:

evento = **record**
.
.
.
end

I campi contenuti in questo tipo sono:

1. il campo:

tempo evento: numero reale non negativo

il cui valore rappresenta l'istanza di tempo per il quale è schedulato l'evento (la lista è ordinata in base a valori crescenti dei campi "tempo evento" delle istanze di "evento"; il valore corrente del tempo simulato è quindi definito dal valore di "tempo evento" della prima istanza di "evento")

2. il campo:

successivo: range dell'indice del vettore di elementi di tipo "evento";

il cui valore è la posizione della istanza di evento successiva nella lista: avremo quindi il tipo:

vettore di elementi di tipo evento = **array** (range del vettore di elementi di tipo "evento") **of** evento

e quindi il tipo:

lista eventi = **record**
primo evento, primo evento libero: range del vettore di elementi di tipo "evento";
lista piena: boolean;
.
.
vettore: vettore di elementi di tipo "evento"
end

3. i campi:

entità: entità del modello
istanza dell'entità: puntatore alla istanza della struttura dati dell'entità

Questi ultimi due campi permettono di associare una istanza di "evento" ad una istanza di entità.

Così, se abbiamo per ogni entità il vettore delle istanze della struttura dati di quella di entità, il valore del campo "istanza dell'entità" sarà la posizione, all'interno del vettore, dell'elemento che rappresenta l'istanza di entità a cui si riferisce l'evento (puntatore all'istanza della struttura dati dell'entità: integer)

3.3. Lo pseudoparallelismo

Lo pseudoparallelismo impiegato nella simulazione discreta richiede che l'esecuzione della parte azioni per una istanza di entità, possa essere sospesa e ripresa poi più avanti nel corso dell'esecuzione del programma di simulazione, a partire dal punto in cui si era interrotta.

A questo scopo definiamo per ognuna delle entità del modello un tipo scalare:

punto di rientro nel ..... = (. . . , . . . , . . .)

Avremo quindi i quattro tipi scalari:

punto di rientro nel task A, punto di rientro nel task B, punto di rientro nel generatore, punto di rientro nel timeout.

Definiamo in ogni "struttura dati del ....." il campo: punto di rientro: punto di rientro nel.....

Per esempio:

punto di rientro nel generatore = (interattivo, arrivo)

struttura dati del generatore = **record**
.
.
.
punto di rientro:
punto di rientro nel generatore
.
.
.
end

La parte azioni di una entità avrà una struttura di tipo **case** sul valore del campo "punto di rientro" dell'istanza considerata.

Per esempio:

punto di rientro nel task B = (trattamento del messagio in testa alla coda di ingresso, esecuzione dell'attività corrispondente a stato-bø stimolo-b1, ...)

procedure azioni del task B
.
.
.
case valore del campo "punto di rientro" della istanza della "struttura dati del task B" puntata dal campo "istanza dell'entità" della prima istanza di "evento"

of
trattamento del messaggio in

testa alla coda di ingresso: **begin**
.
.
.
end
esecuzione dell'attività corrispondente
a stato-bø stimolo-b1: **begin**
.
.
.
end

Ogni segmento del **case** terminerà con un opportuno assegnamento di valori al campo "punto di rientro" della istanza di struttura dati di entità di questione.

Nel corpo di un segmento del **case** possono esserci delle azioni di schedulazione che maneggiano la "lista eventi". Per esempio se un'istanza di entità vuole attivare immediatamente un'altra istanza di entità, allora basterà che aggiunga in testa alla "lista eventi" una nuova istanza di "evento" che punti all'istanza di entità da attivare e che contenga il nome di quella entità.

Così, supponendo di disporre di una lista libera di istanze di "evento", se dovessimo per esempio descrivere le azioni che la istanza del task A deve fare per attivare la istanza 1 del task B, scriveremmo:

1. assegna al campo "entità" del primo evento libero, il valore: "task B"
2. assegna al campo "istanza di entità" del primo evento libero, il valore: 1
3. assegna al campo "tempo evento" del primo evento libero, il valore del campo "tempo evento" del primo evento della "lista eventi"
4. schedula il primo evento libero in testa alla "lista eventi".

Se un'istanza di entità vuole passivarsi, basta che tolga dalla "lista eventi" il primo evento.

Se invece vuole simulare la consumazione di un certo intervallo Δt di tempo deve eseguire:

1. aggiungi Δt al campo "tempo evento" del primo evento
2. inserisci il primo evento nella opportuna posizione nella "lista eventi" secondo l'ordinamento cronologico.

Il nucleo della statement part del programma simulatore è costituito da una iterazione del tipo seguente:

while not simulazione terminata **do**
begin
case valore del campo "entità"
del primo evento
of
task A: azioni del task A;
task B: azioni del task B;
generatore: azioni del generatore;
timeout: azioni del timeout;
end
.
.
.
end

3.4. Strutture dati delle entità task

Una istanza di entità di cui non è in corso l'esecuzione può avere un'istanza di "evento" che riferisce ad essa e che si trova nella "lista eventi" schedulata ad un tempo futuro rispetto a quello attuale. In questo caso, diciamo che quell'istanza di entità è nello stato "consumante tempo", ossia simula la consumazione del tempo necessario per eseguire un certo insieme di operazioni nel mondo reale.

Viceversa, può darsi che per una certa istanza di entità non figuri nella "lista eventi" alcuna istanza di "evento" riferente ad essa.
Diciamo, in tal caso, che quella entità è nello stato "passivo".

Avremo allora il tipo scalare:

stato fisico di un task = (passivo, consumante tempo)

e quindi nei tipi "struttura dati del task A" e "struttura dati del task B" avremo il campo:

stato fisico del task: stato fisico di un task

3.4.1 Coda di messaggi

Come già accennato nel paragrafo 2, ogni istanza di task possiede una propria coda di ingresso i cui elementi sono messaggi (definiti come pacchetti di informazioni).

I seguenti tipi scalari definiscono i due insiemi di stimoli in ingresso rispettivamente per il task A e per il task B:
stimolo al task A = (stimolo-a1, stimolo-a2, stimolo-aø)
stimolo al task B = (stimolo-b1, stimolo-bø)

Ricordando quindi quanto già detto a proposito del messaggio, avremo i seguenti tipi strutturati:

messaggio al task A = **record**
identificatore di processo: intero positivo;
stimolo: stimolo al task A;
puntatore al task A record relativo: range dell'indice del vettore di elementi di tipo "task A record";
successivo: range dell'indice del vettore di elementi di tipo "messaggio al task A";
end

vettore di elementi di tipo "messaggio al task A" =
array (range dell'indice del vettore di elementi di tipo "messaggio" al task A") **of** messaggio al task A

e quindi avremo il tipo:

coda di ingresso al task A = **record**
primo messaggio, ultimo messaggio,
primo messaggio libero: range dell'indice del vettore di elementi di tipo "messaggio al task A";
coda piena, coda vuota: boolean;
.
.
.
vettore: vettore di elementi di tipo "messaggio al task A";
end

Similmente per il task B

3.4.2. Lista di task record

Il significato del task record è già stato esposto nel paragrafo 2.

I seguenti tipi scalari definiscono gli insiemi degli stati assumibili rispettivamente dal task A e dal task B

stato logico del task A = (stato-aø, stato-a1, stato-a2)
stato logico del task B = (stato-bø, stato-b1)

Avremo quindi:

task A recod = **record**
identificatore di processo: intero positivo;
stato logico del task: stato logico del task A;
puntatore all'istanza destinataria della "struttura dati del task B": range dell'indice del vettore di elementi di tipo "struttura dati del task B";
puntatore al task B record destinatario: range dell'indice del vettore di elementi di tipo "task B record";
successivo: range dell'indice del vettore di elementi di tipo "task A record"
end

vettore di elementi di tipo "task A record" = **array** (range dell'indice del vettore di elementi di tipo "task A record") **of** task A record

lista dei task A record = **record**
primo task A record,
primo task A record libero: range dell'indice del vettore di elementi di tipo "task A record";
.
.
vettore: vettore di elementi di tipo "task A record";
end

Similmente e simmetricamente per il task B

3.4.3. Automa

Come è stato detto nel paragrafo 2, un task è modellabile come automa. Ad una coppia di valori stato-stimolo deve corrispondere un opportuno pacchetto di azioni ed il corrispondente valore della quantità di tempo necessaria alla realizzazione di quelle azioni nel mondo reale.

Definiamo perciò i seguenti tipi scalari:

consumazione del tempo di esecuzione di una attività del task A = (consumazione del tempo di esecuzione dell'attività corrispondente a stato-aø stimolo-a1,
consumazione del tempo di esecuzione dell'attività corrispondente a stato-a2 stimolo-aø,
.
.
.
consumazione del tempo di esecuzione dell'attività di recovery)

tempo di esecuzione di una attività del task A =
(tempo di esecuzione dell'attività corrispondente a stato-aø stimolo-a1,
.
.
.
tempo di esecuzione dell'attività di recovery)

Avremo perciò una "matrice automa del task A" così definita:

elemento della matrice automa del task A = **record**
azione: consumazione del tempo di esecuzione di una attività del task A;
tempo: tempo di esecuzione di una attività del task A
end

matrice automa del task A = **array** (stato logico del task A, stimolo al task A) **of** elemento della matrice automa del task A.

Notiamo che gli elementi della matrice che corrispon-

dono a combinazioni stato-stimolo non definite, vengono riempiti con i valori:
"consumazione..... di recovery" e "tempo..... di recovery"

Finalmente possiamo scrivere il tipo:

struttura dati del task A = **record**
automa: matrice automa del task A;
coda di ingresso: coda di ingresso al task A;
lista di task record: lista dei task A record;
stato fisico del task: stato fisico di un task;
punto di rientro nel task: punto di rientro nel task A
end

Similmente per la task B.

3.5. Funzionamento delle entità task

Come si è detto al paragrafo 2, il funzionamento di un task consiste nella iterazione definita dal processare il primo messaggio in coda, eliminare dalla coda il messaggio processato, ritornare a processare il primo della coda e così via fino a che la coda risulta vuota, al che il task si passiva. Va inoltre tenuto presente che la processazione di un messaggio consuma una certa quantità di tempo.

La implementazione della processazione sarà quindi eseguita in due tempi successivi.

Notiamo che, salvo precisazione, quando diciamo: "..... assegnare al campo "stato fisico del task"....., "estrarre il primo messaggio dalla coda di ingresso", ecc....., ci riferiamo sempre alla istanza di struttura dati di task che è puntata dal primo evento della "lista eventi".

La prima parte consiste nel:

1. accedere all'elemento della matrice automa del task utilizzando il valore del campo "stato logico del task" del task record puntato dal campo "puntatore al task record relativo" del primo messaggio della coda ingresso, ed utilizzando il valore del campo "stimolo" del primo messaggio della coda di ingresso;
2. saltare, mediante una struttura **case**, ad eseguire il segmento corrispondente al valore del campo "azione" dell'elemento della matrice automa.

L'esecuzione di tale segmento consisterà nel:

2.1 assegnare al campo "stato fisico del task" il valore: "consumante tempo"

2.2 rischedulare il primo evento al tempo futuro ottenuto sommando il valore del tempo presente con il valore del campo "tempo" dell'elemento della matrice automa

2.3 assegnare al campo "punto di rientro nel task" un valore opportuno onde poter eseguire in un secondo tempo la corrispondente seconda parte (per esempio, consideriamo l'istanza di task A; tra le azioni del segmento individuato da: "consumazione del tempo di esecuzione dell'attività corrispondente a stato-aø stimolo-a1", dovrà figurare l'assegnamento al campo "punto di rientro nel task" del valore: "esecuzione dell'attività corrispondente a stato-aø stimolo-a1", che permetterà di individuare la seconda parte)

2.4 cedere il controllo al programma principale

Seconda parte:

generalmente viene inviato un messaggio di risposta all'interlocutore. Ciò si realizza nel:

1. riempire opportunamente i campi del primo messaggio libero della istanza di task destinataria

2. mettere il messaggio riempito nella coda di ingresso dell'istanza di task destinataria

3. estrarre il primo messaggio dalla coda di ingresso

4. assegnare al campo "stato logico del task" dell'istanza di struttura dati di task, il valore corrispondente al successivo stato dell'automa (per esempio se siamo nel segmento individuato dal punto di rientro: "esecuzione dell'attività corrispondente a stato-a1 stimolo-a2" il valore da assegnare sarà: "stato-a2")

5. analizzare il valore del campo "stato fisico del task" dell'istanza di task destinataria:

5.1 se il valore è "passivo", allora:

5.1.1 l'istanza di task destinataria deve essere attivata

5.1.2 al campo "punto di rientro nel task" della istanza di struttura dati di task, viene assegnato il valore: "trattamento del messaggio in testa alla coda di ingresso"

5.1.3 viene ceduto il controllo al programma principale

5.2 se il valore non è "passivo", allora:

5.2.1 al campo "punto di rientro nel task" della istanza di struttura dati di task, viene assegnato il valore: "trattamento del messaggio in testa alla coda di ingresso"

5.2.2 non viene ceduto il controllo al programma principale e quindi si ritorna in testa alla procedura ad eseguire il **case** sul valore del campo "punto di rientro nel task" dell'istanza di struttura dati di task.

Occorre mettere in evidenza alcuni casi speciali di questa "seconda parte". Essi riguardano:

1. quando un processo è appena nato: questo caso riguarda il comportamento del task A nel segmento individuato dal punto di rientro: "esecuzione dell'attività corrispondente a stato-aø stimolo-a1". Infatti in questo caso l'istanza di task A deve inizializzare un nuovo task B record nella lista dei task B record dell'istanza di task B destinataria.
   Inoltre si è stabilito, nel nostro esempio, che in questa fase l'istanza di task A selezioni quell'istanza di task B che ha la coda di ingresso più corta: il dialogo relativo al processo in corso si svolgerà quindi tra queste due istanze di task.

