1

Capitolo B02
procedure, algoritmi e prime elaborazioni

Contenuti delle sezioni

a. insiemi-E, procedure e algoritmi p. 2
b. istruzioni per un esecutore e per le sue esecuzioni p. 9
c. prime elaborazioni p. 17

32 pagine

B02 0.01 All'inizio di questo capitolo viene ripresa la nozione di procedura al fine di utilizzaarla per dare una prima definizione della nozione di algoritmo.

Si procede quindi a precisare in modo piuttosto dettagliato un automatismo per la elaborazione di caratteri e stringhe che utilizziamo per consolidare la nozione di procedura in modo concreto e convincere la possibilità di automatizzare le attività di risolvere problemi esprimibili mediante simboli, ossia a un livello formale.

Si procede poi all'esame dettagliato di una buona varietà di elaborazioni su stringhe formulabili in modi molto chiari.
Questi esempi di elaborazioni intendono convincere della necessità di studiare le proprietà generali delle stringhe e dei linguaggi formali e a introdurre nei successivi capitoli gli interi naturali, le liste, gli insiemi finiti e le relazioni decidibili.

Contemporaneamente emergerà la necessità di definire modalità di ampia valenza che stabiliscano come organizzare, strutturare e comunicare i risultati sulle possibilità delle elaborazioni.
Emergerà anche la necessità di definire un linguaggio, prevedibilmente molto articolto, dotato di propri termini, di propri simboli e di proprie regole formali che deve soddisfare stringenti requisiti di definitezza e di versatilità, nonché requisiti di leggibilità per la diffusione e per l'ampliamento delle competenze, requisiti di valenza didattica pensati nell'ottica del sostegno della gestibilità delle applicazioni.

Va dichiarato anche che l'approccio adottato (essendo interessato alle esigenze pratiche, consapevole delle radici empiriche del conoscere e attento delle possibilità della tecnologia) si inizia a costruire su una base finitistica e conduce senza scosse alla necessità della simbiosi tra matematica e informatica.

Alberto Marini

B02 a. insiemi-E, procedure e algoritmi

B02 a.01 Ci proponiamo ora di dare un primo chiarimento della nozione di "insieme", termine già utilizzato in modo soltanto intuitivo per indicare una collezione specifica di oggetti che vengono presentati come evidentemente individuati e distinguibili.
Si tratta di una nozione che riveste grande importanza per la matematica e per ogni argomentazione che si sforzi di essere condivisibile e utilizzabile per molti scopi e quindi priva di contraddizioni e lontana dalle ambiguità.

In effetti si incontrano moltissimi insiemi con caratteristiche diverse, ma è opportuno farli afferire ad una unica definizione; questa può essere formulata in un modo formale e rigoroso (per ora trascuriamo il fatto che in proposito si possono introdurre più definizioni formali) che consente di ottenere risultati affidabili e di portata generale (o almeno molto ampia), ma dovendo ricorrere a nozioni e tecniche della logica formale che riteniamo possano essere giudicate troppo impegnative entro i discorsi iniziali in questi primi capitoli.
Per questo approccio formale e rigoroso rinviamo a capitoli successivi, in particolare a [B60], quando si saranno incontrati vari oggetti, varie costruzioni e vari risultati che possono contribuire a convincere il lettore della utilità, della attendibilità e della organicità della matematica nel suo complesso.

Va anche segnalato che nella matematica e nelle sue applicazioni si incontra una ampia varietà di insiemi che nella trattazione formale vengono unificati, ma a costo di rinunciare alla diversità delle motivazioni che inducono a introdurre i vari insiemi specifici e alla possibilità di utilizzare le loro peculiarità per molteplici scopi effettivi.

Qui seguiremo un approccio diverso che presenta due caratteristiche rievanti: fa ricorso a vari esempi particolari ben definiti o facilmente immaginabili in genere ricavati da esperienze comuni o presentati attraverso metafore; inoltre ci preoccuperemo delle possibilità e delle difficoltà che si incontrano quando i singoli insiemi si devono rielaborare e usare nelle applicazioni.

Va segnalato che questo secondo approccio, che chiamiamo intuitivo e applicativo, rinuncia alla generalità e deve procedere attraverso varie distinzioni che richiedono parecchi chiarimenti e che fanno rischiare la frammentarietà; questo approccio tuttavia ha la possibilità di fornire vari suggerimenti a chi cerca soluzioni circoscritte ma effettive.

Occorre anche segnalare che il secondo approccio risulerà del tutto coerente con l'approccio formale: tutte le proprietà trovate per insiemi finiti, insiemi-E, insiemi-P, insiemi-B e insiemi-G trovano riscontro nelle proprietà dimostratea partire dagli assiomi degli insiemi introdotti formalmente.

B02 a.02 Per padroneggiare la nozione di insieme partiamo da esigenze comunicative molto semplici e riprendiamo le considerazioni intuitive ingenue implicite nelle prime utilizzazioni del termine per introdurre le entità che chiamiamo **insiemi-E**.

Più avanti [B08a-c] introduciamo a partire da considerazioni empiriche gli insiemi finiti che troviamo costituire particolari insiemi-E facilmente trattabili e ampiamente utilizzabili .
Definiremo poi [b08f-g] un genere molto ampio di insiemi che chiamiamo **insiemi-P** i quali possono essere insiemi infiniti, ossia insiemi chiaramente distinti dagli insiemi finiti.
Tre gli insiemi-P distingueremo gli **insiemi-B**,
meglio definiti per quanto riguarda la possibilità di controllare gli elementi che li costituiscono.
Più oltre [B18c] viene definito in termini costruttivi il genere degli insiemi-G.

Solo molto più avanti [B66], dopo aver incontrato una vasta gamma di insiemi di consistenet utilità, viene presentata una teoria formali degli insiemi che risulta sufficientemente rigorosa e viene ampiamente adottata come standard prevalnete. Accenneremo anche [B66f] a sue possibili alternative, alcune decismente astratte, altre attente alle esigenze costruttive.

B02 a.03 In molte attività risulta opportuno raggruppare fisicamente o mentalmente più oggetti che presentano caratteristiche simili o che possono servire a scopi assimilabili e che si devono far entrare in varie argomentazioni o che si devono sottoporre a più elaborazioni.
Risulta evidente che spesso sia conveniente trattare i vari oggetti di ciascuno di questi raggruppamenti con un solo nome o con un solo contrassegno il quale rappresenti il loro intero complesso in tutte le situazioni nelle quali si coinvolgono tutti gli oggetti o uno qualsiasi di essi.

Un tale raggruppamento lo chiamiamo **insieme-E** e si considera lecito assegnargli un contrassegno peculiare che consente di identificarlo concisamente come suo conveniente riferimento; lo chiamiamo **identificatore dell'insieme-E**. Osserviamo esplicitamente che la peculiarità di un identificatore di un insieme S consiste nel fatto che tale simbolo non può identificare alcun insieme diverso da S.
Ad esempio consideriamo un insieme-E contrassegnato da S, lettera che richiama l'inglese "set', termine che useremo spesso per richiamare un insieme generico, ossia un insieme per il quale evitiamo di precisare la consistenza.
Gli oggetti che sono raggruppati nell'insieme-E identificato da S si dicono ricoprire il ruolo degli **elementi dell'insieme** S e si dice che ciascuno di essi appartiene all'insieme S, oppure che S contiene tutti e soli gli oggetti che costituiscono i suoi elementi.

Osserviamo che la ampia possibilità di individuare simboli per gli sviluppi formali garantisce la possibilità di trovare uno o più.

Può accadere che gli elementi di un insieme-E S che viene preso in considerazione in una argomentazione siano individuati precisamente (magari che siano elencati uno per uno(nel caso S sia un insieme finito); altri insiemi-E sono caratterizzati da alcune propriet; più o meno soddisfacentemente definite e godute da tutti i loro elementi e solo da essi.
In considerazioni più generiche si considerano insiemi S lasciati indefiniti e i loro elementi sono solo dichiarati come appartenenti ad S senza tentare di caratterizzarli con altre richieste.

Quando risulta opportuno fare riferimento a un elemento di un tale S lasciando indefinita la sua individualità occorre comunque associargli un segno identificativo che qui scriviamo x (come accade spesso); si parla allora di **elemento generico** di S; si usa dire anche che x è una **variabile** che può assumere come **valore** ciascuno dei vari elementi di S; oppure si dice che x è una variabile che si può scegliere arbitrariamente in s.

B02 a.04 L'introduzione di un insieme-E S può essere supportata a livello intuitivo da metafore come quella del contenitore materiale nel quale sono collocati fisicamente tutti i suoi elementi x o quella del collegatore che determina il raggruppamento di persone, cose materiali o documenti con caratteristiche simili a costituire un catalogo di prodotti, una squadra, una categoria sociale, una specie biologica, una raccolta bibliografica, una documentazione con uno scopo operativo, ...

La specificazione "-E" del termine vuole segnalare che gli insiemi-E sono stati introdotti dopo aver osservati, oppure caratterizzati, oppure pensati, oppure prospettati i rispettivi elementi o anche solo alcuni di essi considerati in qualche modo esemplari.

Va osservato che si è introdotto come oggetto di riferimento un insieme-E senza esserci preoccupati di raggiungere una qualche consapevolezza delle precise caratteristiche degli oggetti che si vogliono

dichiarare suoi elementi: questo si giustifica con il carattere provvisorio di alcuni insiemi-E che possono essere presi in considerazione solo in relazione a una poco precisata prospettiva di sviluppo conoscitivo o applicativo.

Per un insieme-E escludiamo che possa coincidere con uno dei propri elementi; come vedremo questa richiesta evita conseguenze che porterebbero a conclusioni contraddiorie che sono totalmente da evitare. Ammettiamo invece la possibilità di assegnare a un insieme-E il ruo di elemento di un altro insieme-E; con questa possibilità gli insiemi-E sono resi oggetti meglio utilizzabili per varie argomentazioni e costruzioni, ossia strumenti di una possibile versatilità.

B02 a.05 Il primo tipo di insieme-E che conviene prendere in considerazione è l'insieme dei caratteri che costituiscono un ben preciso alfabeto, oggetti che possono essere solo in numero finito. Questo insieme è pienamente giustificato dalla possibilità di presentare tangibilmente tutti i segni che lo compongono e dalla assunzione che una squadra di agenti è in grado di riconoscere tutti i suoi caratteri e di distinguere due occorrenze di suoi caratteri come appartenenti a due diversi tipi.

È possibile anche considerare un alfabeto che nel tempo si arricchisce di nuovi caratteri che consentono di soddisfare esigenze che si vanno manifestando nel corso di svolgimento di attività elaborative o conoscitive che si servono di questo alfabeto.

Un altro tipo di insieme-E che si impone naturalmente è la collezione delle stringhe su un dato alfabeto: per ciascuna stringa trattabile da una squadra di agenti questi sanno decidere se appartiene alla suddetta collezione.

Un altro insieme-E che prenderemo in considerazione riguarda i problemi ben definiti, altre entità che stiamo introducendo basandoci su aspettative.

All'opposto si possono considerare insiemi-e tutti i raggruppamenti di oggetti presentabili con elenchi finiti; tra questi tra poco incontreremo i complessi costituenti i dati di una istanza di problema risolvibile e i complessi costituienti i corrispondenti risultati.

Si osserva che abbiamo introdotto gli insiemi-E senza avanzare rilevanti richieste, ma solo prospettando entità che possono essere inserite in argomentazioni a priori imprecisate, ovvero prospettando entità alle quali si richiede solo di essere riconoscibili in argomentazioni condivisibili.
Per prenderle in considerazione basta che si possa confidare nella possibilità di riconoscere la relazione tra un tale raggruppamento e i suoi elementi.

Stante questo ridotto impegno si può presumere di imbattersi in parecchi insiemi-E e in effetti nelle considerazioni che riguardano la soluzione affidabile di problemi ben definiti se ne incontrano tanti. Questo induce spesso a chiamarli sbrigativamente "insiemi" senza ulteriori specificazioni.

In sintesi utilizzeremo il termine insieme-E ed altri termini più specifici (insiemi-B, insiemi-P, insiemi-G) solo quando serve sottolineare le rispettive peculiarità di definizione e di utilizzabilità.

A questo punto occorre introdurre un tipo di notazione di larghissimo uso.
Per segnalare che l'oggetto e appartiene a un insieme S (ch potrebbe essere considerato un insieme-E o un insieme di altro genere) usiamo la notazione concisa $e \in S$.

B02 a.06 Proseguiamo nella precisazione del termine procedura e quindi della stessa nozione di esecutore di procedure.

Per **procedura** si intende un complesso di regole o istruzioni operative atte a governare le azioni che un suo esecutore mette in atto al fine di ottenere soluzioni delle istanze di problemi che è incaricato di affrontare.

Anche la frase precedente va considerata una "definizione di prospettiva", in quanto si limita alla finalità delle procedure e lascia indefinite le caratteristiche delle istruzioni e delle azioni che queste richiedono di eseguire.