2. il comportamento di una istanza di task B nel segmento: "esecuzione dell'attività corrispondente a stato-bø stimolo-b1"
   Infatti viene estratta casualmente la possibilità o meno di continuare il processo:

   2.1 se il processo deve continuare, allora il comportamento segue l'iter normale descritto in precedenza
   2.2 se il processo deve terminare, allora deve essere eliminato, dall'istanza di struttura dati di task, il task B record relativo a quel processo

3. nelle azioni del segmento "esecuzione dell'attività corrispondente a stato-b1 stimolo-bø, si deve eliminare dalla lista il task B record relativo al processo in corso; così anche per il task A; nei segmenti: "esecuzione dell'attività corrispondente a stato-a1 stimolo-aø" e: "esecuzione dell'attività corrispondente a stato-a2 stimolo-aø", occorre eliminare dalla lista il task A record relativo al processo che quindi viene terminato.

3.6 Funzionamento dell'entità generatore

I punti di rientro del generatore sono due: "interattivo" e "arrivo"

interattivo: **begin**
rischedula il primo evento al tempo futuro ottenuto sommando il tempo presente al valore estratto dal generatore casuale con distribuzione esponenziale negativa;
assegna al campo "punto di rientro" della istanza di "struttura dati del generatore" il valore: "arrivo";
cedi il controllo
end

arrivo: **begin**
riempi opportunamente i campi della prima istanza libera di "messaggio al task A";
metti tale istanza nella "coda di ingresso al task A";
assegna al campo "identificatore di processo" del primo "task A record" libero il numero di identificazione del processo;
inserisci il primo "task A record" libero nella "lista dei task A record";
esamina il valore del campo: "stato fisico del task" della istanza di "struttura dati della task A".

1. se è "passivo"

   · attiva la istanza di task A
   · assegna al campo "punto di rientro", il valore "interarrivo"
   · cedi il controllo

2. se non è "passivo"
   · assegna al campo "punto di rientro", il valore "interarrivo"
   · non cedere il controllo (e quindi ritorna in testa alla procedura per eseguire il **case** sul valore del campo "punto di rientro")
   end

3.7 Funzionamento del timeout

Punto di rientro nel timeout = (inizio simulazione, fine simulazione)
procedure azioni del timeout
begin
case valore del campo "punto di rientro" della istanza della "struttura dati del timeout" puntata dal primo evento

of
inizio simulazione: **begin**
rischedula il primo evento al tempo futuro di fine simulazione;
assegna al campo "punto di rientro" il valore: "fine simulazione"
end
fine simulazione: **begin**
simulazione terminata: = **true**
end
end

3.8 Statement part: struttura del programma principale

La statement part è strutturata nel seguente modo:

begin
· inizializzazioni
· **while not** simulazione terminata **do**
begin
case valore del campo "entità" del primo evento
.
.
of

```
    task A: azioni del task A;
    task B: azioni del task B;
    generatore: azioni del generatore;
    timeout: azioni del timeout
    end
gestione interattività utente-simulatore nel corso della
simulazione
end
· gestione interattività utente-simulatore al termine
  della simulazione
end.
```

3.8.1 Inizializzazioni

Viene richiesto all'utente di fornire i valori del tempo totale di simulazione, e dei parametri delle distribuzioni di probabilità.

I campi "punto di rientro" del generatore e del timeout vengono inizializzati rispettivamente a "interarrivo", e "inizio simulazione".

L'inizializzazione delle istanze delle strutture dati di task, consiste nei riempire opportunamente le matrici automa, nel creare le liste libere dei task record e le liste libere dei messaggi. Per ogni istanza di task, il campo "stato fisico del task" viene inizializzato al valore "passivo"; il campo "punto di rientro nel task" al valore "trattamento del messaggio in testa alla coda di ingresso".

I campi "stato logico del task" dei task A record vengono inizializzati al valore: "stato-aø", mentre quelli dei task B record, al valore: "stato-bø".

Infine, viene inizializzato il meccanismo di simulazione, ossia viene creata la lista libera delle istanze del tipo "evento" e quindi vengono eseguite le seguenti operazioni:
- assegna ai campi "entità" e "tempo evento" del primo evento libero, i valori "timeout", e "ø" rispettivamente;
- schedula il primo evento libero in testa alla "lista eventi";
- assegna ai campi "entità" e "tempo evento" del primo evento libero, i valori "generatore" e "ø" rispettivamente;
- schedula il primo evento libero in testa alla "lista eventi";
- assegna alla variabile booleana "simulazione terminata" il valore **"false".**

Notiamo, inoltre, che anche se abbiamo una sola istanza generatore ed una sola istanza timeout, possiamo, per omogeneità, considerare il vettore delle istanze del tipo "generatore" ed il vettore delle istanze del tipo "timeout". Naturalmente, in questo caso, ciascuno di questi vettori è costituito da un solo elemento e quindi dovremo far si che il campo "istanza dell'entità" dei primi due eventi punti all'unico elemento del rispettivo vettore.

3.8.2 Gestione interattività utente-simulatore nel corso della simulazione

L'utente deve poter controllare passo passo lo stato del sistema nel corso della sua evoluzione.
Possiamo pensare che l'utente del simulatore utilizzi una modalità interattiva basata sul concetto di menù dinamici e gerarchici.
Per esempio, ad un primo livello (livello del menù principale) potremmo considerare comandi del tipo:

comandi menù livello ø = (processa il prossimo evento, continua automaticamente la simulazione, riparti dall'inizio con una nuova simulazione, termina l'uso del simulatore, help, visualizza lo stato del sistema).

Quest'ultimo comando genera sul video il menù di secondo livello o menù delle visualizzazioni, nel quale si chiede di specificare che cosa deve essere visualizzato: lo stato di un processo, o di una istanza di task, ecc. Volendo, si potrebbe ulteriormente dettagliare la richiesta, per esempio se si tratta di una istanza di task, richiedere se deve essere visualizzata la coda di ingresso, oppure la lista dei task record, e così via.

Notiamo che il comando "continua automaticamente la simulazione", inibisce l'interattività fino al termine della simulazione. Avremo cioè:

```
while not simulazione terminata do
begin
  case.....
  end.....
  if interattività abilitata then
  begin
  visualizza menù principale
  lettura comando
  case comando menù principale of
  processa il prossimo evento: go to end while
  continua automaticamente la simulazione:
       begin interattività abilitata: = false
       go to end while
end                    .
                       .
                       .
    end
  end
end while:
end
```

3.8.2 Gestione interattività utente-simulatore al termine della simulazione

Si tratta di gestire il menù ottenuto da quello precedente eliminando, ovviamente, i comandi "processa il prossimo evento" e "continua automaticamente la simulazione" ed aggiungendo il comando "stampa statistiche".

4. MISURE SUL MODELLO

Come è già stato detto, il modello di simulazione può servire oltre che a verificare la bontà della logica del sistema, anche ad ottenere alcuni dati sulle prestazioni del sistema. Tali dati sono statistiche calcolate effettuando opportune misurazioni sul modello nel corso della sua simulazione.

Per esempio, classici sono i seguenti indici di prestazione:
- utilizzazione (di una risorsa)
- throughput
- lunghezza media di una coda
- tempo medio di attesa in coda.

Possiamo misurare queste grandezze per ogni istanza di task, per fare un esempio riferiamoci all'istanza 1 del task B.

Allora:

1) il grado di utilizzazione della istanza 1 del task B al termine della simulazione sarà dato da:
tempo totale di occupazione della istanza 1 del task B nel corso della simulazione/tempo totale di simulazione

   Basterenne quindi tener conto di tutti i "tempi morti" in cui questa istanza di task è passiva, ossia aggiornare un apposito campo accumulatore definito nella "struttura dati del task B", ogniqualvolta l'istanza viene attivata, aggiungendo la differenza tra il valore del tempo presente e il valore del tempo al quale si era passivata.

2) Il throughput della istanza 1 del task B sarà dato da:

   numero totale di messaggi processati dall'istanza 1 del task B/tempo totale di simulazione.

   È quindi necessario che al termine di ogni processazione di messaggio da parte dell'istanza venga incrementato un apposito campo contatore definito nella "struttura dati del task B".

3) La lunghezza media della coda di ingresso all'istanza 1 del task B sarà data da:

   valore dell'area dell'istogramma relativo alla coda di ingresso alla istanza 1 del task B/tempo totale di simulazione.

   Dobbiamo, infatti, definire nella "struttura dati del task B" un nuovo campo "area istogramma", che viene aggiornato nel seguente modo ogni volta che varia la lunghezza della coda:

   area istogramma: = area istogramma + (valore del tempo presente - valore del tempo al momento dell'ultima variazione della lunghezza della coda) x nuovo valore della lunghezza della coda.

   Dovremo quindi riservare un apposito campo della "struttura dati del task B" per registrare il valore del tempo al momento dell'ultima variazione della lunghezza della coda.

4) Il tempo medio di attesa nella coda di ingresso all'istanza 1 del task B sarà dato da:

   valore dell'are dell'istogramma relativo alla coda di ingresso alla istanza 1 del task B/numero totale di messaggi processati dall'istanza 1 del task B.

   Notiamo che tali valori di prestazione saranno più attendibili (cioè statisticamente significativi) quanto più grande sarà il campione di misure effettuate e quindi quanto più grande, fissato per il generatore un certo tempo medio di interattivo, sarà il valore di "tempo totale di simulazione", cioè quanto più lungo sarà il run di simulazione.

4. RIFERIMENTI

(1) S. Mussi, SIMULAZIONE SOFTWARE DI AMBIENTE TELEFONICO, Tesi di laurea, Università degli Studi di Milano, Anno accademico 1974-75, Corso di laurea in Fisica

(2) F. Andreis, D. Baldoni, E. Husu, F. Mazza, S. Mussi, SISTEMA PROTEO: IL SIMULATORE AMBIENTALE PROSIT COME STRUMENTO PER LA PROVA DI PROGRAMMI E PER LA VALUTAZIONE DELL'AFFIDABILITÀ DEL SOFTWARE, Atti congresso AICA, 1977

(3) S. Schoemaker, COMPUTER NETWORK AND SIMULATION, North-Holland Publishing Company

(4) G. Fishman, PRINCIPLES OF DISCRETE EVENT SIMULATION, John Wiley and Sons, New York (1977)

(5) Dahl, Myhrhaug, Nygaard, THE SIMULA 67 COMMON BASE LANGUAGE, Norwegian Computing Center, Oslo

(6) N. Wirth, K. Jensen, PASCAL USER MANUAL AND REPORT, Springer-Verlag (74)

COMPILERS PORTABILITY THROUGH INTERMEDIATE DATA STRUCTURES

G. Castelli(*), M. Fiorentini (*) (+)
(*) Istituto di Cibernetica, University of Milano (+) Etnoteam S.p.A.

1. Abstract

This paper describes the techniques used to implement a family of different Pascal compilers on three different machines. The main goal, in order to minimize the maintenance effort in an industrial environnment, was to have one front end with three different code generators. Every compiler is conceptually divided into three phases and can be structured as a single, double or triple pass, so that any desired compromise between memory occupation and compilation speed can be achieved. Furthermore this structure allows an easy machine independent code optimization during the graph generation or during its subsequent analysis.
These phases communicate among each other by means of a data structure, organized as a graph, acting as an intermediate language.

2. INTRODUCTION

This presentation describes how an appropriate data structuring may affect a number of characteristics, often critical, in the implementation phase and subsequent extension and maintenance phases of a compiler.

While graphs are quite widely used in implementing compilers on machines with small memory partitions, see for example the Standard C compiler, the Portable C compiler and the Fortran 77 compiler under Unix, we want to give to the reader a greater level of detail about the definitions and the use of such an approach

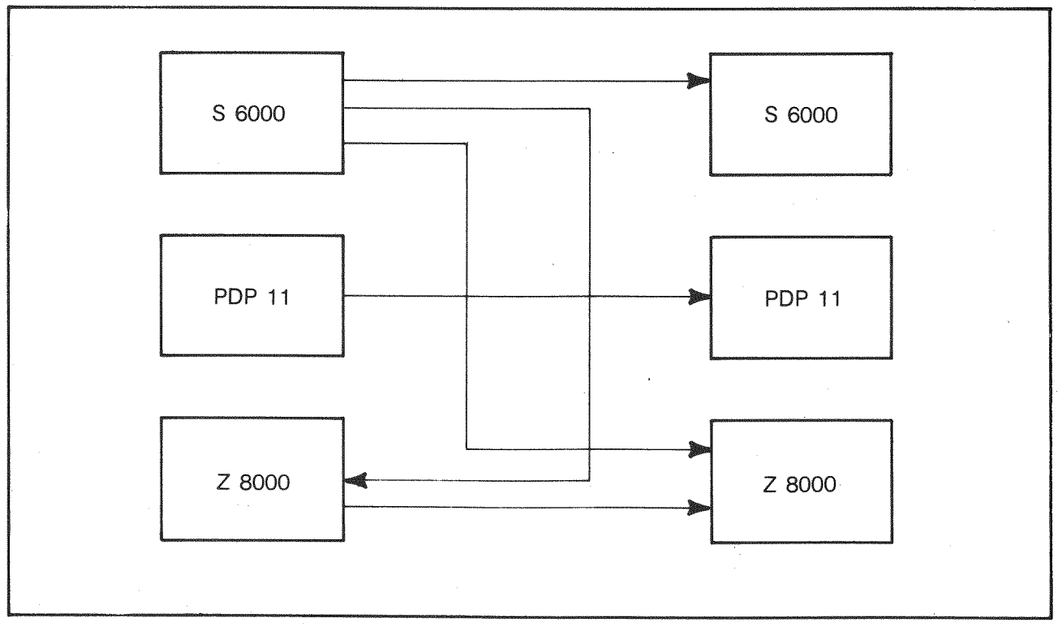


FIG. 1 - Existing Pascal Compiles and Cross-Compilers

to the problem. We will concentrate on two aspects, which are very important in modern compilers: portability (or more appropriately retargetability) and quality of generated code. The illustrated structures increase the first by clearly separating machine independent features from machine dependent ones; the second is increased by identifying a number of possible optimization during the source analysis phase, without using complex code optimizers.

A total of six Pascal compilers have been implemented using this approach (see figure 1), in the same development environment, with a very tight schedule. In order to reduce both implementation costs and maintenance efforts, it was necessary to design an architecture with a unique front-end and different code generators depending only on the target machines. This allowed a very small group of people to manage all these tasks in parallel, being most of changes and improvement sharable among all the compilers.

The three machines presently used have very different architectures: each CPU is a 16 bit processor, but the S6000 is a pure stack machine with one data segment of at most 64K and up to 256 code segment of at most 64K each; the PDP 11 is an 8 register machine with some stack handling capabilities; the size of the data and the program segments can not exceed 64K each. The Z8000 has 16 register, some stack handling capabilities and up to 128 data or program segments of at most 64K each.

The compiler structure must then satisfy different constraints and it must be easy and inexpensive to rehost the same compiler.

The compiler is divided into three parts, the classical ones: declaration analysis, body analysis and code generation, and it is possible to assemble them as a single or multi pass compiler depending on available memory and compilation speed requirements.

3. DATA STRUCTURES

Usually a Pascal compiler describes the language objects through two kinds of descriptors: the first one called "n-node" contains some information related to the identifiers encountered during the declaration analysis, such as the declaration level, the kind of the object, its name and so on. The second one, called "t-node", contains the identifiers attribute in term of structure, descriptions of the elements, range value and other related informations (see figure 2).

```
t = packed record
  bytes: integer;
  ftype, ptype: boolean;
  case form: typeforms of
    scalar: (case scalkind: declkind of
          standard: ();
          declared: (maxconst: pn));
    subrange: (rangeof: pt; min,max: integer);
    pointer: (pointerto: pt);
    sets: (setof: pt; minsetel, maxsetel: integer);
    carrays,
    arrays: (arrayof, indexedby: pt; pckdarr: boolean);
    records: (pckdrec: boolean; fields: pn;
          fstfield: pn; varpart: pt);
    tagfield: (tagname: pn; variants: pt);
    variant: (varfldlst: pn; nextvar, subvar: pt; varva-
          lus: pintlist);
    files: (pckdfile: boolean; fileof: pt);
    sconts: (stringlen: 1..maxstringlen);
  end;

n = packed record
  case node: nodekind of
    identnode:
      (name: alfa; llink,rlink,next: pn; idtype: pt;
      case class: nameclass of
        types: ();
        mods: (listofimp: pn);
        consts: (valueof: valu);
        vars: (vkind: accesskind;
          vlev: levrange; voff: integer);
        field: (foff: integer);
        proc,
        func: (case pfdeclkind: declkind of
          standard: (key: 1..maxstanprocs);
          declared: (paramlist: pn;
            locconstptr: constptr;
            retparams, foffset: integer;
            pflev: levrange;
            pfdecl: (decl,extdecl,
              forwdecl,formal,imported);
            lc: integer))
end;
```

figure 2 - T. node and n-node

In our implementation the "n-nodes" have been modified in order to describe not only identifiers attributes, but also a generalized form of encountered operators and their relations with the operands. This has been obtained by making the old "n-node" one of the variants of a record extended with four more variants: one to describe unary operators and their operand, one to describe binary operators and their operands, one to describe constants and one to describe subrange checking (this is just a special case of the second, but has been introduced for efficiency reasons) (see fig. 3).

```
n = packed record
  case node: nodekind of
    identnode:
      (name: alfa; llink,rlink,next: pn; idtype: pt;
      case class: nameclass of
        types: ();
        mods: (listofimp: pn);
        consts: (valueof: valu);
        vars: (vkind: accesskind;
          vlev: levrange; voff: integer);
        field: (foff: integer);
        proc,
        func: (case pfdeclkind: declkind of
          standard: (key: 1..maxstanprocs);
          declared: (paramlist: pn;
            locconstptr: constptr;
            retparams, foffset: integer;
            pflev: levrange;
            pfdecl: (decl,extdecl,fprwdecl
              forma,imported);
            lc: integer));
    unnode: (unop: 0..maxunop; unarg: pn);
    binnode: (binop: 0..maxbinop; leftarg, rightarg:
      pn);
    cstnode: (csttype: pt; cstvalu: valu; nextref: pn);
    rangenode: (rpn: pn; rangehi,rangelo: integer);
end;
```

figure 3 - Extended n-node

The statement, that usually are not described in any data structure of a compiler, but are implicitely described by the structure of the routines analyzing them, are here described by an appropriate data structure: a record whose variants describe all the possible statements (see fig. 4).

Of course these records are linked together to represent the program being compiled.

The first compiler phase, the declarations analysis, is not very different from the declarations analysis part of most Pascal compilers; it checks declarations syntax and build a complete structure of "n-nodes" (for identifiers) and of "t-nodes" (for the structure of identifiers). The graph built in this phase is just the symbol table.

All the machine dependencies related to this phase, such as memory allocation strategies, packing algorithms, memory occupation of predefined types and address computation depend on a set of parameters that are presently compiler constants that can be easily changed, but in the future will be actualized at installation time by reading them from a file characterizing the targed machine.

The body analysis phase operates in a way similar to the declaration analysis phase. It checks for syntax errors and generates a statement node for each source statement and an appropriate "n-node" for each language constructs; This means that not only the classical language operators, such as the arithmetic ones, are described but a richer set is implemented (see fig. 5), so that it is completely described how to access an array, a record and so on. Figure 6 describes how a procedure call is represented using this notation.


```
stmt = packed record netxstmt: pstmt; stmtck: boo-
  lean; stlinenum: integer;
  case stmtop: stmttypes of
    assignst: (assvar, assexpr: pn),
    fortost,
    fordownst: (forvar: pn; forinit, forlimit: pn; forst:
    pstmt);
    ifst: (ifexpr: pn; thenst,elsest: pstmt);
    withst: (withvar: pn; withbody: pstmt);
    repst,
    whilest: (condexpr: pn; loopstmt: pstmt);
    callst: (procpn: pn; parglist: pn);
    gotost: (gotolab: plabrec; lablev: levrange);
    casest: (caseexpr: pn; cstmtlist: pstmt; otherstmt:
      pstmt);
    cstmtst: (casevals: pintlist; thiscase: pstmt; thispc:
      integer);
    labedst: (stlab: plabrec; labstmt: pstmt);
    exitst,
    returnst: ();
end;
```

figure 4 - Statement node

The third phase, the code generation one, uses the described structure as input and generates machine and system dependent code. A number of checks and code optimizations are possible in a way totally independent from the target machine: for example checks on goto's into structured statements are easily implemented scanning the statement list linked to the structured statement being compiled during the body analysis phase. In the same way any illegal reference to a control variable of a for statement is detected scanning the statement list immediately after its construction. The scanning is implemented through a recursive procedure that examines each statement checking assignments, read operations and var parameter passing. Furthermore many optimizations are implemented in one of the following ways:

```
(unary operators)

notop = 1;          (not)
inegop = 2;         (integer negation)
rnegop = 3;         (real negation)
nilop = 4;          (nil)
ssetop = 5;         (singleton set)
pointerop = 6;      (pointer)
nullsetop = 8;      (null set)
floatop = 9;        (float)
filewinop = 10;     (file window)
descop = 11;        (descriptor)

(binary operators)

indexop = 1;        (index)
fieldop = 2;        (field)
fcallop = 3;        (fcall)
argsop = 4;         (args)
colonop = 5;        (colon)
iaddop = 6;         (integer add)
raddop = 7;         (real add)
rsubop = 8;         (real sub)
rsubop2 = 9;        (real sub)
imultop = 10;       (integer mult)
rmultop = 11;       (real mult)
idivop = 12;        (integer div)
rdivop = 13;        (real div)
modop = 14;         (mod)
andop = 15;         (and)
orop = 16;          (or)
inop = 17;          (in)
unionop = 18;       (union)
diffop = 19;        (diff)
interop = 20;       (inter)
setrop = 21;        (set range)
iltop = 32;         (integer lt)
rltop = 40;         (real lt)
pacltop = 56;       (paclt)
unknownop = 64;
peqop1 = 76;        (pointer eq)
peqop2 = 84;        (pointer neq)
```

figure 5 - Used operators

– Machine independent optimizations

  Examining the data structure being generated; for example when the body analysis phase is generating an integer to floating conversion it checks if the object being converted is a constant and, in this case, it generates just a new constant node describing the floating representation of that value. Similarly when the compiler is generating a node describing a binary operator and detects that both operands are constants, the node itself is substituted by a constant node which describes the result of the operations. Run-time checks too get benefit from this organization: no code is generated to check array indexes or subrange assignments if the expression to be checked can be reduced to a constant.

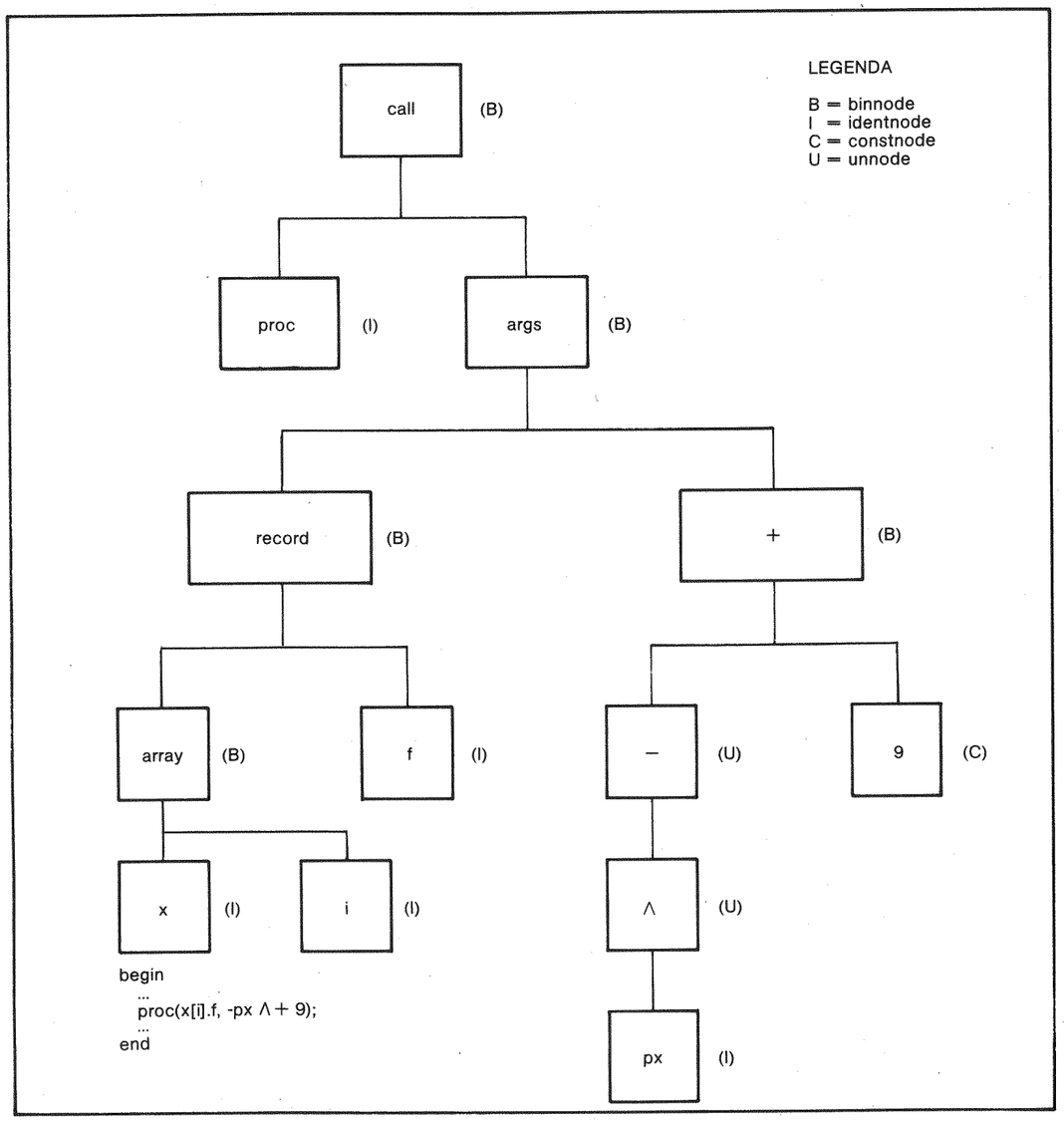


FIG. 6 Internal structure of a procedure call with parameters

– Machine dependent optimizations

  Examining the data structure while generating code; this applies to compile time generation of complete addresses for indexed variables with constant indexes, record fields and so on.

4. COMPILERS OVERVIEW

Presently there are two host machines very different in their architectures: a PDP 11/70 under Unix and an Olivetti S6000 under EMOS. As it has already been seen, the PDP 11/70 has just two segments of 64K each, one for the code and one for the data, and it is not possible to give an overlay structure to program. The S6000 may have up to 256 code segments as long as 64K each, and just one for the data 64K long. Because of the size of each phase, the PDP version of the compiler is structured as a multi step job, where each step is an independent program executed sequentially and where the communication is obtained through intermediate files. These files contain the mentioned data structures, generated by the preceeding phase, and other informations related to communication variable.