In precedenza [B01b09] si è segnalato che si propongono per il ruolo di esecutori di procedure artificiali, non umani, sia macchine formali di vari generi (anche molto astratte), che all'opposto prodotti industriali molto articolati come i computers dotati di programmi scritti in uno o più linguaggi di programmazione.

Nel seguito preciseremo come primo tipo di procedura da proporre per il modello $\mathcal{MAG}$ le cosiddette macchine sequenziali multinastro (MSM) e successivamente amplieremo la gamma delle possibili procedure ricorrendo a dimostrazioni di equivalenza tra le diverse macchine con l'intento di chiarire la vastità e la sostanziale unitarietà delle loro prestazioni.
Queste dimostrazioni si ottengono ricorrendo al meccanismo che chiamiamo **simulazione** del quale ora diamo una prima definizione piuttosto generica che riguarda solo esecutori di procedure.

Si dice che un EP C_1 può simulare un secondo EP C_2 sse si trova una trasformazione che a ogni elaborazione E_1 eseguita da C_1 che a partire da dati D porta ai risultati R associa una elaborazione E_2 di C_2 che partire dagli stessi D ottiene gli stessi risultati.

Diciamo poi gli EP C_1 e C_2 sono **esecutori equivalenti per prestazioni** sse il primo può simulare il secondo e questo è capace di simulare il primo.

In seguito svilupperemo dimostrazioni di equivalenza attraverso simulazioni che consentono di allargare progressivamente la gamma dei tipi di EP che hanno prestazioni equivalenti alle MSM.

Questo modo di procedere è evidentemente laborioso, ma riesce a condurre a una visione unitaria degli esecutori e delle procedure in grado di trovare soluzioni affidabili di estesi insiemi-E di problemi.

Inoltre questo modo di procedere corrisponde a vari passi iniziali di un progressivo sviluppo dell'*apparato*, ossia a vari ampliamenti iniziali dell'insieme dei problemi che si sanno risolvere affidabilmente e degli strumenti materiali e metodologici che portano a loro risoluzioni.

B02 a.07 Quando si definisce una procedura si dovrebbe precisare anche quali dati possono caratterizzare le istanze di problema che essa può affrontare e quali no.
Procediamo ancora con indicazioni genericamente programmatiche, ma servendoci di scenari un po' più definiti.

Consideriamo di avere un problema $\mathcal{P}$, di avere definito un tipo D di complessi di dati forniti da stringhe che ha senso proporre per caratterizzare le istanze di $\mathcal{P}$ e di disporre di una procedura $\mathfrak{A}$ adatta ad essere utilizzata per risolvere le istanze di $\mathcal{P}$.
Se d denota un dato del tipo D, ossia se $\mathsf{d} \in DSd$ denotiamo con $\mathcal{P}_\mathsf{d}$ la relativa istanza del problema $\mathcal{P}$ e con $\mathfrak{A}(\mathsf{d})$ la corrispondente elaborazione che ci si aspette da parte della procedura $\mathfrak{A}$.

Nel contesto delineato, e in particolare in dipendenza da $\mathcal{P}$, si pone il problema $\mathcal{P}_\mathsf{D}$ di stabilire per ogni dato $\mathsf{d} \in DSd$ se la elaborazione $\mathfrak{A}(\mathsf{d})$ giunge a conclusione fornendo un risultato (che potrebbe essere giudicato utile, ma anche di nessuna utilità), oppure che si rivela in grado di procedere illimitatamente senza fornire risultato, oppure che si osserva in grado effettuare un certo numero di passi che lasciano in una situazione di incertezza.

Questo nuovo problema viene detto **problema dell'arresto** della procedura $\mathfrak{A}$ rivolta alla soluzione del problema $\mathcal{P}$ relativamente al tipo di dati D.
Si tratta di un problema di carattere strategico che denoteremo con il simbolo $\mathsf{Halt}(\mathcal{P}, \mathfrak{A}, \mathsf{D})$.

B02 a.08 Accade che si conoscono molti problemi per ciascuno dei quali si sanno definire procedure che per ciascuno dei dati atto ad esprimere istanze del problema stesso sono in grado di giungere a una conclusione (ammesso che le risorse effettivamente disponibili lo permettano).
Per queste procedure, cioè per le procedure per il quale si sa risolvere il problema dell'arresto, adottiamo il termine **algoritmi**.

Si individuano invece procedure per le quali non si rivela affatto facile decidere se le elaborazioni che controllano si concludano per ogni scelta sensata dei dati iniziali o all'opposto per le loro esecuzioni a partire da alcuni dati iniziali non sappiamo garantire la conclusione anche dopo aver eseguito un cospicuo numero di passi.

Va segnalato che il problema dell'arresto è decisamente cruciale: il carattere algoritmico o meno delle procedure rimane una questione aperta e gli studi relativi hanno condotto a risultati che costituiscono dei limiti per la stessa matematica.

Evidentemente anche per definire il termine algoritmo è stata data solo una definizione di natura programmatica sulla base di un requisito definito vagamente.
Nelle prossime pagine dovremo procedere a definire singoli algoritmi ai quali si può assegnare un alto livello di attendibilità condivisibile; per questo procederemo costruttivamente attraverso la presentazione di una serie di esempi che iniziano con problemi e procedure molto elementari.

Insieme agli algoritmi presenteremo argomentazioni e proprietà metodologiche che consentono di definire le prime entità matematiche e le prime nozioni computazionali.

Conviene osservare esplicitamente che sul piano pratico gli algoritmi, in linea di massima, sono le procedure che risultano più utili.

Incontreremo però dei limiti che possono essere superati solo passando da una trattazione che riconosce solo entità e processi finiti ad una che prende in esame anche la nozione di infinito potenziale; solo allora si potrà meglio definire la distinzione tra algoritmi e procedure riconosciute come non algoritmiche.

B02 a.09 Procediamo ora, ancora con intenti programmatici e fiduciosi e in modo generico ossia senza ricorrere a tipi specifici di esecutori, a esporre con qualche dettaglio i requisiti degli algoritmi considerati fondamentali.

(a) Un algoritmo è costituito da un complesso finitamente formulato di istruzioni rivolte a un esecutore reale o convincentemente giudicato realizzabile che può avere natura umana o artificiale (**finitezza della descrizione di un algoritmo e sua eseguibilità**).

(b) Un algoritmo definisce un processo per la costruzione, a partire da alcune informazioni date, di nuove informazioni da considerare risultati. Queste informazioni possono essere di varia natura (simbolica, numerica, geometrica, fisica, ...) e si chiede solo che esse abbiano definizioni condivisibilmente riconoscibili. (**finitezza e definitezza dei dati e degli obiettivi di un algoritmo**).

(c) Il processo eseguito da ogni esecutore cui si demanda l'esecuzione dell'algoritmo si richiede che sia effettuato attraverso una sequenza di **passi** ciascuno dei quali che collega due successive **configurazioni**; queste sono caratterizzate dalle informazioni via via disponibili per l'esecutore; passi e configurazioni successive sono descrivibili come componenti di una sequenza di **fasi successive**, la quale determina una sequenza di **istanti successivi** (**discretezza di un algoritmo**).

(d) Nell'istante iniziale di una elaborazione l'esecutore dispone solo del del complesso (finito) di informazioni proprie e di informazioni che gli sono state sottoposte; in ogni passo successivo l'esecutore è in grado di ottenere nuove informazioni derivanti dalle informazioni disponibili nella fase precedente (**automatismo e capacità di memoria di un algoritmo**).

(e) Le informazioni ottenute in ciascun passo sono determinate univocamente dalle informazioni possedute nella fase precedente (**determinismo di un algoritmo**).

(f) Le regole che consentono all'esecutore di effettuare un passo esecutivo sono semplici o riducibili a regole elementari, ciascuna tale da coinvolgere un insieme circoscritto di elementi (**elementarità delle istruzioni e dei cambiamenti delle configurazioni di un algoritmo**).

(g) Il complesso delle informazioni date in partenza può essere scelto in un insieme circoscritto ben riconoscibile (**definita ampiezza della portata di un algoritmo**).

B02 a.10 Introduciamo il termine **coppia in forma canonica** per denotare in un determinato contesto $\mathcal{C}$ un tipo di stringa che segue lo schema con le caratteristiche che seguono.

Lo schema presenta cinque successive sezioni:

$$\langle \textit{ primo membro , secono membro } \rangle ,$$

dove i due membri sono stringhe che non contengono i segni $\langle$, ",", e $\rangle$.

Le sezioni vengono così caratterizzate:

$\langle$ è detto **delimitatore iniziale** della coppia canonica,
al primo membro si assegna il ruolo di *primo componente della coppia*,
il segno , è chiamato **separatore**,
il secondo membro ha il ruolo di *secondocomponente della coppia*,
$\rangle$ è detto **delimitatore finale**.

Più generalmente diciamo **coppia in forma quasicanonica** ogni stringa che segue uno schema, ancora con cinque sezioni, della forma

delimitatore iniziale primo membro separatore secondo membro delimitatore finale ,

nel quale i delimitatori e il separatore sono segni o semplici stringhe ben riconoscibili e i due membri sono stringhe alle quali si chiede di contenere caratteri riconoscibili diversi dai delimitatori e dal separatore e che devono denotare entità significative nel contesto$\mathcal{C}$ nel quale la coppia viene usata.

Esempi di coppie in forma canonica sono

$$\langle\ 2.7182818284...\ ,\ 3.142128...\ \rangle \quad , \quad \langle\ \mathtt{e}\ ,\ \pi \quad , \quad \langle a, b\rangle\ .$$

Esempi di coppie in forma semicanonica sono

[Castore / Polluce] , < Abel, Niels ; Galois, Évariste > , (1839-1914) ,

nelle quali si riconoscono facilmente i delimitatori "[" con "]" , "<" con ">" e "(" con ")", nonché i separatori "/", ";" e "-" .

Il delimitatori iniziale e finale di ciascuno dei precedenti schemi costituiscono una cosiddetta coppia di delimitatori coniugati e si potrebbero presentare con la seguente coppia in forma semicanonica

<delIni> *delimitatore iniziale* <sep> *delimitatore finale* <delFin>

nella quale i delimitatori iniziale e finale sono simboli a loro volta presentabili dalla coppia in forma canonica

$$\langle\ ¡delIni¿\ ,\ ¡deFIN¿\ \langle\ .$$

B02 a.11 È banale precisare un algoritmo in grado di riconoscere le sezioni di una forma canonica.
È anche facile, in un contesto nel quale si sono poste restrizioni ragionevolmente riconoscibili alla forma dei simboli che possono costituire i delimitatori e il separatore, precisare un algoritmo per un esecutore umano affidabile o per un esecutore artificiale condivisibilmente considerato ben definito il

quale esaminando una stringa qualsiasi stabilisce se si tratta di una coppia in forma semicanonica e in caso affermativo fornisce i suoi due membri su due nastri o su altre memorie riutilizzabili.

Risulta quindi ragionevole confidare che le coppie canoniche o semicanoniche possano costituire oggetti formali utili per il trattamento di coppie di entità per le quali si voglia enunciare un qualche tipo di collegamento associabile a qualche genere di utilizzo.

Come vedremo in :b le coppie in forma canonica e semicanonica sono casi particolari delle entità formali che chiamiamo "coppie" *tout court*; queste coppie generiche le considereremo casi particolari delle importanti entità formali che chiameremo "liste".

Si sottolinea che coppie e liste vengono definite confidando convintamente che possono essere facilmente riconosciuti ed elaborati da algoritmi; per rendere la loro definizione più apprezzabile aggiungiamo che si riesce a constatare che coppie e liste risultano utili per molte argomentazioni e per molte applicazioni (ad esempio nella geometria analitica del piano [B21, B31, B43]).

B02 a.12 Ad ogni algoritmo $\mathfrak{A}$ è possibile associare effettivamente coppie di informazioni della forma

$$\langle\ \textit{dato}\ ,\ \textit{risultato}\ \rangle\ ,$$

dove *dato* intende rappresentare un complesso di informazioni in grado di innescare una esecuzione di $\mathfrak{A}$ e *risultato* sta per il complesso di informazioni che l'algoritmo è sicuramente in grado di fornire alla conclusione della suddetta esecuzione (che assumiamopossa essere effettuata in un numero finito di passi).

Le coppie *dato-risultato* costituiscono collegamenti tra informazioni che contribuiscono alle conoscenze sugli algoritmi disponibili e che in molti casi risultano molto utili allo sviluppo dell'*apparato*.
Vedremo che il complesso di queste coppie è strettamente collegato a un insieme di entità importanti per la matematica e le sue applicazioni che chiamiamo "funzioni costruibili", in quanto la disponibilità di algoritmi in grado di produrle rende possibili molti estesi ampliamenti del complesso degli enunciati che l'*apparato* rende disponibili; questi enunciati li chiameremo anche **fatti matematici**.

Conveniamo che la funzione costruibile associata all'algoritmo $\mathfrak{A}$ venga identificata con la notazione $\mathfrak{Fun}_{\mathfrak{A}}$.
Se d denota un dato sottoponibile all'algoritmo $\mathfrak{A}$, con la scrittura $\mathfrak{Fun}_{\mathfrak{A}}(\mathsf{d})$ denotiamo il risultato fornito dalla corrispondente esecuzione; questa informazione viene detta **valore assunto da una funzione** $\mathfrak{F}$ in corrispondenza del particolare dato d.

Va segnalato che le funzioni costruibili discrete ora individuate sono casi particolari di un genere di entità matematica chiamate funzioni *tout court*, che si incontrano in moltissime argomentazioni matematiche, scientifiche e tecniche.
Le funzioni sono casi particolari di un'altro genere di entità di grande importanza e utilità, le "relazioni" che in una prima versione introduciamo in :b.

Evidentemente le caratteristiche dei dati ammissibili e dei risultati qualificano marcatamente le funzioni costruibili e nel seguito, quando saranno meglio definite le nozioni di insieme e di funzione, preciseremo meglio queste caratteristiche per i diversi tipi di funzioni alle quali attribuiremo una qualifica che chiameremo "genere".

B02 b. istruzioni per un esecutore e per le sue esecuzioni

B02 b.01 Proseguiamo a esaminare i comportamenti degli esecutori, umani o artificiali, che si dedicano esclusivamente ad algoritmi che riguardano solo stringhe, assicurando il pieno rispetto dei requisiti esposti in a04.

Mentre per le elaborazioni descritte nella prossima sezione presenteremo vari dettagli per l'utilizzo dalle MSM dotate di nastri, più avanti presentiamo descrizioni che cercano di essere del tutto precise ma sono meno rigidamente dettagliate, dando per scontata l'equivalenza in linea di principio tra esecutori umani e artificiali e facendo riferimento ad esecutori artificiali che costituiscono generalizzazioni più versatili delle MSM e chiameremo MSP, macchine sequenziali programmabili [B17a01] alle quali essenzialmente chiediamo solo di essere effettivamente costruibili e attendibili.

Più oltre daremo per scontata l'equivalenza delle prestazioni degli strumenti procedurali (prima di approfondirla in C21 e parleremo genericamente di macchine, di istruzioni e di programmi.

Conviene inoltre segnalare che quando si rende necessario dare indicazioni più effettive è opportuno prospettare o precisare elaborazioni eseguite dagli odierni computers e controllate da un linguaggio procedurale o da un sistema software più user friendly che in genere presupponiamo conosciuto a grandi linee dal lettore.

Dobbiamo inoltre osservare che la restrizione riguardante algoritmi ed esecuzioni che agiscono solo su stringhe mon va giudicata drammaticamente riduttiva.
Infatti, come avremo modo di esemplificare, attraverso stringhe si possono rappresentare tutti gli oggetti e i procedimenti che possono essere invocati nelle attività che consentiamo dedicate all'ottenimento delle SAP e alla stessa definizione dei problemi razionalizzati [B01a04].

B02 b.02 Riprendiamo a descrivere le elaborazioni di un esecutore che denotiamo con **C** e che per procedere con gradualità assegnamo al tipo MSM.

All'inizio **C** dispone del complesso delle istruzioni che consentono di affrontare un problema $\mathcal{P}$ di sua competenza e del dato (più o meno composito) d che caratterizza una istanza di $\mathcal{P}$.

I nastri di lavoro di **C** che stanno per essere utilizzati e i nastri di uscita per i risultati sono vuoti, ovvero contengono solo istanze del demarcatore di inizio nastro che scegliamo essere $\vdash$ e una sequenza di caselle contenenti il segno insignificante che chiamiamo "null".

Come si è anticipato per semplicità ipotizziamo che le caselle disponibili siano in grado di contenere tutte le informazioni necessarie per ottenere le soluzioni di tutte le istanze dl problema che delle quali **C** dispone attraverso sequenze di un numero finito di passi.

Per descrivere talune elaborazioni è conveniente supporre che prima delle caselle disponibili per i dati intermedi e finali sia registrato un segno di fine nastro che scegliamo essere $\dashv$ e supporre che questo segno dopo ogni estensione delle caselle utilizzate possa essere ricollocato. a destra dell'ultima utile.

Del segno $\dashv$ si potrebbe anche fare a meno e ci si rende conto senza difficoltà che questi due modi di procedere si dimostrano equivalenti.
Comunque supponiamo che i dettagli della gestione della parte finale dei nastri possano essere facilmente individuati, ad esempio servendoco del null e non ci dilungheremo sui relativi dettagli.

Osserviamo tuttavia che l'equivalenza tra operare con $\dashv$ o senza fa parte di un genere di situazioni che si riscontrano spesso.

Tutte le procedure e tutti gli algoritmi possono presentare numerose varianti sostanzialmente equivalenti, almeno quando si trascurino le questioni dell'efficienza e quando si possano trascurare alcune istanze "di nicchia" e i relativi risultati.

Molte delle varianti possibili presentano differenze che riguardano solo dettagli delle manovre sulle quali non è opportuno soffermarsi per evitare di addentrarsi in considerazioni inutilmente minuziose, confidando sulla precisabilità dei dettagli ignorati e delle equivalenze tra queste varianti.

Altre varianti presentano differenze complessive e strutturali che vanno esaminate attentamente quando si studiano l'efficienza, i costi e la adattabilità del procedimento risolutivo, ma che nei nostri attuali discorsi iniziali possono interessare solo quando contribuiscono comprensibilmente all'arricchimento dei contenuti matematici.

B02 b.03 In ogni fase di una elaborazione ciascuno dei nastri in uso contiene stringhe su un ben definito alfabeto A che consideriamo parte di $\mathbb{A}W(\mathsf{C}, \mathcal{P})$.
Ad ogni passo della elaborazione i nastri vengono modificati attraverso singole operazioni elementari richieste dalle istruzioni, in accordo con la elementarità in a06(f).

Dal punto di vista dell'organizzazione complessiva del lavoro di una squadra di esecutori, si osserva che i dati iniziali di una elaborazione di **C** possono essere sia i risultati di sue precedenti elaborazioni, che dati ricevuti da altri agenti o da committenti.

Osserviamo anche che la situazione iniziale di un nastro completamente disponibile, rappresentabile dal digramma $\vdash\ \dashv$ o dallo schema $\vdash$ *caselle con repliche del carattere null* , si può descrivere anche come nastro contenente 0 caratteri significativi.
Questo contenuto può essere rappresentato convenientemente anche dalla **stringa muta** [B01d10], l'entità che denotiamo con il segno $\sqcup$.

B02 b.04 Vediamo le caratteristiche generali dei sistemi di istruzioni **I** che determinano i comportamenti di un generico esecutore **C** che ora pensiamo primariamente sia una artificiale MSM, ma riservandoci di accennare anche i corrispondenti comportamenti di un esecutore umano.

Uno schema generico delle MSM presenta al suo apice un dispositivo che chiamiamo **unit'a di controllo** (*control unit*), o più brevemente **controllo**, che ha il compito di interpretare e di far eseguire le istruzioni, una per ciascuno dei passi di ogni esecuzione.
Il controllo di **C** si può assimilare da un lato all'unità centrale di un computer e dall'altro al lavoro mentale di un esecutore umano che è in grado di effettuare elaborazioni sostanzialmente equivalenti, eventualmente consultando un libretto di istruzioni e servendosi di propri appunti.

Il controllo si serve dei nastri tramite le relative testine e di contenitori particolari di caratteri a sua diretta disposizione che chiamiamo **registri** e che nello schema conviene collocare al di sopra di testine e nastri.

Il controllo si serve a ogni passo di un registro chiamato **registro di stato** della macchina; esso in ogni fase esecutiva contiene una informazione, lo **stato dell'esecutore**, che può assumere una gamma (finita) di valori caratteristica dei sistemi di istruzioni gestibili da **C**, ma se vogliamo anche caratteristica della totalità dei problemi della problematica della quale si occupa **C**.

I diversi stati in genere vengono contrassegnati da simboli peculiari scelti opportunamente o da brevi stringhe alfanumeriche convenzionali che conviene scegliere preoccupandosi della leggibilità delle istruzioni.

Una MSM può disporre anche dei contenitori di informazioni nei nastri di I/O che chiameremo **words** costituiti da una o poche caselle (nei comuni computers tipicamente si hanno words che portano 2, 4,

8 o 16 caratteri del genere chiamato byte) il cui contenuto in genere ha forma e ruolo ben definiti dal punto di vista delle elaborazioni eseguibili da **C**.
Queste words differiscono dai registri centrali per essere di uso meno agevole (attraverso un indirizzo che le contrassegna), ma possono essere molto più numerosi, ad esempio 2^30 o 2^40.

In ogni fase esecutiva **C** si trova in un preciso stato; nelle descrizioni delle elaborazioni può essere efficace raffigurare gli stati come nodi di un digrafo [D27] e illustrare una elaborazione parlando di spostamenti o salti del controllo su questi diversi nodi [e01].

Molte elaborazioni compiute da un esecutore sono presentate efficacemente caratterizzando i successivi stati via via "visitati" dal controllo in conseguenza della manovra con cui avvia l'esecuzione (determinismo degli algoritmi [a05(e)] e in relazione con l'esigenza che deve soddisfare.

B02 b.05 Mentre le istruzioni per un esecutore umano possono essere presentate in termini discorsivi, ogni programma per una MSM è costituito da un complesso di enunciati imperativi, le istruzioni, espressi da stringhe di forme ben precise che conviene pensare registrate su un apposito nastro istruzioni a sola lettura nel quale, come chiariremo, si può individuare l'accennato digrafo degli stati visitabili.

Ogni istruzione va assegnata a un particolare **tipo di istruzione** e tutte le istruzioni di un dato tipo hanno una forma espressa da uno schema peculiare del tipo.

L'elenco delle istruzioni dei vari tipi che possono essere eseguite da una macchina MPM e da ogni automatismo viene chiamato il suo **repertorio delle istruzioni**.

Il repertorio delle istruzioni eseguibili da un esecutore lo caratterizza totalmente, in quanto nel caso dell'esecutore artificiale consente di individuare i dispositivi per le registrazioni e per le trasformazioni dei quali dispone, mentre per un esecutore umano si trova in stretta relazione con le sue competenze operative.

B02 b.06 Entriamo nel merito delle istruzioni di un esecutore **C** del genere MSM, cioè con le azioni che esso può compiere.

Ciascuna istruzione è costituita da una sequenza di campi, stringhe separate da un apposito segno con il ruolo di separatore di campi, che denotiamo a livello formale con σ_f e qui abbreviamo con ","; ogni istruzione si conclude con un segno al quale assegnamo il ruolo di terminatore di istruzione e che denotiamo con τ_i e abbreviamo con ";".

Ciascuna istruzione nei primi due campi contiene, risp., il contrassegno di uno stato che viene detto "stato inizio passo" e il segno peculiare che denota il tipo della istruzione stessa e che chiamiamo **codice del tipo di istruzione** e denotiamo con S_i.

Buona parte delle istruzioni nel terzo campo contiene un fornitore di operando che può essere il contrassegno di un nastro, il contrassegno di un registro o un esplicito carattere di lavoro; contrassegni e caratteri di lavoro vanno attribuiti all'alfabeto $\mathbb{A}W_{\mathbf{C},\cdot}$.

Molte istruzioni nel quarto campo presentano un registro con il ruolo di destinazione per il risultato che l'esecuzione di una tale istruzione deve fornire.

In gran parte delle istruzioni compare anche un campo per uno stato con il ruolo di "stato fine passo" il quale può sia coincidere che differire dallo stato inizio passo; tale stato S_f costituisce il nuovo contenuto del registro di stato e possiamo descrivere questa operazione dicendo che il controllo alla conclusione del passo attuale passa a / visita S_f.

Sono disponibili anche le cosiddette "istruzioni di scelta dicotomica" che presentano due possibili stati fine passo tra i quali ogni esecuzione del passo deve scegliere; questi compaiono in due campi che seguono quelli che determinano la scelta.

Il primo dei due possibili stati fine passo verrà visitato sse il carattere fornito dal terzo campo coincide con il carattere fornito dal quarto campo, il secondo verrà visitato se i due suddetti caratteri sono diversi.

Questo fondamentale tipo di istruzione rende possibili agli algoritmi di effettuare scelte di comportamento in dipendenza dalle informazioni che si vanno rendendo disponibili nei registri, nelle words e sui nastri.

Molti tipi di istruzioni prima del terminatore di istruzione presentano una sequenza di terne di campi ciascuna associata a un diverso nastro; aventi come scopo la scrittura di un carattere sopra tale nastro. Ciascuna di queste terne presenta, nell'ordine, il contrassegno di un nastro T, un fornitore del carattere da scrivere sulla casella corrente di T e una richiesta di eventuale spostamento della testina di T che assume uno dei tre valori -1, 0 e 1; questi implicano, risp., uno spostamento all'indietro, nessuno spostamento e uno spostamento in avanti.

B02 b.07 Tra le diverse istruzioni di una procedura, e in particolare tra le diverse istruzioni di una MSM, non se ne possono avere due che presentano lo stesso stato iniziale; questa richiesta, unita al fatto che l'esecuzione di ogni passo conduce a un solo nuovo stato, garantisce che ogni passo di una esecuzione può essere eseguito da una sola istruzione e assicura che ogni dato iniziale determini la conseguente elaborazione corrisponente a una unica sequenza di configurazioni (e in particolare a una sola sequenza di stati visitati successivamente). In altre parole risulta garantito il determinismo delle elaborazioni.