While analyzing declarations, each new procedure/function encountered adds some new records to the files. Similar files are generated by the body phase with all the structures describing the body of each procedure/function. In the S6000 version of the compiler the different phases resides simply on different segments. Therefore the body analysis is called just at the end of the declaration analysis of each procedure. This allows to eliminate all intermediate files, except those between the initializing program and the compiler itself.

The architectural modification has been obtained by simply replacing pseudo procedure calls in the PDP version, with the actual calls in the S6000 version: the subsequent phase is actually called instead of calling the intermediate file generation routines.

5. ACKNOWLEDGMENTS

The first compiler, i.e. the PDP 11/70 for PDP 11/70 itself version, was developed by the Silicon Valley Software Inc., Cupertino CA. that first designed the described architecture. A number of characteristics, later refined by us, were present in the original version, while no optimization was implemented nor any easy way to rehost and retarget the compiler was designed.

STRING MANIPULATION: A LIBRARY PACKAGE

Lucio D'Amico
Istituto di Cibernetica, University of Milano

Abstract

This document describe a package of C language library functions which allow:

string replication
positioning strings
character positions and substrings
string analysis
string scanning
regular expression handling

This package uses the standard C functions strcat, strncat, strcmp, strncmp, strcpy, strncpy, strlen. The package is implemented under UNIX (*) V7 operating system.

Riassunto

Questo documento descrive un package di funzioni di libreria per il linguaggio di programmazione C che permette:

replica di stringhe
posizionamenti di stringhe
sottostringhe e posizioni di caratteri
analisi di stringhe
scansione di stringhe
gestione di espressioni regolari

Questo package usa le funzioni C strcat, strncat, strcmp, strncmp, strcpy, strncpy, strlen. Il package è implementato sotto la versione 7 del sistema operativo UNIX (*).

1. OVERVIEW

The string package uses a terminology similar to that of ICON language, a string oriented language developed at the University of Arizona by C. Coutan, R. Griswold and S. Wampler.

This document is organized into chapters describing the major features of the string package. Every function is described separately or is grouped with others of similar nature. After the description, the examples of usage are given. The examples illustrate possibilities that may not be obvious.

It's not necessary to learn the examples of usage; don't read the paragraph unless you need to.

2. COMPILING

In order to use the library, it is necessary to have compilations of the form:

cc [flags] file .... - lstrings

3. TERMINOLOGY

3.1. Characters

Characters as well as strings are data objects in C.

The characters set used by C is based on ASCII. There are 256 different characters available for use in C programs. The octal code 40 character is the blank.

It is customary to think of characters in terms of their graphic representations and control functions, characters are basically integers.

Internally the integers corresponding to ASCII are represented by octal codes from 000 to 177. The order of characters in the sequence is the basis for comparing strings and for sorting.

3.2. Strings

The type of a string is "array of char".

(*) UNIX è un marchio di fabbrica dei Bell Laboratories

A string constant is a sequence of zero or more characters surrounded by double quotes. When a string constant is written in a C program, the compiler creates an array of characters containing the characters of the string; and terminates it with a "\0" (null character).

This representation means that there is no real limit to how long a string can be, but programs have to scan one completely to determine its length.

The physical storage required is one more location than the number of characters written between quotes.

3.3. Character Pointers

Perhaps the most common occurrence of string constants is as arguments to functions. What the function receives is a pointer to the character array. A pointer contains the address of an object and it is possible to access the object "indirectly" through the pointer.

Suppose that x is a variable and px is a pointer. The unary operator & gives the address of an object, so the statement px = &x assigns the address of x to the variable px and px point to x.

The operator * treats its operand as can address and accesses that address to fetch the contents.

If y and x are variables of the same type then

y = *px

assigns to y the contents of whatever px point to.

Any operation which can be achieved by array subscripting can also be done with pointers.

The declaration char a[20], defines an array a of size 20. If pa is a pointer to a char, declared as char *pa, then the assignment pa = &a[0] sets pa to point to the zeroth element of a; that is pa contains the address of a[0] in c. If pa points to a[0], *(pa+1) refers to the contents of a[1], *(pa+i) refers to the contents of a[i].

Note that in pa+i, i is multiplied by the size of the objects that pa points to before being added to pa.

Well, given a pointer pa to a character we are able to identify uniquely a string. The first character of the string is *(pa) and the string is terminated by the null ascii character.

So in C an alternate definition of string is "pointer to char".

3.4. Functions

In C language there is not a real distinction between functions and procedures. A function that contains a "return(expression)" statement is analogous to a Pascal like function, a function that contains return without expression or does not contains return is analogous to a pascal like procedure.

Functions can returns a pointer. If a C function that could be like a Pascal procedure has among its parameters one with the aim to receiving the results of the function, it is profitable that the function returns the receiving string.
In this way it is possible to pass the function like a parameter to another function. For instance the UNIX version 7 function strcat (see par. 4.4.1) is defined by

char *strcat(s1, s2)

and contains a return(s1).

It is satisfactory to have the call

strcat(s1, strcat(s2, s3));

Otherwise if strcat is defined by

strcat(s1, s2)

and don't contains a return(s1) it is requires to solve the same problem in this way:

strcat(s2, s3)
strcat(s1, s2)

3.5. Character sets (cset)

Whereas a string is an ordered sequence of characters in which the same character may appear more than once, in this document we can see a character set as an ordered collection of characters in which the same character may appear only once.

For example the character set corresponding to the string "reverse" is "revs".

4. THE FUNCTIONS

The functions operate on null-terminated strings.

4.1. Conversion functions.

```
char *itoa(n, s)
char *s;
```

itoa converts the integer number n to character string s. The function returns s that is the receiving string.

```
char *ltoa(n, s)
long n;
char *s;
```

ltoa converts a long integer number n to character string s. The function returns s that is the receiving string.

```
char *atocset(s, c)
char *s, *c;
```

atocset returns the receiving character set c corresponding to the conversion of the string s.

Unix supplies the following functions (see ATOF(3)):

atoi(s) that converts string to integer
atol(s) that converts string to long integer
atof(s) that converts string to double precision

4.2. Character set functions

char *ascii()

ascii returns a character set consisting of characters from 040 to 128.

char *ucase()

ucase returns a character set consisting of upper case letters.

char *lcase()

lcase returns a character set consisting of lower case letters.

char *digit()

digit returns a character set consisting of 0-9 digits.

char *special()

special returns a character set consisting of ascii special chars.

```
char *unioncset(c1, c2, c3)
char *c1, *c2, *c3;
```

unioncset returns the receiving character set c3 that is the union of c1 and c2.

Error conditions:
    c1 is not a character set
    c2 is not a character set

```
char *intercset(c1, c2, c3)
char *c1, *c2, *c3;
```

intercset returns the receiving character set c3 that is the intersection of c1 and c2.

Error conditions:

    c1 is not a character set
    c2 is not a character set

```
char *diffcset(c1, c2, c3)
char *c1, *c2, *c3;
```

diffcset returns the receiving character set c3 that is the difference of c1 and c2; that is, all the characters in c1 that are not in c2.

Error conditions:
    c1 is not a character set
    c2 is not a character set

```
char *notcset(c1, c2)
char *c1, *c2;
```

notcset returns the receiving character set c2 that is the complement of c1 respect to ascii().

Error condition:
    c1 is not a character set

```
iscset(s)
char *s;
```

iscset returns 1 if the string c is a character set, returns 0 if c is not a character set.

```
incset(ch, c)
char ch, *c;
```

incset returns 1 if char ch is in character set c else returns 0

Error conditions:
    c is not a character set

4.3 Character positions and substrings.

The positions of characters in a string are numbered from the left starting at 1. The numbering identifies positions between characters.
For example, the positions in the string HELP are

$^{1}H^{2}E^{3}L^{4}P^{5}$

Positions may also be specified with respect to the right end of a string, using nonpositive numbers starting at 0 and continuing with negative values toward the left:

$^{-4}H^{-3}E^{-2}L^{-1}P^{0}$

For this string, positions 5 and 0 are equivalent, positions 4 and -1 are equivalent, and so on.
Positions for a string s must be specified in the range - strlen(s) to strlen(s) + 1 inclusive, where strlen is a Unix supplied function (see STRING(3)).
The only allowable position for the empty string are 0 and 1.
Note that the package functions never return negative or null positions in strings.
A substring is a sequence of characters within a string. An initial substring of s is one that begins at the first character of s. A terminal substring of s is one that ends at the last character of s.
Substrings are determined by beginning and ending positions, using a range specification.
Sometimes we use notation s[i:j] to indicate the substring of s between positions i and j.

The package supplies the following functions:

```
char *subsbet(s1, i, j, s2)
char *s1, *s2;
```

subsbet returns the receiving string s2 that consists of the characters between positions i and j of s1.
Note that i and j may be given by positive or nonpositive specifications and subsbet(s1,i,j,s2) is equivalent to subsbet(s1,j,i,s2).

Error conditions:
    position i is out of allowed range
    position j is out of allowed range

```
char *subsfol(s1, i, k, s2)
char *s1, *s2;
```

subsfol returns the receiving string s2 that is the substring of s1 consisting of k characters following position i.

Error conditions:
    position i is out of allowed range
    k drives out of allowed range

```
char *subsprec(s1, i, k, s2)
char *s1, *s2;
```

subsprec returns the receiving string s2 that is the substring of s1 consisting of k characters preceding position i.

Error conditions:
    position i is out of allowed range
    k drives out of allowed range

Unix supplies (see STRINGS(3)) the following functions:

```
char *strcpy(s1, s2)
char *s1, *s2;
```

strcpy copies string s2 to receiving string s2, stopping after the null character has been moved and returns s1.

```
char *strncpy(s1, s2, n)
char *s1, *s2;
```

strncpy copies exactly n characters, truncating or null padding s2 and returns the receiving string s1.

4.4 Constructing strings

4.4.1 Concatenation

Since a string is a sequence of characters, one of the most natural string construction operations is concatenation: appending one string to another.

Unix supplies (see STRING(3)) two operations:

```
char *strcat(s1, s2)
char *s1, *s2;
```

strcat appends a copy of string s2 to the end of string s1 and returns the receiving string s1.

```
char *strncat(s1, s2, n)
char *s1, *s2;
```

strncat appends at most n characters of s2 to the end of string s1 and returns the receiving string s1.

The package supplies some more generalized functions called insert, substitute and on the other side delete:

```
char *insert(s1, s2, i)
char *s1, *s2;
```

insert returns s2 modified with the insertion of s1 beginning at i position of s2.

Error condition:
    position i is out of allowed range

```
char *delete(s, i, k)
char *s;
```

delete returns s modified by a deleting k characters starting at i position.

Error conditions:
    string s null
    position i is out of allowed range
    k drives out of allowed range
    k equal to 0

```
char *substitute(s1, s2, i, j)
char *s1, *s2;
```

substitute returns s2 after the substitution of characters between positions i and j in s2 with the string s1.

Error conditions:
    string s2 null
    position i is out of allowed range
    position j is out of allowed range

4.4.2 String replication.

```
char *repl(s, i)
char *s;
```

repl returns s that is the concatenation of i copies of s. The permitted range for i is between 0 and $2^{15}-1$. The value of repl(s, 0) is the null string.

4.5 Positioning strings

Positioning data in strings of a specified size is frequently used, especially when printing output in columns. This package supplies three functions for doing this.

char *left(s1, i, s2, s3)
char *s1, *s2, *s3;

left returns the receiving string s3 of length i, consisting of s1 positioned at the left of s3, s2 is used to fillout the remaining position of s3 to the right of s1, and is replicated as necessary, starting from the right. The last copy of s2 is truncated at the left if necessary to obtain the proper size. If the size of s1 is greater than i, s3 contains s1 truncated at the right end.

Error condition:
i greater than s1 length and s2 null

char *right(s1, i, s2, s3)
char *s1, *s2, *s3;

right is similar to left(s1, i, s2, s3), except that s1 is placed at the right of s3, s2 is replicated starting at the left of s3, with the truncation of the last copy of s2 at the right if necessary. If the size of s1 is greater than i, s3 contains s1 truncated at the left end.

Error condition:
i greater than s1 length and s2 null

char *center(s1, i, s2, s3);
char *s1, *s2, *s3;

center returns che receiving string s3 of length i, consisting of s1 centered in s3, s2 is used for filling on the left and right of s3 as in the functions above. If the size of s1 greater than i, s3 contains s1 truncated at the left and at the right to produce its center section. If s1 cannot be centered exactly in s3, it is positioned to the left of center.

Error condition:
i greater than s1 length and s2 null

In all the functions the range of i is between 0 and 2^{15}-1.

4.6 Other string operations

char *reverse(s)
char *s;

reverse returns the string s in reverse order.

char *trim(s, c)
char *s, *c;

trim returns the string s modifying in manner that consists of the initial substring of s with the omission of the trailing substring of s which consists solely of characters contained in c. c is a character set.

Error conditions:
string s null
c is not a character set
character set c null

char *map(s1, s2, s3)
char *s1, *s2, *s3;

map returns the string s1 modifying from a character mapping on s1, where each character of s1 that is contained in s2 is replaced by the character in the corresponding position in s3. Characters of s1 that do not appear in s2 are left unchanged. If the same character appear more than once in s2, the rightmost correspondence with s2 applies. The sizes of s2 and s3 must be the same.

Error conditions:
s2 length not equal s3 length
string s2 null
string s3 null

4.7 String comparison

String, like numbers, can be compared, but the basis for comparison is lexical (alphabetic) order rather than numerical value.
Lexical order includes all characters and is based on the ascii sequence. If a character c1 appears before c2 in the sequence, c1 is lexically less than c2.
The lexical order for single character strings is based on this order. This, X is less than x, but z is greater than x.
For longer strings, lexical order is determined by the lexical order of characters in corresponding positions, starting at the left. Two strings are lexically equal if and only if they are identical, character by character.
If one string is an initial substring of another, then the shorter string is lexically less than the longer one.