Un sistema di istruzioni che non soddisfa la precedente richiesta va considerato illegale, cioè inaccettabile come sistema atto a reggere le elaborazioni dell'esecutore.

Si osserva che questa richiesta può essere sttenuata chiedendo che non vi possono essere due istruzioni riguardanti lo stesso stato e lo stesso carattere che determina la mossa; abbiamo scelto la richiesta più drastica solo per maggiore semplicità; la scelta più permissiva consente di avere programmi che si servono di un numero sensibilmente minori di stati con evisenti risparmi costruttivi sui quali non intendiamo soffermarci.

Un ultimo tipo di istruzione, la istruzione di arresto, presenta due soli campi: il primo contiene lo stato inizio passo e il secondo contiene il codice specifico che comporta la conclusione dell'elaborazione.
Ogni repertorio di istruzioni per essere legale deve contenere almeno una istruzione di arresto.

Tutte le istruzioni descritte comportano azioni tanto elementari da convincere da poter essere implementate senza difficoltà da esecutori artificiali e da poter essere interpretate senza dilemmi e senza esitazioni da esecutori umani dotati di precisione e di meticolosità.
La semplicità delle azioni dovrebbe convincere anche che tutti gli esecutori che affrontano una istanza di problema producano le stesse conseguenze, cioè che abbiano comportamenti coerenti.

Questa convinzione serve a garantire la riproducibilità delle elaborazioni con uguale esito da parte di diversi esecutori, quale che sia la loro costituzione fisica e quali che siano gli ambienti nei quali essi sono posti in grado di operare correttamente.

B02 b.08 Precisiamo i tipi, le forme e le prestazioni delle istruzioni delle MSM servendoci delle seguenti convenzioni:

- S_i = stato inizio passo attuale;
- S_f = stato fine passo attuale;
- S_g = stato finale del passo attuale alternativo a un copresente S_f;

- `u` = operazione di caricamento di un carattere da un nastro o un registro;
- `d` = operazione di scaricamento di un carattere su nastro o registro;
- `a` = operazione di assegnazione di un carattere a un nastro;
- `m` = operazione di spostamento della testina di un dato nastro;
- `k` = operazione di confronto tra caratteri;
- `h` = operazione di arresto dell'esecuzione;
- T_1 = nastro la cui lettura determina il passo attuale o nastro primario di uscita;
- T_2 = nastro su cui scrivere o nastro secondario di uscita;
- R_i = registro fornitore di carattere da utilizzare nel passo attuale;
- m_i = spostamento della testina sul nastro T_i cui si rivolge l'istruzione; può valere -1 per spostamento all'indietro, `0` per nessuno spostamento, $+1$ per spostamento in avanti;
- c = carattere esplicito dell'alfabeto di lavoro;
- ; = abbreviazione del separatore delle istruzioni;
- , = abbreviazione del separatore dei campi di una istruzione.

B02 b.09 Presentiamo la forma e l'effetto di ciascun tipo di istruzione.

Istruzione di caricamento: S_i , `u` , T_1 , R_2 , m_1 , S_f ;
effetto: carica il carattere letto dalla casella corrente del nastro T_1 nel registro R_2, effettua lo spostamento m_1 della testina su T_1 e passa allo stato S_f.

Istruzione di scaricamento: S_i , `d` , F_1 , T_2 , m_2 , S_f ;
effetto: scarica il carattere dal fornitore F_1 (che può essere un registro o un carattere esplicito) nella casella corrente di T_2, effettua spostamento m_2 sul nastro T_2 e salta allo stato S_f.

Istruzione di spostamento: S_i , `m` , T_1, m_1 , S_f ;
effetto: esegue lo spostamento m_1 della testina del nastro T_1 e passa ad S_f.

Istruzione di assegnazione: S_i , `a` , F_1 , R_2 , S_f ;
effetto: scrive il carattere dal fornitore F_1 (registro o carattere esplicito) nel registro R_2 e passa allo stato S_f.

Istruzione di confronto: S_i , `k` , F_1 , F_2, S_f , S_g ;
effetto: se il carattere dal fornitore F_1 e quello dal fornitore F_2 coincidono manda allo stato S_f, se sono diversi il controllo visita S_g.

Istruzione di arresto: S_i , `h`
effetto: si conclude l'esecuzione.

Si chiede che gli identificatori delle istruzioni, dei registri, dei nastri e degli spostamenti siano caratteri diversi e ben distinguibili; essi costituiscono l'alfabeto $\mathbb{A}I_{\mathsf{C},\mathcal{P}}$ e per semplicità chiediamo sono diversi dai caratteri degli altri alfabeti $\mathbb{A}S$, $\mathbb{A}D_{\mathsf{C},\mathcal{P}}$ e $\mathbb{A}M$ introdotti in B01e05.

Va segnalato che per i problemi piuttosto elaborati l'alfabeto $\mathbb{A}II,\mathcal{P}$ deve contenere un elevato numero di caratteri.

B02 b.10 Della MSM definita in precedenza si possono introdurre facilmente molte varianti equivalenti che posseggono dispositivi per istruzioni con prestazioni più elaborate delle precedenti.

Questi strumenti più prestanti presentano il vantaggio di consentire formulazioni più concise e significative di quelle richieste dalle elaborazioni della MSM assunta come modello base, ma per queste pagine introduttive risultano inessenziali e dispersive.

Una prima variante che identifichiamo con MSMa che si ottiene mpliando il repertorio di una MSM con istruzioni che consentono di usare le caselle dei nastri e le words come se fossero registri.
Si possono per esempio avere istruzioni della forma
S_i , a , T_1 , m_1, T_2 , m_2 , S_f ;
che trasferiscono il contenuto di una casella di T_1 in una casella di T_2 e possono spostare le testine dei due nastri.
Si possono anche avere istruzioni di confronto che riguardano contenuti dei nastri come quella della forma
S_i , k , T_1 , m_1, T_2, m_2, S_f , S_g che confronta le caselle di due nastri ed eventualmente ne sposta le testine.
Inoltre si possono avere istruzioni simili alle precedenti nelle quali compare un registro o un indirizzo di word invece di una coppia (nastro,spostamento)

È comprensibile come ciascuna delle precedenti istruzioni possa essere simulata dalle istruzioni consentite inizialmente che possiamo dichiarare basilari.

Una variante maggiore che chiamiamo MSMb può disporre di istruzioni registrate sopra un supporto di memoria bidimensionale che possa essere consultato da una testina che si può muovere sia verticalmente per scegliere tra le diverse istruzioni quella corrispondente allo stato attuale, sia orizzontalmente per prendere visione dei diversi campi di una istruzione.
Questa possibilità consente di immaginare e descrivere più agevolmente lo svolgimento di elaborazioni che hanno a che fare con raffigurazioni piane.

Si possono proporre memorie bidimensionali anche per i dati da elaborare, per i dati intermedi e per i risultati. I dispositivi di memoria bidimensionali sono assimilabili ai tamburi magnetici, ai pannelli di nuclei di ferrite, e alle pile di dischi magnetici e ottici utilizzati ampiamente fino a quando si sono rese disponibili le memorie circuitali o altre tecnicamente più avanzate.

La precisazione della possibilità di simulare le operazioni con queste memorie con operazioni su nastri e quindi l'equivalenza delle memorie sequenziali con le bidimensionali e le tridimensionali potrà essere esposta chiaramente ricorrendo alle cosiddette manovre di iterazione che vedremo in c05.
La loro utilizzazione da parte di esecutori artificiali risulta molto più agevolmente descrivibile e le elaborazioni da parte di esecutori artificiali in grado di eseguirle rapidamente possono risultare molto più efficienti.
In particolare esse risultano vantaggiose per l'ampia gamma delle elaborazioni grafiche.
Inoltre per gli esecutori umani risulta naturale servirsi di informazioni registrate o schematizzate su due dimensioni (fogli di carta, lavagne, poster, schermate, ...) talora anche di informazioni schematizzate su tre dimensioni (schedari, modellini plastici, ...).

B02 b.11 Ogni elaborazione viene rappresentata fedelmente (ed esaurientemente) da una sequenza di configurazioni esprimibile con una scrittura della forma

$$\langle \mathcal{C}_0, \mathcal{C}_1, \mathcal{C}_2, ..., \mathcal{C}_{n-1}, \mathcal{C}_n \rangle \ .$$

La possibilità di riferirsi a una tale notazione suggerisce di chiamare passo i-esimo il passaggio da una configurazione $\mathcal{C}_{i-1}$ alla successiva $\mathcal{C}_i$.

Ogni passo di una elaborazione vede cambiamenti molto contenuti della configurazione attribuibili alla sola istruzione che può essere eseguita nello stato attuale.
Questo si accorda con i requisiti del determinismo e della elementarità degli algoritmi, qualità che favoriscono l'ampia accettazione della affidabilità dei risultati ottenibili con le elaborazioni che stiamo prospettando.

B02 b.12 Combinando opportunamente le istruzioni dei tipi sopra presentati si possono definire moltissimi programmi che portano a risultati sensati. Per contro una aggregazione poco attenta di tali istruzioni, anche se rispetta la regola per le istruzioni dell'unicità degli stati inizio passo, presenta elevata possibilità di non costituire un programma sensato, cioè volto alla ricerca di soluzioni di un problema significativo.

La garanzia che un programma sia del tutto accettabile non è facile da dimostrare ma solleva essa stessa problemi nell'area matematico-informatici, spesso impegnativi e talora origini di interi filoni di ricerca.

Si possono facilmente redigere programmi che in relazione a certi dati iniziali avviano elaborazioni che conducono a configurazioni nelle quali non si trova alcuna istruzione applicabile.

In questi casi si decide che l'elaborazione non porti alcun risultato utile e che il programma non sia accettabile ma vada modificato, anche solo per individuare ed evitare le suddette situazioni di impossibile avanzamento.
Una tale modifica in genere si può effettuare con l'inserimento di opportune istruzioni di controllo che possono portare a una istruzione di arresto in uno stato finale che verrà interpretato come conclusione con esito negativo.

Può anche accadere di incontrare un programma che con certi dati porta verso situazioni da giudicare patologiche nelle quali vengono effettuate manovre che dopo un numero finito (in genere elevato) di passi non ha condotto ad una conclusione.Se l'esecutore constata di trovarsi in una tale situazione, si pone il problema di prevedere cosa possa succedere se si proseguisse l'esecuzione.

Potrebbe accadere che dopo ulteriori passi si giunga a un risultato utile, ma potrebbe invece accadere di giungere a una configurazione identica ad una precedente e quindi stabilire che si è di fronte a una sequenza di configurazioni che si ripetono ciclicamente senza portare ad alcun risultato utile.
Si potrebbe inoltre giungere a una situazione più oscura caratterizzata dal dubbio se sia possibile giungere a una successiva conclusione, oppure se si vada incontro a inutili sequenze di configurazioni che non sembrano ripetersi ciclicamente ma che potrebbero essere del tutto inconcludenti.

Queste situazioni critiche richiedono di essere analizzate per cercare di giungere a un loro chiarimento o di evitarle.

B02 b.13 Si scorgono dunque questioni delicate sulla correttezza, sulla sensatezza e sulla adeguatezza dei programmi e sulla possibilità di arresto delle elaborazioni, questioni che aui ci limitiamo a segnalare come questioni che richiedono attenzione.

Nelle prossime pagine ci proponiamo di presentare una serie di programmi molto semplici per i quali risulta chiaro che ogni sistema di dati iniziali conduce a conclusioni sensate dopo un numero finito di passi e che quindi corrispondono ad algoritmi per le MSM dotati di interesse.

Come si è detto, per avere algoritmi che governino elaborazioni di interesse pratico si chiede che siano finiti, non solo l'alfabeto di lavoro, il sistema delle istruzioni e quindi gli stati dell'esecutore, ma anche il numero delle words e delle caselle occupate in ciascuno dei nastri utilizzati nel corso di tutte le elaborazioni che si ritengono sensatamante eseguibili.

Possiamo tuttavia prospettare che nel corso di una elaborazione, in base alle esigenze che si vanno rivelando, sia possibile aggiungere ad nuove words e ad alcuni nastri nuove caselle in coda a quelle previste all'inizio della elaborazione quando di era data una prima valutazione della memoria necessaria rivelatasi insufficiente.

Le possibilità di facile collegabilità delle odierne apparecchiature digitali rendono ragionevole questa prospettiva di adeguamento progressivo degli automatismi.

B02 c. prime elaborazioni

B02 c.01 Procediamo ora a presentare le prime elaborazioni che una MSM **C** deve essere in grado di eseguire.
Nel fare questo ci preoccupiamo innanzi tutto di far giudicare le prestazioni delle MSM solidamente affidabili e riproducibili e di evidenziare quanto i loro comportamenti si avvicinino a quello di realistici esecutori umani, in modo da convincere sulla compatibilità delle prestazioni dei primi automatismi presentati con tutti i procedimenti che hanno consentito di risolvere con successo i problemi che si sono dovuti affrontare nel passato.