Unix supplies (see STRINGS(3)) all the necessary functions for string comparison:

strcmp(s1, s2)
char *s1, *s2;

strcmp compares its arguments and returns an integer greater than, equal to, or less than 0, according as s1 is lexicographically greater than, equal to or less than s2.

strncmp(s1, s2, n)
char *s1, *s2;

strncmp make the same comparison but looks at most n characters.

4.8 String analysis

Most programming operations on strings involve analysis.
There are two functions for identifying specific substrings.

match(s1, s2, i, j)
char *s1, *s2;

If s1 is an initial substring of s2[i:j], the function **match** returns the position of the end of the substring, else returns 0.

Error conditions:
position i is out of allowed range
position j is out of allowed range

find(s1, s2, k, i, j)
char *s1, *s2;

find returns the leftmost position in s2 where s1 is the Kth occurrence in the sequence of the positions, from left to right, at which s1 is a substring of s2[i:j].

Error conditions:
string s1 null
s1 length greater than s2[i:j] length
k out of allowed range (0 < k <= s2[i:j] length)
position i is out of allowed range
position j is out of allowed range

4.9 Lexical analysis

Lexical analysis involves sets of characters rather than substrings.
There are four lexical analysis functions.

any(c, s, i)
char *c, *s;

If the character following position i of s is contained in the character set c, **any** returns i+1 else returns 0.

Error conditions:
c is not a character set
character set c null
string s null
position i is out of allowed range

upto(c, s, k, i, j)
char *c, *s;

upto returns the leftmost position in s where a character of c is the Kth occurrence in the sequence of the positions, from left to right, at which a character of c occurs in s[i:j].
If the Kth occurrence does not exist, upto returns 0.

Error conditions:
c is not a character set
character set c null
string s null
k out of allowed range
position i is out of allowed range
position j is out of allowed range
positions i and j equals

many(c, s, i, j)
char *c, *s;

many returns the position in s after the longest initial substring of s[i:j] consisting solely of characters contained in c, where c is a character set. If the first character in s[i:j] is not contained in c, then many returns 0.

Error conditions:
c is not a character set
character set c null
string s null
position i is out of allowed range
position j is out of allowed range
positions i and j equals

bal(c1, c2, c3, s, k, i, j)
char *c1, *c2, *c3, *s;

bal returns the position in s of the Kth occurrence in the sequence of positions, at which successively longer balanced strings terminate inside s[i:j]. An occurrence of balanced is when a count is zero, and follow a characters in c1. A character in c2 causes the count to be incremented by one, while a character in c3 causes the count to be decremented by one. All other characters leave the count unchanged. If the count becomes negative or if the substring being examined is exhausted bal returns 0.

Error conditions:
c1 is not a character set
c2 is not a character set
c3 is not a character set
character set c1 null
character set c2 null
character set c3 null
k out of allowed range
position i is out of allowed range
position j is out of allowed range

4.10 Regular expression handling

A regular expression specifies a set of strings of characters.
A member of this set of strings is said to be matched by the regular expression.
In the following specification for regular expressions the word "character" means any character but null.

[1] Any character except a special character matches itself.
Special characters are the regular expression delimiter plus [. and sometimes ^ * $.

[2] A . matches any character.

[3] A \ followed by any character except a digit or (or) matches that character.

[4] A nonempty string s bracketed [s] (or [^s]) matches any character in (or not in) s. In s, \s has no special meaning, and] may only appear as the first letter. A substring a-b, with a and b in ascending ASCII order, stands for the inclusive range of ASCII characters.

[5] A regular expression of form 1-4 followed by * matches a sequence of 0 or more matches of the regular expression.

[6] A regular expression, x, of form 1-8, bracketed \(x\) matches what x matches.

[7] A \ followed by a digit n matches a copy of the string that the bracketed regular expression beginning with the nth \(matches.

[8] A regular expression of form 1-8, x, followed by a regular expression of form 1-7, y matches a match for x followed by a match for y, with the x match being as long as possible while still possible a y match.

[9] A regular expression of form 1-8 preceded by ^ (or followed by $), is constrained to matches that begin at the left (or end at the right) end of a line.

[10] A regular expression of form 1-9 picks out the longest among the leftmost matches in a line.

[11] An empty regular expression stands for a copy of the last regular expression encountered.

University California Berkeley supplies (see(REGEX.3)) all the necessary functions:

```
char *re__comp(s)
char *s;
```

re-comp compiles a string into an internal form suitable for pattern matching and returns 0 if the string s was compiled successfully.

Error conditions:
No previous regular expression
Regular expression too long
Unmatched \(
Missing]
Too many \(\) pairs
Unmatched \)

```
re__exec(s)
char *s;
```

re-exec checks the argument string against the last string passed to re__comp. Re__exec returns 1 if the string s matches the last compiled regular expression, 0 if the string s failed to match the last compiled regular expression, and -1 if the compiled regular expression was invalid.

5. EXAMPLES

5.1 Functions usage

The examples are to provide concrete instances, to show special cases that may not be clear otherwise, and to illustrate possibilities that may not be obvious.

For these reasons, some examples are not typical of ordinary usage.

any("prof","propagation",-8) returns 5
any("prof","propagation",4) returns 5
any("prof","propagation",5) returns 0
any("prof","propagation",1) returns error
any("","propagation",1) returns error
any("prof","",1) returns error
any("prof","propagation",14) returns error
any("prof","propagation",-13) returns error
any("","",14) returns error

atocset("propagation",c) returns the cset "proagtin"

bal("()abc+","(",")","(a)+(b)+(c)",1,1,0) returns 1
bal("()abc+","(",")","(a)+(b)+(c)",2,1,0) returns 4
bal("()abc+","(",")","(a)+(b)+(c)",3,1,0) returns 5
bal("()abc+","(",")","(a)+(b)+(c)",4,1,0) returns 8
bal("()abc+","(",")","(a)+(b)+(c)",5,1,0) returns 9
bal("abc+","(",")","(a)+(b)+(c)",3,1,0) returns 0
bal("+","(",")","(a)+(b)",1,2,0) returns 0
bal(ascii(),"(",")","(a)+(b)a",2,1,0) returns 4

center ("abcd",10,"yz",s) returns the string
"yzyabcdzyz"
center("abcd",4,"",s) returns the string "abcd"
center("abcd",2,"",s) returns the string "bc"
center("abcd",11,"zxyw",s) returns the string
"zxyabcdzxyw"
center("abcde",11,"zxyw",s) returns the string
"zxyabcdexyw"
center("abcde",10,"zxyw",s) returns the string
"zxabcdexyw"
center("abcde",10,"",s) returns error

delete("propagation",3,4) returns the string
"pration"
delete("propagation",4,3) returns the string
"proation"
delete("propagation",4,-3) returns the string
"pagation"

delete("propagation",-8,3) returns the string
"proation"
delete("propagation",-8,-3) returns the string
"pagation"
delete("propagation",-8,0) returns error
delete("propagation",4,10) returns error
delete("propagation",14,-3) returns error
delete("propagation",14,3) returns error

diffcset("abcde", "cdefg",c) returns the cset "ab"

find("a","abaaa",1,1,0) returns 1
find("a","abaaa",2,1,0) returns 3
find("a","abc",1,2,0) returns 0
find("a","abaaa",3,1,0) returns 4
find("a","abaaa",4,1,0) returns 5
find("bc","abcdeabc",1,3,0) returns 7
find("bc","abcdeabc",1,1,0) returns 2
find("ab","abcdeabc",2,1,0) returns 7
find("a","abcd",1,2,0) returns 0
find("ab","abcd",1,1,2) returns error
find("ab","abcd",5,1,0) returns error

incset('c',"abcde") returns 1
incset('c',"fghij") returns 0

If the string s is "element"
insert("house",s,2) returns the string "ehouselement"

intercset("abcde", "cdefg",c) returns the cset "cde"

iscset("centro") returns 1
iscset("contro") returns 0

itoa(32767,s) returns the string "32767"
itoa(-32767,s) returns the string "-32767"

ltoa(420009,s) returns the string "420009"
ltoa(-420009,s) returns the string "-420009"

left("abcd",7,"yz",s) returns the string "abcdzyz"
left("abcd",4,"z",s) returns the string "abcd"
left("abcd",2,"z",s) returns the string "ab"
left("abcd",7,"zwyx",s) returns the string "abcdwyk"

many("a","baabc",1,0) returns 0
many("a","aabc",1,0) returns 3
many("bc","abccbl",0,2) returns 6
many("a","abc",2,2) returns error

If the string s is "XXLC"
map(s,"IXLC","XCDM") returns the string "CCDM"
If the string s is ""
map(s,"IXLC","XCDM") returns the string ""
map(s,"IXLCD","XCDM") returns error

match("a","abc",1,0) returns 2
match("a","abc",2,0) returns 0
match("bc","abc",0,2) returns 4
match("","abcd",2,0) returns 2
match("bc","abcd",7,0) returns error

notcset("abcde",c) returns the cset generated by ascii() without the string "abcde"

If the string s is "null"
repl(s,3) returns the string "nullnullnull"

If the string s is "propagation"
reverse(s) returns the string "noitagaporp"

right("abcd",7,"yz",s) returns the string "yzyabcd"
right("abcd",4,"z",s) returns the string "abcd"
right("abcd",2,"z",s) returns the string "cd"
right("abcd",7,"zxyw",s) returns the string "zxyabcd"
right("abcd",7,"",s) returns error

subsbet("propagation",2,4,s) returns the string "ro"

subsfol("propagation",3,2,s) returns the string "op"
subsfol("propagation",-8,3,s) returns the string "pag"
subsfol("propagation",-8,0,s) returns the string ""

subsprec("propagation",3,2,s) returns the string "pr"
subsprec("propagation",4,3,s) returns the string "pro"
subsprec("propagation",-8,3,s) returns the string "pro"
subsprec("propagation",-8,0,s) returns the string ""

If the string s is "propagat"
substitute("house",s,2,7) returns the string "phouseat"
substitute("house",s,2,8) returns the string "phouset"
substitute("house",s,2,2) returns the string
"phouseropagat"
substitute("house",s,2,0) returns the string "phouse"
substitute("",s,2,0) returns the string "p"
If the string s is ""
substitute("house",s,1,0) returns the string "house"

If the string s is "left abaca"
trim(s,"abc") returns the string "left"
trim("abcd e","e") returns the string "abcd"

unioncset("abcde", "cdefg",c) returns the cset
"abcdefg"

upto("abcd","abcd",1,1,0) returns 1
upto("abcd","abcd",2,1,0) returns 2
upto("abcd","abcd",4,1,0) returns 4
upto("cd","abcd",1,1,0) returns 3
upto("cd","abcd",2,1,0) returns 4
upto("d","abcd",1,2,3) returns 0
upto("a","abcd",1,2,0) returns 0
upto("abcd","abcd",5,1,0) returns error

5.2 Roman numbers

This program converts Arabic numerals to corresponding Roman numerals.
The method of solution is due to Gimpel.
Each digit of the arabic number is mapped into its Roman equivalent.
The multiplication by 10 represented by successive po-

sitions in the Arabic number is reflected in the corresponding Roman numeral by shifting to the next "octave" using character replacement.
The occurrence of an asterisk in the result indicates a number that is too large to be represented by a Roman numeral.

```
/* Given an Arabic number this procedure returns the
corresponding Roman numeral.

The main program allocates space to the variable "re-
sult" */

char *roman(arabic)
register char *arabic;

    register i;
    static char *equiv[10] ={("","I","II","III",
        "IV","V","VI","VII","VIII","IX","X")};
    char *map();
    extern char *result;

        if (atoi(arabic) <= 0)
            return("cannot convert");
        strcpy(result,"");
        for (i = 0; i < strlen(arabic); i++){
            strcpy(result,map(result,"IVXLCDM",
            "XLCDM**")};
            strcat(result,equiv[arabic[i] - '0']);
        }
        if (find("*",result,1,1,0)
            return("cannot convert");
        else
            return(result);
```

6. ACKNOWLEDGMENTS

I would like to thank 0. Babaoglu for his comments and constructive criticism.

7. REFERENCES

[1] C.A. Coutant, R.E. Griswold and S.B. Wampler: Reference Manual for the Icon Programming Language Version 5 TR of Department of Computer Science, University of Arizona.

[2] J.F. Gimpel Algorithms in SNOBOL4 John Wiley & Sons, New York, 1976 pp. 25-26

[3] Unix Time-Sharing System: Unix Programmer's Manual Seven Edition, Volume 1, January 1979

AVVENIMENTI

INTERNATIONAL SYMPOSIUM ON LOCAL COMPUTER NETWORKS

A. Mattasoglio*, M. Meli†

1. PREMESSA

Il convegno "Internation In-depth Symposium on Local Computer Networks" si è svolto a Firenze dal 19 al 21 Aprile 1982, organizzato dall'I.F.I.P. TC 6, con la collaborazione tecnica dell'OLIVETTI OLTECO. Una documentazione completa del convegno può essere trovata nel libro intitolato:

LOCAL COMPUTER NETWORKS
curato da P.C.R. Ravasio, G. Hopkins e N. Naffah
North-Holland - 1982

2. CIRCUITI VLSI PER L'ACCESSO A RETI LOCALI: IL CASO AMD

La disponibilità sul mercato di circuiti integrati a basso costo è sicuramente uno dei fattori chiave per l'affermazione delle reti locali dal punto di vista commerciale. A tutt'oggi il solo Cambridge Ring dispone di un chip funzionante, anche se non commercializzato in grande scala. La rete ETHERNET avrà a breve termine il chip prodotto dalla INTEL (si pensa entro l'anno), ma anche altri costruttori progettano e addirittura hanno già preparato il loro ingresso in questo allettante mercato.