Per ciascuno dei problemi presentati in questa sezione si intende convincere che possa essere risolto con una serie adeguatamente organizzata di trasformazioni molto semplici di stringhe che dalle stringhe esprimenti una istanza di problema portano alle stringhe esprimenti dei risultati.

Per questo le elaborazioni vengono presentate trascurando tutte le complicazioni che si possono incontrano di fronte ai problemi reali e considerando solo gli aspetti essenziali dei problemi stessi.

Cominciamo dagli alfabeti in gioco ricordando le considerazioni e i simboli definiti in e04.

Per l'alfabeto dei caratteri che consentono di identificare i dati iniziali e i risultati finali delle elaborazioni $\mathbb{A}D\mathbf{C},\mathcal{P}$ ci serviamo solo delle lettere, delle cifre decimali e del punto, mentre per le scansioni ci serviamo dei segni "," e ";". quindi ci serviamo solo di una parte dell'usuale alfabeto ASCII-7 [B70c01].

Per il resto dell'alfabeto di lavoro $\mathbb{A}W_{\mathbf{C},\mathcal{P}}$, cioè per esprimere gli stati, le istruzioni e le informazioni intermedie, e per esprimere i comportamenti della macchina a partire dai suoi stati servono altri segni.
Per questi conviene mantenere la biunivocità tra segni e significati che pensiamo riguardare i soli ruoli; la biunivocità sarebbe evitabile e consentirebbe di ridurre gli alfabeti, ma richiederebbe di preoccuparsi della dipendenza dal contesto, complicazione che qui vogliamo senz'altro evitare.
Pensiamo che anche in questa prospettiva bastino poche centinaia di altri caratteri di lavoro: alcuni di questi si possono trovare nell'alfabeto ASCII-8, per altri si deve ricorrere a segni in Unicode o a brevi stringhe definite nei linguaggi della comunicazione come TEX e HTML. Ad esempio ci serviremo di semplici costrutti TEX come "S_i", "q_1", "R_j", "F_k" e "T_2".
Possiamo comunque pensare che gli alfabeti richiesti siano facilmente controllabili anche da un esecutore umano.

Quando in seguito procederemo a definire il linguaggio delle espressioni matematiche e faremo ricorso a elaborazioni controllate da un linguaggio di programmazione [B70, B71] sarà praticamente inevitabile ricorrere a entità rappresentate da identificatori costituiti da brevi stringhe che da definire con precisione e cercando di renderle significative per il lettore.
L'adozione di questi identificatori non si trova in conflitto con l'uso dei semplici monogrammi di queste eleborazioni iniziali, in quanto le espressioni che si servono dei suddetti identificatori si possono ricondurre ad espressioni contenenti singoli caratteri ricorrendo a ricorrendo a ponderate sostituzioni che tratteremo in B04a02.

Occorre tuttavia segnalare che il controllo delle collezioni di questi identificatori, migliaia, è più impegnativo e risulta importante gestire queste collezioni con elenchi [X12] gestiti da programmi appositi.

Per perseguire la semplicità preannunciata supponiamo che la nostra MSM **C**, oltre al controllo centrale con compiti di supervisione, disponga di pochi registri (primo dei quali il registro di stato) in grado di contenere singoli caratteri (o brevi stringhe essenzialmente equivalenti) e disponga di pochi nastri

la cui estensione risulti sufficiente in quanto si pensa si debbano trattare solo sequenze di caratteri di lunghezze contenute o prevedibili.

Tra i nastri si trova un nastro preregistrato e non modificabile contenente le istruzioni del programma che consente di affrontare il problema corrente; si prevedono inoltre pochi nastri di ingresso a sola lettura, pochi nastri di uscita a sola scrittura e pochi nastri di lavoro che invece possono essere letti e modificati.

Supponiamo inoltre che tutti i dati iniziali presentati alla macchina **C** esprimano correttamente le istanze del problema che essa è chiamata ad affrontare, ovvero trascuriamo la possibilità che le vengano forniti dati con possibili errori: questa supposizione è giustificabile con la possibilità di diagnosticare con procedure apposite la correttezza dei dati e anche la possibilità di apportarvi correzioni.

B02 c.02 Iniziamo con l'elaborazione volta a risolvere il dilemma elementare se due occorrenze di carattere presentate su due nastri o registri monocarattere T_1 e T_2 sono dello stesso tipo o meno.

Nello stato iniziale che, come spesso in seguito, denotiamo con q_1, si carica nel registro R_1 il carattere in T_1 e si passa allo stato q_2.
In q_2 si carica nel registro R_2 il carattere in T_2 e si passa in q_3.
In q_3 si confrontano i contenuti di R_1 e R_2: se coincidono si passa allo stato f_1, se no si va in f_0.
In f_0 si emette il carattere 0 su T_u, il nastro monocarattere di uscita e si passa in q_z; in $f_1$si emette 1 su T_u e si passa in q_z. In questo stato viene eseguita la semplice istruzione di arresto.

Questo esempio serve solo a mostrare come si risolve un dilemma elementare attuando nel modo più semplice il tipo di manovra fondamentale chiamata costrutto selettivo o più semplicemente **selezione**.

Per quanto riguarda i due caratteri che possono essere emessi su T_u:

- 1 segnala che la domanda esprimente il problema "i due caratteri sono uguali?" ha come risposta "si", cioè risposta positiva, ovvero che l'enunciato

  su T_1 e T_2 sono registrati due caratteri uguali!

  ha il cosiddetto valore di verità "vero".
- 0 dice che la domanda *i due caratteri sono uguali?* ha come risposta "no", cioè la risposta negativa, ovvero che l'enunciato

  su T_1 e T_2 sono registrati due caratteri uguali!

  ha il valore di verità "falso", il valore di verità opposto del precedente.

B02 c.03 Un problema un po' più complesso, chiede, dati sulle prime tre caselle del nastro T_i tre occorrenze di carattere, quanti tipi di caratteri forniscono, attendendosi la risposta 3 se si hanno caratteri tutti diversi, 2 se due soli coincidono, 1 se tutti i tre caratteri coincidono.

Nello stato iniziale q_1 si legge nel registro R_1 il primo carattere utile su T_i, si fa avanzare la testina sulla seconda casella e si va in q_2.
In q_2 si copia nel registro R_2 il carattere nella seconda casella di T_i, si fa avanzare la testina sulla terza casella e si passa in q_3.
In q_3 si copia nel registro R_3 il carattere nella terza casella di T_i, si fa avanzare la testina sulla terza casella e si passa in q_4.
In questo stato si confrontano i contenuti di R_1 ed R_2 e si attua la prima selezione: se coincidono si va in q_{10}, se no in q_{20}.
In q_{10} si confrontano i contenuti di R_1 e di R_3: se coincidono si passa a f_1, se no in f_2.

Anche in q_{20} si confrontano il contenuti di R_1 ed R_3: se coincidono si passa in f_2, in caso contrario si passa in q_{21}.
In questo stato si confronta il contenuto di R_3 con quello di R_2 e se coincidono si va in f_2, mentre se sono diversi si va in f_3.
I tre stati f_i con i uguale a 1, 2 o 3 comportano la scrittura su T_u, risp., di 1, 2 o 3 e quindi il passaggio allo stato q_z nel quale si ha l'arresto.

Conviene sottolineare che abbiamo introdotto i due valori di verità solo in quanto risultati di una operazione che possono essere utilizzati per scegliere tra le manovre eseguibili successivamente.

Il precedente algoritmo, rispetto al primo, prevede non una ma quattro selezioni e per organizzarle adeguatamente deve servirsi di sette stati.

Possono avere interesse anche i problemi analoghi più impegnativi che chiedono il numero dei tipi di caratteri presenti in stringhe con quattro, cinque o più componenti.

Ancor più impegnativa, ma ragionevolmente motivabile, è la richiesta di un algoritmo di portata più ampia che fornisca il numero dei tipi di caratteri presenti in una stringa con un numero a priori indeterminato di componenti.
Per un tale algoritmo occorre organizzare un nuovo tipo di manovra, l'iterazione, da considerare fondamentale in quanto richiesta da tutti i problemi minimamente articolati; conviene comunque introdurlo per problemi più semplici di quello sopra segnalato.

B02 c.04 Esaminiamo l'elaborazione che prende in carico una stringa $v := a_1\, a_2 \ldots a_n$ data su un nastro T_d di ingresso, preceduta dal carattere $\vdash$ di inizio nastro e seguita dal carattere $\dashv$ di fine nastro, per copiarla su un secondo nastro T_r racchiusa tra gli stessi delimitatori.

Ricordiamo che i due caratteri $\vdash$ e $\dashv$, fanno parte di $\mathbb{A}S$ e sono diversi da tutti i caratteri costituenti i dati e usati nelle elaborazioni appartenenti ad $\mathbb{A}W_{\mathsf{C},\mathcal{P}}$.

Osserviamo la precedente scrittura $v := a_1\, a_2 \ldots a_n$, introdotta come elemento delle considerazioni per fini generali.
In essa i caratteri a_i appartengono a $\mathbb{A}W_{\mathsf{C},\mathcal{P}}$, mentre v e il segno := (che consideriamo semplice) li consideriamo far parte di $\mathbb{A}M$, in quanto := serve a introdurre un nuovo segno, v, perché faccia da identificatore monogramma della stringa alla sua destra.
La precedente scrittura in effetti è un esempio di definizione di un simbolo semplice con il quale risulta opportuno identificare una espressione nota (la $a_1\, a_2 \ldots a_n$).

All'inizio della elaborazione C si trova nello stato di inizio q_1 ed ha le testine sulle due rispettive prime caselle utili di T_d e T_r, cioè sulle caselle che seguono le due rispettive prime e uniche contenenti $\vdash$.

Il primo passo vede il caricamento del primo carattere da T_d, a_1, nel registro R, l'avanzamento della testina di una posizione e il passaggio allo stato q_2.
In q_2 si registra su T_r il carattere in R, si fa avanzare di una casella la relativa testina e si passa in q_3.
In q_3 si confronta il carattere in R con $\dashv$; se sono diversi si torna a q_1, se coincidono si va nello stato q_4.
Tornati allo stato q_1 si ripetono i passi precedenti agendo sul successivo carattere a_2 della stringa v e con i passi successive si copia a_2 come sopra.
La esecuzione dei passi sugli stati q_1, q_2 e q_3, manovra che chiameremo **stadio dell'esecuzione**, viene ripetuta fino a completare la copiatura richiesta dell'intera v, situazione che si riscontra quando si giunge a leggere e a riscrivere sul nastro T_d il carattere $\dashv$.

Si giunge dunque in q_4 solo alla fine della copiatura e in questo stato si incontra l'istruzione di arresto q_4 , h; e si giunge alla conclusivo dell'esecuzione.

B02 c.05 Si noti che lo scorrimento della stringa v, quale che sia il numero dei suoi caratteri componenti, si è ottenuto organizzando mediante selezioni elementari riguardanti l'uguaglianza tra caratteri e cambiamenti di stati , la ripetizione di uno stadio sui tre stati q_1, q_2 e q_3.
Questa manovra costituisce un esempio di un cosiddetto costrutto iterativo, detto anche più semplicemente **iterazione** .

L'iterazione suddetta si è conclusa con il ritrovamento sulla stringa di partenza del particolare carattere terminatore $\dashv$.

Incontreremo vari altri tipi di iterazioni; tutte le iterazioni utili nella pratica devono terminare con il riscontro di una situazione conclusiva, riscontro che si può effettuare in modi diversi.

Va segnalato fin da ora che attraverso opportune organizzazioni di selezioni e iterazioni, oltre che attraverso sequenze di istruzioni senza scelte, si riescono ad automatizzare moltissime manovre utili per l'ottenimento di soluzioni della quasi totalità dei problemi risolvibili attendibilmente.

Queste automatizzazioni si ottengono definendo più o meno esplicitamente algoritmi via via più elaborati e prestanti seguendo schemi costruttivi essenzialmente semplici e questo consente di convincere la massima parte degli interessati a risolvere problemi della affidabilità delle elaborazioni e quindi delle soluzioni ottenibili con macchine MSM o con macchine che si dimostrano loro equivalenti.

Si può in tal modo procedere alla definizione di strumenti e metodi che si possono ragionevolmente proporre come idonei ad affrontare con elevate possibilità di successo "tutti" (auspicabilmente) i problemi matematizzabili che possono essere proposti ai suddetti esecutori.

L'elaborazione elementare descritta nel precedente paragrafo suggerisce un'altra considerazione generale. Con lo stadio sugli stati della iterazione mediante un numero finito di istruzioni (numero che nell'esempio è unico e si può individuare facilmente prima dell'esecuzione) si è organizzata una manovra in grado di controllare una serie di situazioni che può essere molto estesa: la stringa v da replicare potrebbe essere un testo definito da milioni di caratteri e la riproduzione di una tale sequenza di segni ogni giorno viene effettuata milioni di volte, per esempio da chi trasferisce fotografie o filmati tra smartphones e pendrives.

A questo punto si può essere tentati di affermare enfaticamente che si riesce ad organizzare una "serie illimitata di situazioni".
Un tale enunciato, anche se ha il pregio di evitare la locuzione "serie infinita" che giudichiamo criticabile, può essere travisante, sostanzialmente in quanto trascura il fatto che l'ottenimento delle soluzioni effettive delle istanze di ogni problema richiede il consumo di risorse la cui disponibilità è, ovviamente, finita.
La sopra pretesa illimitatezza nel caso di elaborazioni per esigenze pratiche non può ignorare i limiti di disponibilità delle risorse.
Si dovrebbe quindi ripiegare su enunciati con elementi cautelativi come il seguente: "Con costrutti iterativi si possono affrontare serie di situazioni con estensioni molto elevate, ma il cui consumo di risorse deve essere tenutoere sotto controllo ".

B02 c.06 L'importanza delle iterazioni induce a precisare con cura alcuni termini che le riguardano.

Una iterazione $\mathcal{I}$ si realizza attraverso l'esecuzione di una sequenza di manovre parziali che chiamiamo **stadi iterativi** o brevemente "stadi".

Ogni stadio è retto da una sequenza di istruzioni da considerare il corpo dell'iterazione; le esecuzioni di due diversi stadi possono anche essere molto diverse a causa degli effetti diversi che possono avere i contenuti che nei diversi stadi si vengono a trovare in alcune posizioni delle memorie utilizzate e che chiamiamo **variabili dell'iterazione**.

Nella descrizione di una iterazione spesso si ricorre alla metafora di un controllo umano o artificiale che **corre** sui successivi stadi della iterazione stessa e che tiene conto dei valori che vengono ad assumere le variabili dell'iterazione.

Si possono organizzare iterazioni con stadi semplici o con stadi molto elaborati; in particolare si possono organizzare stadi comprendenti una o più iterazioni complete; in questi casi si parla di **iterazioni annidate** in una iterazione più comprensiva. Inoltre si possono avere iterazioni annidate in altre a loro volta annidate in altre, senza limiti a priori.

Si distinguono quindi le **iterazioni di livello 1** o primarie che non fanno parte di alcuna altra iterazione, le **iterazioni di livello 2** o secondarie facenti parte dello stadio di una iterazione di livello 1, ossia che sono annidate in una iterazione primaria, le iterazioni di livello 3 contenute in una di livello 2, ossia annidate in una iterazione secondaria e così via.
Una iterazione rispetto alla iterazione di livello immediatamente inferiore, cioè di livello immediatamente più comprensivo, può essere chiamata **sottoiterazione** di quest'ultima.

B02 c.07 Una elaborazione molto semplice, ma fondamentale, a partire da due stringhe v e w predisposte su due nastri di ingresso T_d e T_e costruisce sul nastro T_r una stringa che presenta inizialmente, sulla sua sinistra, i successivi caratteri di v e di seguito, sulla destra, i successivi caratteri di w.

Essa consiste di una prima iterazione di copiatura della v sul nastro T_r, simile alla iterazione con stadi su tre stati vista in c05, e di una successiva iterazione molto simile innescata dal ritrovamento di $\dashv$ alla fine di T_d, che effettua l'accodamento su T_r dei successivi caratteri che formano la w.

La stringa così ottenuta è chiamata **giustapposizione delle due stringhe** v e w; tale stringa talora viene chiamata "catenazione".
Per esprimerla matematicamente si introduce la notazione $v \shortmid w$; in questa compare il segno $\shortmid$ da collocare nell'alfabeto $\mathbb{A}M$ per le conoscenze generali e da usare come identificatore dell'entità chiamata **operazione di giustapposizione**.

Si tratta del primo esempio di un genere di entità importante, chiamato genere delle **operazioni binarie costruibili**.
Si dice anche che nella scrittura $v \shortmid w$ il segno $\shortmid$ ricopre il ruolo di **connettivo operativo**.

Le operazioni binarie costruibili sono entità associabili a procedure che da una coppia di entità note, nel caso della giustapposizione da una coppia di stringhe qualsiasi, ottengono una entità dello stesso genere, nel caso precedente la stringa costituente la loro giustapposizione.

Nella matematica si incontrano molte altre importanti operazioni binarie: per esempio tra breve incontreremo le operazioni aritmetiche binarie tra numeri interi (somma, sottrazione, prodotto, divisione, ...) e più avanti tratteremo le operazioni binarie tra liste, tra insiemi, tra trasformazioni, tra funzioni, tra matrici, tra enunciati e tra entità di tanti altri generi (v. ad esempio T15).

Una variante della precedente elaborazione si chiama **accodamento di una stringa** w a una stringa v disponibile su un nastro estendibile; questa manovra, se si conviene che la v caratterizzi l'intero problema e che una istanza del problema sia caratterizzata dalla w, viene considerata una cosiddetta **operazione unaria costruibile**.

Per operazione unaria in generale si intende una trasformazione di un oggetto in un oggetto della stessa specie. Per operazione unaria costruibile si intende una operazione unaria il cui risultato (nel caso suddetto la stringa $v \shortmid w$) è ottenibile da un algoritmo che risulta caratterizzato permanentemente da una stringa (nel caso esaminato la v).

Osserviamo anche che la stringa $v \shortmid w$ si può considerare anche come il risultato di una operazione unaria (costruibile) applicata alla coppia in forma $\langle v, w\rangle$.

B02 c.08 Un altro problema semplice ma basilare consiste nel decidere se un fissato carattere c disponibile in un registro C della macchina "occorre" in una data stringa w, cioè compare come componente di tale stringa; a questa w diamo il ruolo di dato caratterizzante ogni istanza del problema.

Si tratta di scorrere le caselle del nastro contenente w e controllare se c coincide con il contenuto di almeno una delle sue caselle.
Che un esecutore umano riesca a far questo affidabilmente appare ovvio. Per completezza vediamo quali istruzioni consentono di farlo alla nostra MSM.

L'elaborazione inizia con il controllo in uno stato iniziale q_1 e con l'istruzione che prevede la lettura del contenuto della casella iniziale del nastro T sul quale viene presentata la w, il suo trasferimento in un registro R, l'avanzamento della testina e il passaggio a uno stato q_2.

Il secondo passo consiste nel controllo della uguaglianza del contenuto di R con il carattere di fine nastro $\dashv$; se si ha la uguaglianza si passa a uno stato f_0, altrimenti si passa a uno stato q_3.
In q_3 si controlla se il contenuto di R coincide con il contenuto c del registro S; se questo si riscontra si passa allo stato f_1, se no si torna a q_1.

Gli stadi successivi riguardano l'esame delle successive caselle di T e si devono concludere o con il reperimento di una occorrenza del carattere c o con il reperimento del terminatore $\dashv$. Nel primo caso il controllo passa in f_1, nel secondo in f_0.

Negli stati f_0 e f_1 sono segnalate le conclusioni: f_1 comporta la emissione di un carattere 1 e f_0 comporta l'emissione di uno 0 su un nastro di uscita T_r di una sola casella al quale si chiede solo di registrare uno dei due caratteri.

Ciascuno dei caratteri 0 e 1 viene detto **bit**, contrazione di *binary digit*.

Con i caratteri 1 e 0 si conviene di rappresentare, risp., la accettazione ed il rifiuto di c in quanto componente della stringa w.
In altre parole: il carattere 1 rappresenta la decisione positiva, il sì, alla richiesta "c occorre nella w?" , mentre lo 0 rappresenta la decisione negativa, la risposta no alla stessa domanda.
In altri termini ancora: 1 rappresenta la valutazione **vero** per l'enunciato "c occorre in w", mentre 0 rappresenta la corrispondente valutazione **falso**.

La procedura trovata consente in particolare di decidere se un carattere appartiene o meno a un particolare alfabeto che evidentemente può essere fornito da una stringa di caratteri tutti diversi.

Ai bits 0 e 1 viene spesso attribuito il ruolo dirappresentare, risp., l'esito negativo e positivo dei cosiddetti **problemi dicotomici**, problemi che si concludono con il decidere se un certo dato di ingresso verifica o meno una certa proprietà.

Questi problemi possono anche essere presentati come dilemmi tra due contrapposte situazioni; uno di questi dilemmi spesso compare come sottoproblema di un problema **P** più comprensivo e nella elaborazione di una delle istanze di **P** la corrispondente soluzione del dilemma serve per decidere quale imboccare delle due strade predisposte per proseguire l'elaborazione.

Nel panorama delle elaborazioni SAP le manovre che effettuano scelte dicotomiche sono molto comuni: in particolare si hanno scelte che dipendono dal confronto tra le valutazioni di due stringhe (rappresentanti nomi, grandezze o configurazioni più composite) e scelte che riguardano la occorrenza o meno di una stringa in un elenco o un testo di qualche genere.

B02 c.09 Un algoritmo che può essere considerato un ampliamento di quello visto in c08 è quello che consente di decidere se una stringa $w := b_1 b_2 ... b_n$ presentata su un nastro T_i coincide o meno con una stringa $v = a_1 a_2 ... a_m$ che supponiamo sia stata precedentemente acquisita e resa disponibile su un suo nastro T_p alla macchina MSM **C**.

Questa decisione viene governata da un programma che organizza lo scorrimento simultaneo dei due nastri T_i e T_p.

Inizialmente la macchina **C** si trova nello stato q_1 con le due testine sui due nastri di ingresso ciascuna posizionata sopra la casella che segue quella contenente il carattere di inizio nastro $\vdash$.
Il primo passo vede il trasferimento del carattere letto da T_i nel registro R, l'avanzamento della corrispondente testina e il passaggio a q_2.

Il secondo passo vede il trasferimento del carattere letto da T_p nel registro S, l'avanzamento della corrispondente testina e il passaggio a q_3.

Se il carattere copiato da T_i in R è $\dashv$ si passa allo stato f. In questo stato se anche il carattere letto da T_p in S è $\dashv$ si passa allo stato f_1 nel quale si ha l'emissione di 1 su un nastro monocasella conclusivo T_z, il segno 1 atto a segnalare che le due stringhe coincidono ($w = v = \sqcup$) e successivamente si ha l'arresto.

In caso contrario si passa in uno stato f_0 in cui si emette su T_z un bit 0 che segnala che le due stringhe sono diverse e si effettua l'arresto.

Se il carattere letto da T_i in R è un a_j (diverso da $\dashv$) si passa allo stato q_4.

Qui se il carattere letto da T_p in S coincide con a_j si torna in q_1, mentre nel caso opposto si passa in f_0 per la giungere alla conclusione negativa in seguito alla constatazione che w e v in due rispettive posizioni omologhe presentano due caratteri componenti diversi.

Con il ritorno nello stato q_1 si è individuato uno stadio iterativo riguardante gli stati q_1, q_2, q_3 e q_4.

L'elaborazione quindi prosegue con nuovi stadi iterativi concernenti le successive posizioni delle due stringhe a confronto.

L'iterazione che si è individuata si deve concludere con il reperimento di un carattere $\dashv$ aut su uno solo dei due nastri in esame e quindi con la conseguente segnalazione su T_z della diversità delle stringhe, aut con il reperimento di $\dashv$ in due caselle che occupano la stessa posizione sui due nastri e con la segnalazione della coincidenza $w = v$.

B02 c.10 Il problema proposto in c09 si è risolto con una iterazione che riguarda le successive corrispondenti caselle di due nastri, ossia con una manovra che chiamiamo "scorrimento parallelo di due sequenze", in termini concreti "scorrimento di due segmenti di nastro".

Le possibili conclusioni del relativo algoritmo vengono formulate matematicamente con le due scritture $w = v$ e $w \neq v$.

In queste scritture i segni (identificatori) v e w e i segni $=$ e $\neq$ (che diremo esprimere relazioni) sono da attribuire ad $\mathbb{A}M$, in quanto non sono utilizzati in elaborazioni, ma servono a formulare considerazioni sopra le elaborazioni.

I segni $=$ e $\neq$ sono particolari esempi di quelli che chiamiamo **connettivi relazionali**, oggetti formali che compaiono in scritture della forma

(1) $$\textit{primo membro} \quad \textit{connettivo relazionale} \quad \textit{secondo membro} \ .$$

Ciascuna di queste scritture ha lo scopo di affermare che i due membri della coppia

$$\langle \ \textit{primo membro} \ , \ \textit{secondo membro} \ \rangle$$

soddisfano la proprietà rappresentata dal connettivo relazionale (come l'essere uguali o l'essere il primo minore del secondo); è evidentemente una situazione positiva se una affermazione di questo genere può risultare algoritmicamente verificabile (come accade nei due casi suddetti).

Anche ogni proprietà si può qualificare come una entità caratterizzata dallo stabilire un collegamento tra due altre entità (eventualmente coincidenti); queste sono da considerare i membri di una coppia associata alla proprietà stessa che si dice "coppia che soddisfa la relazione".
Una tale situazione viene espressa anche affermando che la proprietà "mette in relazione" i due membri.

Segnaliamo che più avanti, servendoci della nozione di insieme, potremo ridurre la nozione di proprietà fatta emergere da un algoritmo che effettua una decisione alla nozione di insieme costruibile di coppie.