Uno di questi costruttori è l'AMD: essa ha descritto la sua nuova famiglia di prodotti architetturalmente proprio al congresso, e disporrà, sembra, di quantitativi per valutazione entro la fine dell'anno in corso.

Il mercato delle reti locali è, secondo la AMD, diviso in due fascie principali:

a) Reti Frontend: sono le reti locali a cui, di solito, si pensa esse interconnettono molte apparecchiature, in generale con una limitata intelligenza locale ed a basso costo. Richiedono velocità sino a 10 Mb/S. Tipici sistemi collegati a tali reti sono terminali, stampanti, fotocopiatrici, ecc.

b) Reti Backend: sono le reti che interconnettono tra loro varie CPU con sistemi ad alta velocità quali cluster di dischi, stampanti laser, ecc. Si tratta di sistemi a costo molto più elevato e le prestazioni richieste sono da un minimo di 10 Mb/s ad un max di 1Gb/s, con ottime caratteristiche di flessibilità nell'implementazione.

2.1 Le reti front end

È chiaro che un costruttore, per far fronte agli elevatissimi costi di sviluppo di nuovi componenti, deve scegliere per gli stessi delle architetture il più possibile multifunzioni, ma all'interno di un ben specifico range di applicazioni. La scelta AMD è andata sull'automazione d'ufficio.

L'architettura da loro proposta consta di 3 chip (fig. 2.1):
- un transcreiver situato nel Tap di rete;
- un Serial Interface Adapter (SIA) che codifica i CRC di ricezione e trasmissione;
- un Local Area Network Protocol Chip (LANCP) che implementa il protocollo Data Link della rete locale e interfaccia un microprocessor a 16 bits che realizza i protocolli di livello superiori.

Una simile architettura a 3 chip può essere utilizzata per varie reti locali: AMD per ora suppone ETHERNET, ma non è scartata la possibilità di supportare un altro tipo di rete con la semplice sostituzione del transceiver e del LANPC, mantenendo il SIA. Come si vede in figura 2-1, l'interfaccia tra il LANPC e il SIA è seriale e consta di soli 4 fili, due per la trasmissione seriale full duplex e 2 che forniscono le temporizzazioni. Il clock di trasmissione è generato da un oscillatore collegato al SIA, quello di ricezione proviene dal mezzo trasmissivo.

* CILEA, Segrate
† Ist. di Cibernetica, Università di Milano

Il LANPC permette 3 funzioni di indirizzamento a livello link: logico, fisico e promiscuo.
L'indirizzamento fisico è quello standard dell'ETHERNET e consente 1.4*10**10 indirizzi distinti, comuni con altri nodi. Il modo promiscuo fa considerare ogni indirizzo come proprio e permette di usare il nodo come stazione di monitoring. Il modo logico permette che la stazione risponda ad un gruppo di un massimo di 64 indirizzi, che sono caricati in un'apposita tavola dal software e vengono controllati dall'hardware di ricezione mediante una tecnica hash.

Il LANPC permette la generazione ed il controllo di FRAMES CHECKING SEQUENCES e contiene i circuiti per la realizzazione dell'algoritmo di Exponentially Binary Back-off per i tentativi ripetuti in caso di collisione.

Molto versatile è anche il progetto dell'interfaccia con il microprocessore e la memoria per la realizzazione di protocolli di livello superiore. Va subito sottolineato anche l'interfaccia a 16 bits con il microprocessore è programmabile e quindi, secondo loro, facilmente adattabile sia a bus multiplexati che non multiplexati.

Il microprocessore e il LANPC formano un'architettura strettamente connessa tramite la memoria, che viene divisa in 3 settori logici

1) Area di inizializzazione: contiene vari campi tra cui il numero di Buffers

2) Descrittori dei buffers: sono 2 code circolari (una in ricezione e l'altra in trasmissione) di n. posti. Contengono anche i puntatori ai singoli buffers.

3) Aree di buffer

Dopo essere stato inizializzato dal microprocessore, in trasmissione il LANPC esegue un polling sulla lista dei descrittori, sino al primo marcato valido nella lista di trasmissione. Esegue allora la trasmissione del buffer puntato dal descrittore, che si troverà in una qualsiasi posizione di memoria, anche disgiunta dagli altri buffers. Se la trasmissione va a buon fine il LANPC azzera il bit di buffer valido nel descrittore corrispondente e ricomincia il polling. Il microprocessore controlla il risultato delle trasmissioni.

Il funzionamento in ricezione è analogo, ricordando che in questo caso è il LANPC a scrivere informazioni di stato e di lunghezza messaggio nel descrittore corrente (questo compito era evidentemente del microprocessore in trasmissione).

La scheda di un nodo ETHERNET, costruita con questi integrati dovrebbe avere 65-70 componenti e costare nell'ordine delle centinaia di dollari; una scheda equivalente nella tecnologia attuale, porterebbe a costi dell'ordine delle migliaia di dollari.

2.2 Le reti back end

Non esistono ancora standard correnti in questo campo (esiste solo una proposta di standard ANSI, vedi il paragrafo sugli STANDARD). Le scelte AMD riguardano una topologia flessibile (sia a bus che a ring) con frame completamente programmabile; è così possibile variare la posizione, la lunghezza e la semantica dei vari campi del frame.

I componenti per la sua realizzazione sono:
- SIA;
- Bit Stream Controller (BIA), che esegue l'inserzione/cancellazione di bits in linea, la conversione Bit/Bytes e la garbage collection per i ring;
- Access Controller Chip (ACC), che regola l'accesso al mezzo trasmissivo;
- Link Protocol Process (LPP), che esegue l'indirizzamento di link, i trasferimenti in DMA tra i buffer in memoria e calcola le CRC.

Il calcolatore che esegue i protocolli di livello superiore è questa volta una scheda microprogrammabile con componenti della serie AMD 2900.

La velocità è selezionabile ed ha un max di 50 Mb/s.

3. LA PROPOSTA IBM PER UNA RETE A RING CON CONTROL TOKEN

Grande risalto è stato dato durante il convegno alla rete locale costruita presso i laboratori IBM di Zurigo; sono state presentate 3 relazioni su di essa, trattanti i dettagli H/W e S/W, la valutazione delle prestazioni, la validazione del protocollo di comunicazione.

L'architettura da loro scelta è quella di una rete ad anello a 4 Mb/s, con trasmissione su twisted pair schermato. L'accesso al ring è regolato da un protocollo a free-token con monitoring e error recovery; viene reso disponibile, oltre ad un servizio tipicamente datagram, anche un servizio di trasmissione sincrono a larghezza di bande costante (tipicamente 64Kb/s) per applicazione real-time (es. le trasmissioni vocali). Tutte queste caratteristiche saranno spiegate più dettagliatamente in seguito.

3.1 Architettura fisica

Gli elementi costitutivi della rete sono:

Ring: è la sottorete effettiva;
Block switch: è l'elemento di connessione tra i Ring;
Bridges: fornisce il necessario buffering nel trasferimento tra i ring.

La topologia è di più rings collegati tra loro via block switch e bridges (locali o remoti).

Ogni singolo ring è più propriamente uno star-shaped ring, cioè una rete a anello che unisce degli elementi passivi detti pannelli di distribuzione, dai quali si dipartono dei lobi, uno per ogni device. I lobi sono formati da due tratte di doppino telefonico, una in ingresso e l'altra in uscita, che collegano il pannello di distribuzione al ring adapter; un tale sistema di collegamento conserva così la topologia a ring, permettendo tuttavia una riconfigurazione dinamica della rete. In caso di break down di un ring adapter, è possibile comandare dei "relais" nei pannelli di distribuzione che escludono il lobo contenente il ring adapter non funzionante. Questo tipo di architettura non è nuovo, è stato infatti studiato all'MIT nell'ambito del progetto di rete per il Laboratory for Computer Science.

I vantaggi di una tale scelta sono fondamentalmente nella facile manutenzione e riconfigurabilità di un tale tipo di rete, che risolve numerosi problemi di installazione. Sembra, tuttavia, che esistano dei problemi:

a) La disconnessione della linea di un lobo viene compiuta da "relais", che interrompono fisicamente la connessione elettrica, isolando il nodo. Tale disconnessione può provocare dei disturbi che si propagano in linea, alterando lo stato corrente, e obbligano la parte receiver dei ring adapter in una condizione di recovery con durata media dell'ordine di 2ms.

b) La presenza dei "relais" nel cammino del segnale altera la propagazione elettrica dello stesso.

c) Allo scopo di minimizzare i possibili problemi di cui ai punti a) e b), è necessario l'uso di componenti sì passivi (quindi meno soggetti a guasti), ma di tolleranze minime e al limite delle prestazioni, con tutti i possibili problemi che una tale scelta può comportare.

All'interno dell'adapter sono implementati anche tutti i protocolli relativi all'accesso alla rete (physical e data link layer), compresi i meccanismi di error detection e recovery del protocollo di accesso.

3.2 Block switch e Bridges

La sottorete nasce già come l'interconnessione di vari ring. Per realizzare una di tali connessioni si usano i block switch, cioè dei commutatori di frames ad alta velocità tra i rings ed i 2 bridges, uno per ring, che contengono due code FIFO di buffer ciascuno per i frames in trasmissione e in ricezione. Il block switch non è un vero e proprio commutatore di pacchetti, perchè non esegue funzioni di Link Control e Routing; ciò permette di costruire un Block Switch con un hardware particolarmente semplice e con una grande velocità di commutazione (il thruput di un Block Switch è circa di un ordine di grandezza superiore a quello di un ring). Un Block Switch può collegare più di due rings, anche distanti tra loro, utilizzando più bridges, locali e remoti. Nel caso che i ring siano pochi, la funzione di switching del block switch può essere sostituita da una rete completamente connessa di commutazione tra i bridges.

Un esempio di collegamento block switch/bridges può essere visto in figura 3.1 ci sono 2 collegamenti di questo genere, per ogni coppia di rings, in modo da avere comunicazione full duplex.

Vediamo ora di comprendere meglio il funzionamento:

PORTS: Collegano i FIFO Buffers al Block Switch e contengono la logica necessaria al trasferimento dei frames.

ACCESS CONTROL UNIT: Esegue la scansione delle porte per vedere se c'è un frame da trasmettere. Cede il controllo alla LOOK AHEAD UNIT quando ne trova uno.

LOOK AHEAD UNIT: Quando attivata dalla ACU, legge nel Bus DABI (Destination Address Bus In) l'indirizzo del destinatario del frame; determina grazie ad esso e ad una tabella interna di routing l'indirizzo della stazione ricevente e lo emette sul Bus DABO (D.A.B.Out) (In ricezione il funzionamento è simile).

DATA TRANSFER UNIT: Una volta che la LOOK AHEAD UNIT ha marcato un frame come trasferibile, la Data Transfer Unit esegue il trasferimento effettivo.

3.3 Protocolli di comunicazione

Vi sono due possibili servizi di trasmissione: uno asincrono e l'altro sincrono. In trasmissione, nel modo asincrono, al passaggio di un token libero, un ring adapter che ha un frame da trasmettere passa dallo stato di REPEATER a quello di TRANSMITTER, cambia il token in occupato e trasmette il frame, interrompendo il ring con un switch elettronico. Al termine della trasmissione di possono presentare i casi seguenti:

a) L'adapter ha riletto di ritorno dal ring il suo pacchetto correttamente quindi il messaggio non è stato disturbato.

b) L'indirizzo del destinatario del frame è diverso da quello voluto dal trasmittente (si è avuto quindi un errore nel protocollo di accesso).

c) Il bit di priorità nel Transport Control Field del Frame è nullo.

Nel caso a) l'adapter genera un frame a zero con il token libero, mentre nei casi b) e c) l'adapter in trasmissione non genera un token libero.

Dopo aver generato il frame e l'eventuale token libero, il ring adapter trasmette un segnale vuoto sinchè non riceve il prossimo delimeter di chiusura. Esso è il delimeter del PROPRIO frame appena trasmesso, se non ci sono stati errori. Infine l'adapter torna in modo REPEATER e chiude lo switch.

Se un adapter è in modo REPEATER, ritrasmette con ritardo di un bit i frame che passano e controlla l'indirizzo del destinatario: se incontra il suo, copia semplicemente il frame senza modificarlo.

Nel caso SINCRONO, una stazione deve richiedere un tale tipo di connessione ad una stazione di controllo, che la concede a seconda del carico di rete. Va premesso, per comprendere più a fondo il funzionamento, che, al fine di ottenere affidabilità nel protocollo di link, vi sono sempre due tipi di stazioni attive sul ring: la stazione di monitoring, che è addetta al controllo ed all'eventuale recovery del token, e la stazione trasmittente. Tutti i ring adapter possono essere stazioni di monitoring, ma ce ne può essere attiva solo una ad ogni momento. Solo il monitor esegue delle effettive azioni di recovery, mentre tutte le altre stazioni, nel caso si accorgano di un errore, si limitano a passare in modo REPEATER senza eseguire alcuna azione sul token.

Un altro importante compito della stazione di monitor è quello di generare dei token con priorità che regolano la trasmissione sincrona. Sono possibili due casi:

a) all'inizio della trasmissione sincrona il token è libero; in tal caso il monitor lo marca token con priorità;

b) se invece il token è occupato, il monitor non genera un token libero finchè il frame non è stato trasmesso, ma marca lo stesso il bit di priorità, impedendo così alla stazione trasmittente di generare un token libero e facendole invece trasmettere un segnale vuoto. Quando il monitor sente il delimiter di fine frame, genera un token con priorità, libero.