L'algoritmo esaminato in c09 si applica, per esempio, alla gestione dei processi che effettuano l'accettazione o il rifiuto di una richiesta di utilizzare un dispositivo personale (PDA, smart phone, tablet, ...) da parte di un possibile utente che si presenta al dispositivo stesso inviandogli una stringa che deve costituire il corretto codice di accesso.

Si osserva che organizzando l'operazione di confronto tra caratteri attraverso opportune iterazioni si è ottenuta la manovra di verifica dell'uguaglianza tra due stringhe qualsiasi.

Questo fa intravvedere la possibilità in generale di trasformare una procedura P_c che implementa una costruzione basata su caratteri c_1, c_2,... e su loro sequenze v_1, v_2, ... in una procedura P_w che implementa la analoga costruzione che opera su stringhe w_1, w_2, ... e su sequenze di stringhe L_1, L_2, ... aventi ruoli e obiettivi che generalizzano quelli, risp., dei suddetti caratteri c_i e delle suddette stringhe w_j.

Questo ampliamento di prestazioni si ottiene sostituendo i gruppi di istruzioni C_h costituenti P_c e riguardanti caratteri e stringhe con corrispondenti gruppi di istruzioni S_h i quali effettuano operazioni analoghe ma più elaborate in quanto riguardanti, risp., stringhe e sequenze di stringhe.

B02 c.11 Un'altra manovra semplice che si può estendere a importanti sviluppi, consiste nel trasformare una stringa $v := a_1\, a_2 \ldots a_{s-1}\, a_s$ nella sua cosiddetta **stringa riflessa** $a_s\, a_{s-1} \ldots a_2\, a_1$.

Per esempio la riflessa della stringa `ROMA` è `AMOR`, la riflessa della stringa `riflessa` è `asselfir`.

Ovviamente la riflessa di una stringa monogramma, cioè di un carattere, è il carattere stesso.

Inoltre conviene assumere che la riflessa della stringa muta sia la stessa stringa muta; si constata facilmente che tale scelta apre la strada ad argomentazioni ed elaborazioni più generali e più versatili.

La esecuzione della riflessione di una stringa v si ottiene organizzando una prima iterazione alla ricerca sul nastro di ingresso della fine della v, cioè la ricerca di $\dashv$ (o di un primo carattere null), e una seconda successiva iterazione che scorre all'indietro la v per copiarne le componenti sulle successive caselle del nastro di uscita fino alla emissione di un $\dashv$ o di un null conclusivo.

La trasformazione riflessione di stringhe costituisce, dopo l'accodamento visto in c07, un secondo esempio di operazione unaria costruibile, cioè di una operazione che da un qualsiasi oggetto di un certo genere, nel caso attuale da una stringa, ottiene un oggetto dello stesso genere.

Si osserva che effettuando la riflessione di una stringa precedentemente ottenuta riflettendo una stringa v si ottiene la stessa stringa v.

Questa proprietà si esprime dicendo che la riflessione è una **operazione unaria involutoria**, o, più in breve, che essa è una **involuzione**.

Sul termine involuzione, che riprenderemo innparticolare in c14, ora diciamo solo che riguarda trasformazioni di certe entità in entità simili (in stringhe che possono essere attribuite a uno stesso alfabeto di lavoro) in modo tale che applicata due volte riproduce l'entità di partenza.

In seguito incontreremo molte involuzioni, in particolare in B13c, in B16e e in molte parti della geometria e dell'algebra [B22, B54].

B02 c.12 Per denotare la riflessa della stringa v useremo la notazione $v^{\leftarrow}$, espressione nella quale compare il segno $\leftarrow$ con il ruolo di identificatore dell'operazione riflessione di stringhe, trasformazione definita costruttivamente dall'algoritmo che consente di attuarla in c11.
Anche questo segno lo collochiamo nell'alfabeto $\mathbb{A}M$.

Il segno " $\leftarrow$ ", rispetto alla stringa v su cui agisce, viene collocato immediatamente a destra, nella cosiddetta **posizione suffissa** o posizione postfissa.
La stringa v rispetto all'operazione riflessione svolge il ruolo di **operando unario**.
Data la sua posizione innalzata si dice anche che l'identificatore della riflessione rispetto al suo operando si colloca in **posizione esponenziale**.

Per esprimere il carattere involutorio della riflessione delle stringhe si utilizza il seguente enunciato:

$$\text{Per ogni stringa } w \text{ si ha } \left(w^{\leftarrow}\right)\leftarrow \;=\; w \;.$$

La riflessa di una stringa si individua anche con la seguente definizione nella quale la stringa operando è riferita ai suoi caratteri componenti presentati in una forma generica:

$$\text{Per ogni stringa } w \;=\; a_1\, a_2\, a_3\, \cdots a_{s-1}\, a_s \,:\, w^{\leftarrow} \;:=\; a_s\, a_{s-1}\, \cdots\, a_3\, a_2\, a_1 \;. \tag{1}$$

Si conviene che l'espressione precedente implichi anche l'invarianza per riflessione dei monogrammi e della stringa muta.
Questa definizione viene preferita quando non si vogliono richiamare dettagli operativi.

Nel seguito incontreremo molte altre operazioni unarie; per le azioni di molte di queste si usano notazioni nelle quali il simbolo che le denota viene collocato alla sinistra dell'operando, nella cosiddetta **posizione prefissa**.

In effetti si incontrano varie notazioni della forma ω,x, dove ω denota una operazione e x un suo operando; la precedente scrittura in genere si può semplificare nella $\omega\, x$. Un esempio è dato dalle notazioni dei numeri negativi come -3 e $-n$ [B04e04]

B02 c.13 La definizione del paragrafo precedente, come quella in c11, non si impegna sulla costituzione delle stringhe che vengono riflesse. Quando si invocano le elaborazioni di una MSM le considerazioni sulla riflessioni sono assicurate per le stringhe su $\mathbb{A}W_{\mathbf{C},\cdot}$.
Tuttavia la riflessione può essere applicata utilmente a ogni genere di stringa, anzi, come vedremo, ad ogni genere di sequenza finita [B06b11].

Questo sospetto poco impegno si può giustificare con alcune considerazioni sulle definizioni degli alfabeti.

La definizione di $\mathbb{A}W_{\mathbf{C},\cdot}$ e di ogni altro alfabeto si basa su convenzioni che dipendono dalle problematiche che si intendono affrontare.

La crescita dell'*apparato* AMI va decisamente favorita per renderlo in grado di affrontare la più ampia gamma di problemi che possono essere matematizzati.
Quando si passa da una problematica ad un'altra e in particolare quando si amplia una problematica per affrontare problemi di maggiore portata o di maggiore incisività risulta spesso utile modificare gli alfabeti per i diversi scopi.

In altre parole in genere conviene fare riferimento ai problemi e ai relativi alfabeti in modo elastico, facilitando l'adarrabilità dei discorsi.
Spesso si presenta l'opportunità di estendere la portata delle argomentazioni sulle elaborazioni che sono processi osservabili e dipendenti dagli ambienti e dalle circostanze nelle quali i problemi si presentano.
La motivazione evidentemente spiegabile è quella di cercare di disporre di procedimenti di maggiore portata e incisività.

Molte di queste estensioni si possono ridurre ad ampliamenti degli alfabeti, compreso l'alfabeto delle istruzioni quando si adotta una macchina più prestante di una precedente.
Altre estensioni corrispondono al passaggio di alcuni cartatteri dall'alfabeto delle considerazioni generali a quello dei dati.

In particolare una operazione come la riflessione si rivela utile quando si applica alle formule matematiche: un esecutore $\overline{\mathsf{C}}$ (o un programma) effettivamente incaricato di una tale operazione tratta un certo gruppo di simboli che precedentemente sono stati assegnati a $\mathbb{A}M$) come appartenenti a $\mathbb{A}W_{W,\overline{\mathsf{C}}}$.

Osserviamo anche che in questa fase iniziale dell'*esposizione* disponiamo di una strumentazione decisamente povera e dobbiamo procedere lasciando inespresse molte precisazioni che rischierebbero la prolissità, limitandoci alle precisazioni necessarie per evitare fraintendimenti.
In particolare, confidando nella loro familiarità, utilizziamo nozioni ben note (numeri specifici, funzioni, relazioni, ...) ma che ci proponiamo di definire con precisione e di giustificare i loro ruoli.

B02 c.14 Consideriamo una trasformazione algoritmica, entità che chiamiamo anche **funzione algoritmica** e che denotiamo con τ, segno da collocare in $\mathbb{A}M$.
Supponiamo inoltre che essa sia in grado di agire su oggetti di un tipo T_D e che produca oggetti di un tipo T_C e precisiamo che, se v_i denota un generico oggetto di tipo T_D, l'oggetto nel quale τ lo trasforma lo denotiamo con $v_i{}_!\tau$.

L'oggetto v_i viene detto argomento della funzione τ e $v_i{}_!\tau$ valore che la funzione τ assume in corrispondenza di v_i.

Se la funzione algoritmica produce oggetti dello stesso tipo si parla di **endofunzione algoritmica**

Una evidente endofunzione algoritmica è la riflessione delle stringhe su qualsiasi alfabeto.

Se si trova una funzione algoritmica θ in grado di agire su tutti gli oggetti $v_i{}_!\tau$ e che per ciascuno di essi, cioè per ogni v_i, fornisce lo stesso $v_i = \big(v_i{}_!\tau\big)_!\, tet$, allora questa θ viene detta **funzione inversa algoritmica** o trasformazione inversa algoritmica della funzione τ.

Una funzione algoritmica dotata di funzione inversa algoritmica si dice **funzione algoritmica invertibile** o trasformazione algoritmica invertibile.

Una evidente funzione algoritmica invertibile è la riflessione delle stringhe su qualsiasi alfabeto. Questa è una involuzione ed è anche evidente che ogni involuzione che può essere effettuata da un algoritmo è una funzione algoritmica invertibile.

Si osserva che, dati due diversi oggetti v_i e v_j sui quali agisce una funzione algoritmica invertibile τ, non può accadere che sia $v_i \tau = v_j \tau$; infatti se così fosse sarebbe

$$v_i \;=\; \big(v_i \tau\big)\theta \;=\; \big(v_j \tau\big)\theta \;=\; v_j \;,$$

contro l'ipotesi $v_i \neq v_j$.

Si osserva poi che anche ogni trasformazione θ inversa di una trasformazione invertibile algoritmica τ è una trasformazione invertibile algoritmica e si constata che la sua inversa è la τ stessa. Infatti se scriviamo $w_i := v_i \tau$ si ha

$$\big(w_i \theta\big)\tau \;=\; \big((v_i \tau)\theta\big)\tau \;=\; v_i \tau \;=\; w_i \;.$$

Per affermare che la θ è la trasformazione inversa della τ scriviamo, servendoci del segno $^{-1}$ da collocare in $\mathbb{A}M$,

$$\theta^{-1} \;=\; \tau \;.$$

Nello scenario precedente τ e θ si possono scambiare e possiamo anche affermare $\tau^{-1} \;=\; \theta$.

B02 c.15 Si osserva anche che il passaggio da una trasformazione algoritmica τ invertibile da un insieme D a insieme C alla sua inversa τ^{-1} si può considerare una trasformazione T che riguarda oggetti definiti in modo parzialmente circostanziato.
Più precisamente osserviamo che questa T potrebbe essere rappresentata dalla stessa scrittura $^{-1}$ quando la supponessimo agente su trasformazioni algoritmiche invertibili.
Constatiamo anche che questa T possiede una trasformazione inversa.
Si osserva infine che questa T si può chiamare endofunzione relativa all'insieme delle trasformazioni algoritmiche invertibili e che, essendo dotata di inversa, si può considerare una involuzione.

Non siamo invece in grado di affermare che essa sia una trasformazione algoritmica, o costruibile, in quanto gli oggetti su cui può agire, le trasformazioni algoritmiche invertibili, sono state definite solo in base alla constatazione che soddisfano una proprietà definita solo di prospettiva e quindi siamo ben lontani dall'essere in grado di affermare che si trovi un algoritmo che trasformi ciascuno degli algoritmi di trasformazione tra due insiemi imprecisati C e D nel suo algoritmo inverso.

A questo punto dobbiamo riconoscere che se si vogliono stabilire enunciati affidabilmente condivisibili e riutilizzabili costruttivamente si devono precisare numerosi elementi.

B02 c.16 Torniamo a situazioni assodate e constatiamo che i più semplici esempi di trasformazioni di stringhe invertibili algoritmiche, oltre alla riflessione, sono forniti dalle **ricodifiche alfabetiche**, le trasformazioni che sostituiscono alcuni caratteri di un alfabeto A con un ugual numero di caratteri di un secondo alfabeto B costituito dallo stesso numero di caratteri (senza escludere che possa essere $\mathsf{A} = \mathsf{B}$).

Un semplice esempio è la ricodifica che trasforma la stringa $alfa$ nella stringa $\alpha\lambda\phi\alpha$.