Dopo la trasmissione del token con priorità libero, la stazione chiamata marca occupato il token, scrive gli indirizzi di chiamante e chiamato ed i dati. La stazione chiamata controlla, come al solito, il traffico passante e, quando riceve l'indirizzo chiamante, copia i dati ed inserisce i propria dati nello stesso frame. All'arrivo di tali dati, il chiamante, invece di eliminare semplicemente il frame, copia il campo dati e quindi elimina il frame e genera un token (libero come nella trasmissione asincrona. Dopo che tutte le stazioni coinvolte in una trasmissione sincrona hanno avuto l'opportunità di trasmettere, il monitor elimina il token con priorità libero e termina la trasmissione sincrona.

3.3.1 Conclusioni

Le scelte del gruppo IBM di Zurigo sono state sicuramente influenzate dalle analisi da loro eseguite sulle prestazioni delle sottoreti. Alcuni dei loro risultati si possono trovare in:

W. Bux Local-area Subnetworks: a performance comparaison.
IEEE TRANS. ON COMMUNICATIONS, VOL. COM 29 (1981) PP.1465,1473

Tali risultati, come molti ottenuti mediante simulazione, si prestano a varie interpretazioni. I detrattori delle scelte a token affermano che un tale tipo di protocollo impedirebbe la trasmissione di segnali in tempo reale, come ad esempio quello vocale, a causa del ritardo insito nel protocollo. I risultati sperimentali del prototipo IBM diranno se il modo di trasmissione sincrono sarà sufficiente a sopperire ai ritardi, senza per questo degradare eccessivamente le prestazioni della trasmissione asincrona. Inoltre il protocollo è stato accuratamente validato e quindi da loro ritenuto molto affidabile.

Tuttavia la lunga serie di controlli può contribuire a diminuire sensibilmente le prestazioni.

4. COMUNICAZIONE VOCALE SU RETI LOCALI

L'Olivetti OLTECO ha studiato la possibilità di implementare le funzioni di un centralino di commutazione telefonica digitale impiegando la sua rete locale Ethernet, compatibile con lo standard Digital, Intel, Xerox. Tale studio ha come obiettivo l'integrazione di tutte le funzioni di comunicazione di un ufficio di medie dimensioni in un unico mezzo trasmissivo.

4.1 Hardware impiegato

Il nodo che permette l'accesso al cavo coassiale, nella terminologia Olivetti, si chiama NIM (Network Interface Module). Esso contiene l'hardware che esegue il protocollo CSMA/CD e schede specializzate diverse, a seconda dei devices che devono essere connessi alla rete.

Nel caso telefonico c'è un modulo che permette il collegamento di un massimo di 8 telefoni, comprendendo un CODEC che converte il segnale analogico in segnale digitale a 64 Kb/s e lo memorizza in un buffer. Un micro-processore provvede alla gestione delle varie operazioni e all'inoltro delle richieste di segnalazione: selezione di un nuovo numero, invio del segnale di occupato.

4.2 Protocolli impiegati nella comunicazione

Il protocollo implementato sopra al protocollo standard CSMA/CD è chiamato RT ed è svolto integralmente da componenti hardware. Esso provvede a ricevere pacchetti dalla rete ed a bufferizzarli nel buffer di

ricezione; quando ne ha almeno 3, li invia al CODEC che li converte per l'inoltro al telefono. Se il buffer diventa vuoto ripetutamente, il protocollo segnala il problema agli strati superiori del sistema, i quali provvederanno all'abbattimento della comunicazione mediante il protocollo di segnalazione SCP.
In trasmissione, il protocollo RT inoltra un pacchetto non appena lo riceve dal CODEC; se, per problemi di collisione, RT non riesce ad inviare il pacchetto entro il tempo di 3 pacchetti, tale pacchetto viene scartato e ne vengono informati gli strati superiori di protocollo. Il ritardo di 3 pacchetti, dalla ricezione nel buffer all'inoltro al CODEC, viene introdotto per assorbire la varianza del ritardo tra un pacchetto e l'altro, dovuta alla natura statistica dell'accesso al canale di comunicazione.

4.3 Problemi nell'utilizzo di Ethernet per la voce

Sono già stati fatti degli studi per il trasporto in real-time di voce su reti a commutazione di pacchetto e si è scoperto che il principale problema è costituito dai ritardi nella trasmissione. I ritardi sono di due tipi:

- ritardo di pacchettizzazione. Un pacchetto non può essere inviato in rete fino a che non è completo. Questo inconveniente induce a ritenere desiderabili pacchetti piccoli, ma la trasmissione su una rete locale Ethernet è tanto più efficiente quanto più lunghi sono i pacchetti trasmessi. Il compromesso scelto è di inviare in rete pacchetti di 64 bytes.
  ritardo di trasmissione. Esso è variabile con il carico e, per la natura statistica del metodo di accesso CSMA/CD, non può essere limitato superiormente.

Un altro problema che si presenta è il fenomeno della sincronizzazione. Supponiamo di avere 20 telefoni attivi sul cavo, che il ritardo di pacchettizzazione sia di 8 ms e che le richieste della linea siano uniformemente distribuite negli 8 ms. Ciò significa che c'è un pacchetto da inviare ogni 400 micro secondi. Se, per intervento di una stazione dati con un pacchetto lungo da inviare, un pacchetto voce è ritardato più di 400 micro secondi, è sicura la collisione con il pacchetto successivo e così via, e si degrada quindi la capacità del canale.

L'accettabilità del ritardo per un uso normale dei telefoni varia moltissimo a seconda del tipo di telefono impiegato: per telefoni a 2 fili, che presentano riflessioni e quindi echi, questo ritardo deve essere contenuto entro gli 80 millisecondi al massimo. Molto più tollerabile esso risulta impiegando telefoni a 4 fili e/o soppressori di eco.

4.4 Un migliore protocollo di comunicazione

Si può pensare di migliorare il comportamento della rete impiegando un altro tipo di protocollo di comunicazione, l'MVP. Questo protocollo sfrutta il fatto che in un NIM telefonico sono presenti più telefoni, e quindi i pacchetti voce possono venir raggruppati in pacchetti più grandi contenenti più pacchetti. Sfruttando le proprietà dell'indirizzamento Multicast offerto dalla rete locale, è possibile anche che i pacchetti siano destinati a telefoni su NIM diversi.

4.5 Conclusioni

Un esteso studio di simulazione ha mostrato che l'ipotesi di impiego di una rete locale come centralino di commutazione telefonica distribuito è possibile per configurazioni con un numero limitato di telefoni e un traffico dati non troppo pesante (inferiore al 20-30% della capacità del canale). In tali condizioni possono essere supportate 30-40 conversazioni telefoniche contemporaneamente, e così si possono collegare circa 150-200 telefoni.

Le possibilità di mercato di un tale sistema sono interessanti, perchè il 60% di tutti i centralini telefonici del mondo hanno capacità comprese tra 150 e 700 telefoni.

5. RETI LOCALI E MODELLO O.S.I.

Il modello O.S.I. è, secondo il Prof. Danthine, orientato ad uno stile di comunicazione mediante circuiti virtuali e, in particolare, il livello "Data Link" è stato grandemente influenzato dall'esistente protocollo HDLC. Per le reti locali il livello "Data Link" è diverso, ed in una singola rete locale il concetto stesso di routing perde il significato per la natura broadcast del mezzo trasmissivo.

Si è quindi tentati di limitare il modello O.S.I. di una rete locale ai primi 2 layer del modello O.S.I., come sta facendo l'IEEE con il suo comitato di standardizzazione per le reti locali 802. Il livello "Data Link" viene diviso in due sotto-livelli: il "Logical Link Control" ed il "Medium Access Control". L'introduzione del sottolivello MAC permette di separare le funzioni di framing e sincronizzazione, che dipendono dal mezzo trasmissivo e dalla topologia usata, da quelle di controllo del collegamento logico. È stato spesso detto che le reti locali offrono un servizio di tipo datagram, ma si tratta di un servizio datagram di alta qualità, in cui la probabilità di pacchetti persi o fuori sequenza è molto bassa. Se si vuole, però, far comunicare due reti locali con formati dei pacchetti e lunghezze differenti a Link Level, è necessario definire a livello Network, un sottolivello di Internetworking, che si occupi del routing e dell'eventuale frammentazione per permettere la comunicazione tra reti con lunghezze massime di pacchetto diverse.

5.1 Reti per siti estesi

Viene puntualizzata l'esigenza di reti per collegare siti dispersi geograficamente all'interno di una città, separati da distanze dell'ordine della decina di Km., ma di-

pendenti dalla stessa amministrazione, come potrebbe essere un'Università o il centro direttivo di una grande industria.

Le specifiche di una tale rete dovrebbero essere le seguenti:

- 3000 stazioni di lavoro
- 300 stazioni di acquisizione dati o minicomputer
- 15 computer di media grandezza per servizio time sharing
- 1 grande centro di calcolo con un insieme di mainframes

La tecnologia necessaria per la realizzazione di una simile rete probabilmente non esiste ancora ed è probabile che vengano adottate soluzioni parziali mediante le reti locali attuali adatte per un singolo edificio. La rete di cui si parla ne dovrà permettere l'interconnessione mediante il protocollo di internetworking O.S.I. fig. 5.1, di cui si avverte l'esigenza, che costituirà il legame indispensabile per il collegamento di queste reti disgiunte.

Il tipo di servizio offerto da questa rete sarà di tipo datagram. Se si vorrà connettersi a risorse esterne alla rete, sarà necessario servirsi di gateway fig. 5.2, che realizzino, mediante servizi offerti dal livello di trasporto, le capacità di circuito virtuale indispensabili per connettersi a reti pubbliche del tipo X.25.

5.2 Naming e indirizzamento nelle reti

Secondo J.H. Saltzer esistono 4 classi di oggetti che possono avere un nome in un ambiente di comunicazione dati

- servizi che sono i programmi utente o di sistema destinazione logica dei dati trasmessi;
- nodi che sono i calcolatori su cui detti programmi sono in esecuzione;
- punti di accesso alla rete che sono le interfaccie fisiche di collegamento dei calcolatori alla rete;
- vie che rappresentano l'itinerario che i pacchetti devono percorrere per raggiungere i punti di accesso alla rete.

Ognuno di questi oggetti ha un diverso meccanismo di naming cioè di collegare un simbolo (stringa di caratteri o sequenza di bits all'entità fisica di cui tale simbolo è rappresentante.

Molto spesso tali meccanismi non sono apparenti perchè il naming viene dato per scontato e immutabile (i nomi dei servizi e degli host viene molto spesso "cablato" dentro programmi e sistemi operativi e non può essere cambiato che con estese modifiche ai simboli) o perchè i due oggetti hanno lo stesso nome (vedi punto di accesso alla rete e nodo in Ethernet con la buffa conseguenza che lo stesso calcolatore non può avere due interfacce Ethernet sulla stessa rete perchè non c'è modo di distinguerle).

I name-server prevedendo la possibilità di effettuare legami dinamici tra i vari oggetti presenti in una rete aggiungono flessibilità ed affidabilità ai servizi presenti in rete. Grande cura deve però essere messa nel loro progetto per non incorrere negli inconvenienti citati. Un altro esempio citato è quello di ARPANET in cui il nome del servizio si identifica con il nome del punto di accesso di rete.

6. PROTOCOLLI UTILIZZATI IN NET-ONE

La relazione svolta da J.M. Davidson è stata molto interessante, perchè presenta l'architettura interna della prima rete locale commercialmente disponibile di impiego generale, programmabile dall'utente.

La relazione descrive i protocolli orientati alla connessione che realizzano in Net-One l'astrazione dei circuiti virtuali tra 2 porte qualsiasi della rete. La rete è basata su nodi intelligenti detti NIU, che contengono un processore Z-80 con 64 KB di memoria, con varie interfacce parallele e seriali ed un accesso alla rete locale CSMA/CD in banda base a 10 MHz, compatibile con lo standard Ethernet (Digital, Intel, Xerox).

Ogni processore fornisce un ambiente di programmazione multitasking, in cui sono eseguiti un insieme di processi. Ogni processo è identificato univocamente da una terna

<rete,NIU,identificatore processo>

Il sistema operativo fornisce un meccanismo di comunicazione interprocesso IPC non affidabile, per mezzo del quale i processi in esecuzione su un processore della rete possono inviarsi l'un l'altro messaggi di limitata lunghezza.

6.1 Servizi offerti da Net-One

Quando i processi hanno bisogno di comunicare tra di loro in maniera affidabile, possono utilizzare i "circuiti virtuali" invece dell'IPC. I "circuiti virtuali" forniscono un servizio di comunicazione affidabile full duplex, e possono venir creati e distrutti dinamicamente.

Ogni periferica collegata a Net-One ha un processo che la rappresenta e quindi, quando si vuole connettere fra di loro due periferiche, che possono essere l'interfaccia seriale di un calcolatore ed un terminale, è necessario collegare i due processi che li gestiscono. Tutti i protocolli che permettono questa connessione sono comunque basati sul servizio di IPC visto precedentemente.

6.2 Protocolli di rete

I protocolli che permettono l'esistenza dei circuiti virtuali sono essenzialmente 4:

- NP (Name Inquiry Protocol) per trovare con chi connettersi;
- RP (Rendezvous Protocol) per stabilire la connessione;
- CP (Connection Inquiry Protocol) per conoscere lo stato di una connessione;
- DTP (Data Transport Protocol) per l'uso di una connessione aperta.

6.2.1 Name Inquiry Protocol

Non si tratta di un protocollo di Name server completamente generale, perchè risulta polarizzato verso la ricerca di una risorsa libera per la connessione. Esso impiega una base di dati distribuita utilizzando tavole statiche così definite:

<nome,identificatore_risorse,"unico">

La relazione implementata è una relazione molti-a-molti. Ogni nome, infatti, può puntare a più risorse ed ogni risorsa può avere molti nomi. Il flag "unico" segnala se il nome è unico, cioè se il nome punta a quell'unica risorsa o meno.

6.2.2 Connection Inquiry Protocol

Serve per vedere se una data risorsa è impegnata in una connessione. Si tratta di un'interazione punto-a-punto ed ora è utilizzata solo per il comando di rete EXAMIN.

6.2.3 Rendezvous Protocol

È utilizzato dal software di rete per iniziare ed abbattere le connessioni. Esso prevede più scambi di messaggi per creare sicuramente una connessione. Non essendo questa un'operazione idempotente, i duplicati dei messaggi costituiscono un problema. Non sono i processi che gestiscono i devices che stabiliscono la connessione; essi si servono di un processo "Rappresentante" che, in cooperazione con un processo "Operatore" (centralinista), fissa l'"appuntamento" tra i due "Rappresentanti" dei devices e quindi la connessione.

L'"operatore" è sempre quello che esegue le richieste, mentre il "Rappresentante" risponde con un ACK contenente le informazioni. Per difendersi da possibili messaggi duplicati e recapitati in ritardo, entrambi i "Rappresentanti" danno un numero alla connessione in corso di stabilimento. Il numero è legato ad un clock della macchina, in modo che sia estremamente improbabile il fatto che due connessioni successive su uno stesso processore abbiano lo stesso numero.

6.2.4 Data Transport Protocol

Su una connessione aperta mediante il "Rendezvous Protocol" i messaggi vengono scambiati in pacchetti dai processi gestori dei devices. Questi pacchetti prevedono nel loro header, vedi fig. 6-1, numeri di sequenza per prevenire duplicati o pacchetti in disordine, ack per prevenire pacchetti persi (un ack può riconoscere un certo numero di pacchetti fino al numero di sequenza precisato nel campo), un'informazione di flow control, che dice all'interlocutore il massimo numero di sequenza che il mittente può accettare, oltre, naturalmente, ai dati da inviare.

6.2.5 Conclusioni

La divisione dei compiti tra Rendezvous Protocol e Data Transport Protocol può apparire inusuale nel normale contesto dei protocolli di trasporto in cui queste funzioni sono svolte da un unico agente (la Transport Station), ma presenta dei vantaggi implementativi notevoli. L'aver confinato le complesse operazioni di apertura e chiusura di una connessione in un protocollo a parte permette, infatti, di avere un protocollo per trasporto dati semplice e veloce, mentre le operazioni dell'Operatore e del Rappresentante possono continuare un background a più bassa priorità per lo stabilimento di una nuova connessione.

7. LAVORO DEI COMITATI DI STANDARDIZZAZIONE

Al dibattito hanno partecipato come moderatori Dave Carlson dei Bell Labs, in rappresentanza dell'IEEE, Dave Bradley per l'ECMA, e Gary Anderson, che ha illustrato il lavoro dell'ANSI.

7.1 Il Comitato IEEE 802

Si è trattato di un lavoro estremamente impegnativo cui hanno partecipato in 2 anni circa 500 persone, che ora sono ridotte a circa 100-120. Essi hanno compilato un documento quasi pronto per l'approvazione finale. Il Comitato è composto da diversi sottocomitati:

- media
- DLMAC
- High Level Protocol

Per il Comitato IEEE 802 vale la seguente definizione:

- Una rete locale è un sistema di comunicazione dati in un'area geografica di dimensioni limitate che for-

nisce un canale di comunicazione di prestazioni da moderate ad alte con un basso tasso di errore.

7.1.1 Requisiti funzionali

Gli obiettivi proposti sono i seguenti:

- indipendenza da mezzo e topologia
- comunicazione diretta tra le parti
- applicazioni per il commercio e l'industria leggera
- possibile coesistenza di sistemi diversi interagenti
- aderenza al modello O.S.I. e relativi standard

Le applicazioni prefigurate sono le seguenti:

- File transfer
- Grafica
- Word Processing
- Electronic mail
- Voce digitalizzata

Tecnicamente lo standard si prefigge di occuparsi di apparecchiature dotate dei seguenti requisiti:

- 1-20 Mb/s data rate
- 2 Km distanza massima coperta
- + di 200 apparecchiature connesse
- connessione di Server in grado di gestire simultaneamente richieste multiple

7.1.2 Modello proposto

Lo standard inquadra le reti locali in un modello a 2 livelli che si inseriscono nei primi 2 livelli del modello O.S.I., il Fisico ed il Data Link.

Il livello Data Link proposto è poi strutturato in 2 sottolivelli: lo strato Logical Link Control, che tratterebbe dei problemi di error detection e recovery, flow control etc., e lo strato Medium Access Control, per cui sono emerse 3 alternative:

- CSMA/CD
- TOKEN BUS
- TOKEN RING

Il protocollo Logical Link Control sarebbe unico per tutti questi tipi di metodi di accesso al mezzo di comunicazione fisica.

Sono stati definiti 3 tipi di procedure Logical Link Control:

- tipo 1 : datagram
- tipo 2 : circuiti virtuali
- tipo 3 : datagram con riconoscimento

7.1.2.1 Mezzi di comunicazione

Sono state inoltre fissate le specifiche per l'impiego dei seguenti mezzi di comunicazione con i protocolli precedentemente visti:

- CSMA/CD
  - cavo coassiale in banda base
  - cavo coassiale a radiofrequenza
- Token bus
  - cavo coassiale in banda base
  - cavo coassiale a radiofrequenza
- Token ring
  - cavo coassiale in banda base
  - doppino telefonico in banda base

7.1.3 Scadenze

Lo standard 802, che in realtà rappresenta un complesso insieme di norme e nozioni progettuali, verrà presentato per l'approvazione finale all'incontro di Boulder Colorado nel prossimo settembre.

7.2 ECMA

Si occupano di reti locali i seguenti organi dell'ECMA all'interno del TC 24 (Comitato Tecnico per i protocolli di comunicazione):

- TGE protocolli di trasporto
- TGLN local area network
- TGNS network standard

7.2.1 TGLN

Il Technical Group sulle reti locali si è occupato, in particolare, dei seguenti argomenti:

- CSMA/CD a 10 Mb/s
- Token Bus
- Token Ring
- PABX ed accesso alla futura ISDN
- reti locali low speed - low cost

Seguendo una strada diversa da quella percorsa dall'IEEE 802, il TGLN ha subito pubblicato uno standard che codifica i protocolli fino al livello 4 di trasporto per una rete CSMA/CD a 10 Mb/s, sfruttando parte del lavoro compiuto in IEEE. Essendo l'ECMA un'associazione di costruttori, era importante uscire rapidamente con uno standard perchè i suoi associati non perdessero opportunità di mercato e perchè le reti locali dei diversi costruttori rimanessero interconnettibili. Il rapporto tecnico pubblicato nel mese di marzo 1982 riguarda il sistema di connessione al cavo coassiale, del livello fisico, del livello Data Link.

Il livello Network è stato nullificato, non considerando il caso di più reti locali interconnesse, e per il Transport si è scelto lo standard ECMA 72 classe 4. Un aspetto non considerato è pure il cavo si connessione transceiver controller, rimandato ad una futura edizione dello standard.

7.3 ANSI

L'ANSI, l'Istituto nazionale di standardizzazione americano, ha stabilito norme per reti locali ad alte prestazioni, del genere di quelle che vengono usate tra grandi mainframes e dischi per costruire reti di backend e condividere grandi masse di dati tra calcolatori di grandi prestazioni. Lo standard denominato X.395, è ispirato alla tecnologia introdotta con reti consimili dalla CDC e da Hyperchannel della Network Scientific Corporation. Il data rate considerato è di 50 Mb/s e la topologia impiegata è quella a bus.

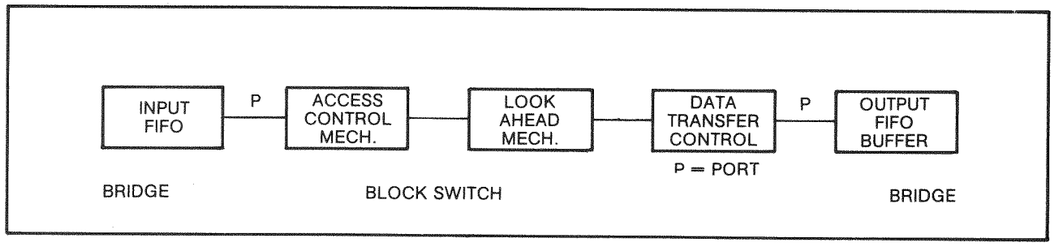


FIG. 3-1 Struttura di collegamento Block Switch/Bridges

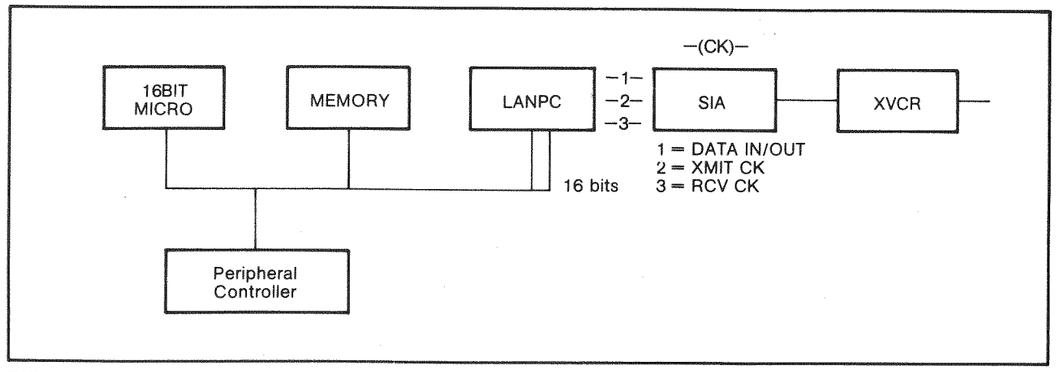


FIG. 2.1

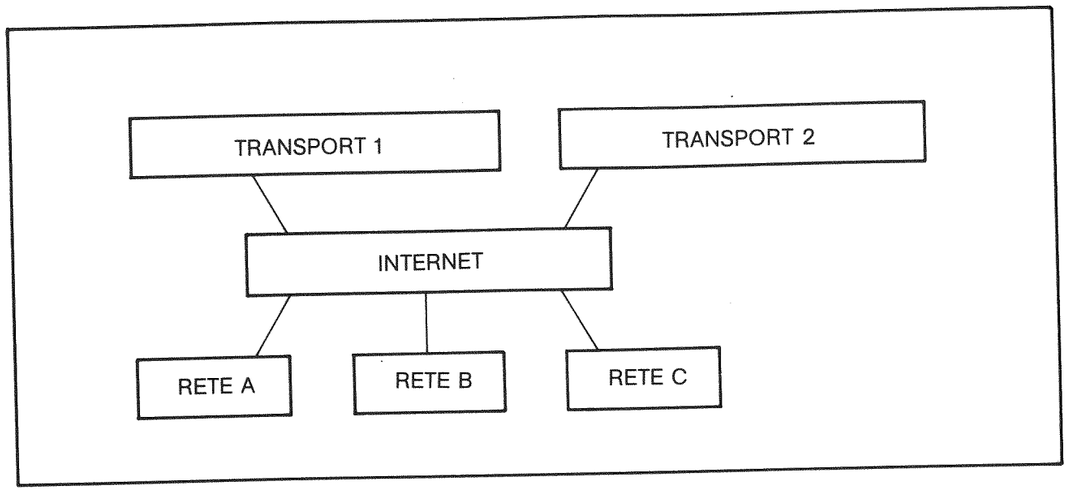


FIG. 5.1 INTERNET SUBLAYER

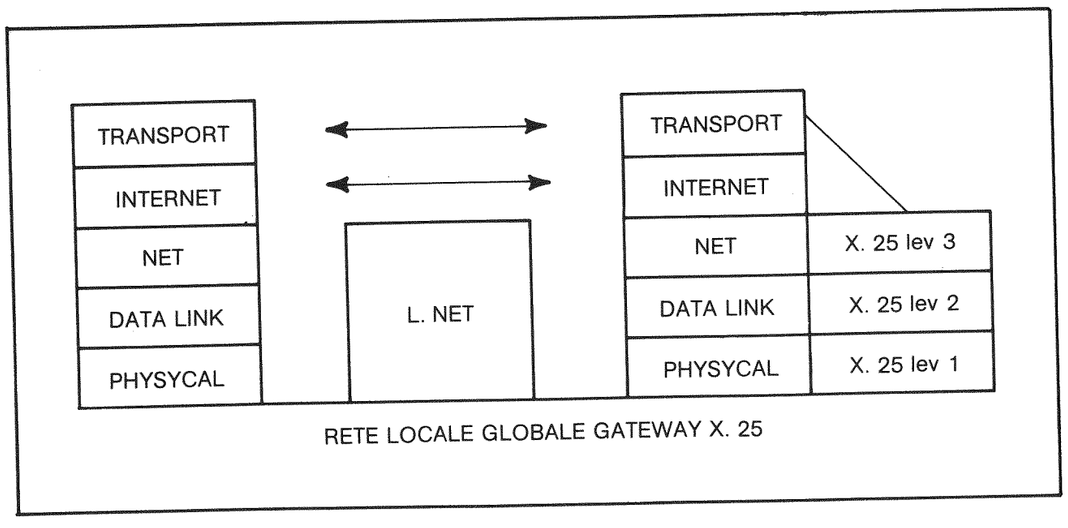


FIG. 5.2 POSSIBILE GATEWAY X. 25 .