L'algoritmo che implementa la ricodifica alfabetica inversa, che costituisce un algoritmo inverso del primo, è strettamente simile: è semplicemente la ricodifica alfabetica che trasforma le letter greche minuscole nelle corrispondenti latine e quindi che in particolare trasforma la stringa $\beta\epsilon\tau\alpha$ nella $beta$.

Si osserva che ogni ricodifica alfabetica può essere effettuata da un prevedibile algoritmo che si serve di due stringhe, la prima contenente l'alfabeto da sostituire, la seconda l'alfabeto sostitutivo con i caratteri collocati nelle stesse posizioni dei loro rispettivi caratteri sostituendi.

Quindi gli algoritmi di ricodifica alfabetica possono essere trasformati negli inversi da un unico algoritmo che si limita a scambiare le due strighe che forniscono i due alfabeti e quindi l'inversione degli algoritmi invertibili consistenti in ricodifiche alfabetiche va considerata una involuzione algoritmica.

B02 c.17 Per molte operazioni unarie, alias trasformazioni, alias funzioni, possono risultare convenienti notazioni diverse da qualla riguardante espressioni come $v_i{}_\shortmid\tau$ e che chiamiamo "a funzione suffissa".
Ad essa equivale la notazione che diciamo "a funzione prefissa" della forma $\tau_\shortmid v_i$, nella quale compare il connettivo $_\shortmid$ da collocare in $\mathbb{A}M$.

In genere le due notazioni viste si possono semplificare senza incorrere in serie ambiguità, risp., nella $v_i\,\tau$ e nella $\tau\,v_i$.

Inoltre si usa molto spesso quella che diciamo **notazione funzionale** la quale esprime l'effetto della operazione unaria ϕ sull'oggetto x cpn la scrittura $\phi(x)$.
Qui il segno dell'operazione unaria viene seguito dalla notazione dell'operando, x, racchiusa tra due parentesi che possono essere considerati membri dell'alfabeto di scansione e che hanno il preciso ruolo di **delimitatori di argomenti di funzione** che in questo caso si limitano a delimitare un solo argomento.

Con queste notazioni alternative la proprietà di involutorietà viene espressa, risp., dalle seguenti scritture:

$$\phi(\phi\,x) \;=\; x \qquad , \qquad \phi(x) \;=\; \phi\big(\phi^{-\mathbf{1}}(x)\big)\ .$$

Si constata che per la riflessione di una giustapposizione di stringhe si può affermare

(1) per stringhe arbitrarie u e v abbiamo $(u_\shortmid v)^{\leftarrow} \;=\; (v^{\leftarrow})_\shortmid(u^{\leftarrow})$.

Per esempio $(\mathtt{abc}_\shortmid\mathtt{de})^{\leftarrow} \;=\; \mathtt{ed}_\shortmid\mathtt{cba}$.

Osserviamo che, evidentemente, si può assumere come definizione della riflessione di stringhe anche l'uguaglianza (1): infatti applicandola ripetutamente a stringhe ottenute giustapponendo caratteri e tenendo conto che la stringa muta e tutti i monogrammi non cambiano per riflessione, si giunge alla (1).

B02 c.18 Le endofunzioni algoritmiche invertibili su oggetti di un insieme T si dicono **permutazioni algoritmiche** di T.
Particolarmente interessanti sono le permutazioni, evidentemente algoritmiche, delle lettere di un alfabeto A.
Si constata facilmente che ciascuna di esse, che denotiamo con $\boldsymbol{P}$, induce una permutazione algoritmica delle stringhe sullo stesso alfabeto A.

Tra le permutazioni di un alfabeto si trovano in particolare gli scambi tra due sue lettere.
Evidentemente esse sono delle involuzioni algoritmiche, cioè sono endofunzioni algoritmiche che applicate due volte riproducono tutti gli oggetti (stringhe) sui quali possono agire.

Tra le **endofunzioni involutorie algoritmiche** si rivela opportuno collocare anche le cosiddette **trasformazioni identità**, operazioni che, meramente, lasciano invariati tutti gli oggetti sui quali è loro consentito di agire.

L'algoritmo di riproduzione di una stringa [c04] può considerarsi una implementazione della trasformazione identità per l'insieme delle stringhe sulle quali opera.

Si dicono **invarianti di una trasformazione** o **punti fissi di una trasformazione** tutti gli oggetti sui quali essa non fa che trasformarli in se stessi.
Ovviamente gli invarianti di ogni trasformazione identità sono tutti gli oggetti sui quali essa si conviene possa agire).

Consideriamo due alfabeti B e C privi di caratteri in comune e l'alfabeto A costituito dai caratteri di B e da quelli di C (per il quale scriveremo $\mathsf{A} = \mathsf{B}\,\dot\cup\,\mathsf{C}$ [B08e02).

Una ricodifica delle stringhe su A che viene indotta da una permutazione di tutte le lettere di B e non modifica alcuna lettera di C ha come invarianti tutte e sole le stringhe che non contengono lettere appartenenti ad A.

Particolari ricodifiche su un alfabeto A sono quelle che sono indotte dallo scambio di due diverse lettere di A. Evidentemente queste ricodifiche sono delle involuzioni.

B02 c.19 Si dice **stringa palindroma** una stringa w che coincide con la sua riflessa, cioè tale che valga l'uguaglianza $w^{\leftarrow} = w$.
Evidentemente le stringhe palindrome sono gli invarianti della riflessione di stringhe.

Tra le stringhe palindrome si trovano la stringa muta, tutti i monogrammi, tutte le stringhe sopra un solo tipo di carattere e stringhe come *abaacbcaaba*, *emme*, *ANNA* e *anilina*; se non si da importanza allo spazio bianco si considera stringa palindroma anche *etna gigante* .

Segnaliamo anche la scritta in lingua latina ritrovata su molti reperti archologici

SATOR AREPO TENET OPERA ROTAS

e la recente

in girum imus nocte et consumimur igni .

Questa stringa viene spesso presentata nelle caselle di una matrice quadrata 5 per 5 per mostrare che può essere letta sia per righe procedendo dall'alto in basso da sinistra a destra, sia per righe dal basso in alto da destra a sinistra, sia per colonne da sinistra a destra e dall'alto in basso, sia infine per colonne da destra a sinistra e dal basso in alto.

S	A	T	O	R
A	R	E	P	O
T	E	N	E	T
O	P	E	R	A
R	O	T	A	S

Le palindrome sono elementi lessicali attentamente cercate in tutte le lingue; molte palindrome sono segnalate sul Web, per esempio in `http://www.palindromelist.net/` e in `http://plgrs.lacab.it/palindromi/` .

B02 c.20 Si constata facilmente che le stringhe palindrome su un dato alfabeto A sono tutte e sole le stringhe delle forme $w_{\shortmid}(w^{\leftarrow})$ e $w_{\shortmid}c_{\shortmid}(w^{\leftarrow})$, con w stringa arbitraria su A e c carattere arbitrario in A .

Si constata anche che ogni stringa palindroma o è la stringa muta, o è un monogramma, o assume la forma $w_{\shortmid}(w^{\leftarrow})$ oppure assume la forma $w_{\shortmid}c_{\shortmid}(w^{\leftarrow})$.

Evidentemente le stringhe della terza forma (come *atta* e *avallava*) sono le palindrome con un numero positivo pari di lettere, mentre quelle della quarta forma (come *anilina*, *ingegni*, *radar*) sono le palindrome con un numero dispari di lettere.

Si osserva che ciascuna delle stringhe palindrome costituite da repliche di un unico carattere ricade sotto la terza oppure sotto la quarta forma.

B02 c.21 In generale per ogni involuzione ogni oggetto al quale si può applicare che non sia invariante insieme con il suo trasformato costituisce quello che chiamiamo **duetto di oggetti coniugati per una involuzione**.
oggetto e *otteggo* costituiscono un esempio di duetto di oggetti coniugati e più in generale una stringa nonpalindroma e la sua riflessa costituiscono un duetto di stringhe coniugate per la riflessione.

Occorre segnalare che il sostantivo "duetto", che qui compare spesso, di solito viene usato raramente e gli si preferisce il termine "coppia di oggetti coniugati".
Al termine coppia qui si preferisce duetto, in quanto i due componenti di una coppia svolgono i due ruoli diversi di primo e secondo componente, mentre non si vuole attribuire alcuna preferenza ai membri di un insieme di due elementi.

È semplice delineare un algoritmo che decide se una data stringa è palindroma o meno.

Con una prima manovra si ricava dal nastro di ingresso procedendo da destra a sinistra la riflessa della stringa in esame memorizzandola su un nastro di lavoro; con una seconda manovra si decide se stringa data e stringa riflessa coincidono in modo da poter affermare o negare la proprietà di palindromia.

Si osserva che questo algoritmo si può costruire riprendendo due algoritmi più semplici visti in precedenza [c11 e c09].

Questo atteggiamento del riprendere algoritmi tendenzialmente semplici per definire parti di algoritmi più articolati, come vedremo, è spesso molto conveniente ed è molto praticato; una generalizzazione di qiesto atteggiamento risulta conveniente e praticato per l'intera organizzazione dell'*apparato*.

B02 c.22 Illustriamo le soluzioni di altri semplici problemi sopra le stringhe le cui soluzioni possono essere usate come "sottosoluzioni" facenti parte di vari problemi di rilevante interesse.

Cominciamo con il problema di stabilire se una stringa $w := a_1\, a_2\, ...\, s_{s-1}\, a_s$ presenta o meno caratteri ripetuti.

Per questo si organizza una iterazione primaria, di livello 1, ciascuno stadio della quale inizia copiando in un registro R i caratteri della w procedendo dal primo, a_1, al penultimo, a_{s-1}.

Nel generico stadio primario nel quale si è posto in R il carattere a_i (per $i = 1, 2, ..., s-1$) si esegue una cosiddetta iterazione di livello 2 o iterazione secondaria o **sottoiterazione** della primaria; in essa si confronta il carattere a_i con ciascuno dei caratteri a_j che lo seguono nella w, cioè con ciascuno degli a_j per $j = i+1, ..., s$.

Se si incontra una coincidenza si evitano i successivi stadi iterativi primari e si conclude l'elaborazione segnalando che la stringa presenta ripetizioni;
se non si incontra nessuna coincidenza in ciascuna degli stadi primari la stringa è riconosciuta come **stringa nonripetitiva**; come sinonimo di questa locuzione si può usare anche stringa iniettiva.

Una semplice variante di questo algoritmo permette di eliminare da una stringa tutti i suoi caratteri ripetuti.
Con una prima manovra si trascrive sul nastro di uscita T_u il primo carattere. Quindi si organizza una iterazione primaria con la quale si procede sui successivi caratteri in ingresso a partire dal secondo per decidere per ciascuno di essi se coincide con uno dei caratteri già registrati su T_u, attività che richiede una iterazione secondaria sulle caselle occupate di T_u; nel caso si riscontri un carattere ripetuto lo si trascura, mentre in caso contrario lo si accoda ai quelli già trascritti in T_u.
Alla fine sul nastro T_u si ottiene la stringa che chiamiamo la **stringa ridotta nonripetitiva-c** di quella esaminata.

La distinzione tra stringhe nonripetitive e stringhe ripetitive è quindi chiaramente decidibile. In particolare un esecutore che riceve stringhe da sottoporre a una elaborazione che dovrebbero essere nonripetitive (ad esempio stringhe che devono definire degli alfabeti) è in grado di verificare se i dati che gli sono proposti sono adeguati oppure se sono da dichiarare illegali.

B02 c.23 In alcune circostanze operative risultano utili algoritmi di eliminazione che da una stringa tolgono occorrenze di carattere non voluti.

Il più semplice di questi algoritmi trasforma una stringa w nella stringa privata delle occorrenze di un carattere c considerato spurio.
Esso consiste in una sola iterazione nella quale si scorre la w per confrontare i suoi successivi caratteri a_i con c; nel caso a_i sia diverso da c lo si trascrive sul nastro di uscita che inizialmente conteneva solo $\vdash \dashv$, mentre in caso $a_i = c$ non si fa che passare alla occorrenza di carattere successiva nella w.

Un poco più elaborato è l'algoritmo che elimina da una prima stringa di ingresso w tutti i caratteri che compaiono in una seconda stringa data v.

Per ottenere questo occorrono una iterazione primaria sui caratteri a_i costituenti la w e una iterazione secondaria sui caratteri b_j della v; se nessuno di questi coincide con a_i trascrive tale carattere sulla destra del nastro di uscita, mentre si evita questa manovra se si incontra un b_j coincidente con a_i.

Si osserva che potrebbe essere conveniente sostituire preliminarmente v con la corrispondente stringa ridotta senza ripetizioni in modo da evitare confronti inutili con i suoi caratteri b_h coincidenti con un precedente b_j.
Quale di questi due modi di procedere sia da giudicare come il più conveniente dipende da quello che si può prevedere sulla composizione della v e della w.

Come nel caso precedente, molti dei semplici algoritmi visti in grado di modificare o analizzare stringhe possono essere utilizzati per attività qualificabili come **validazioni dei dati**, attività volte a garantire che i dati presentati per affrontare istanze di un definito problema siano in grado di individuare correttamente ciascuna di tali istanze.

Manovre simili a quelle di validazione si possono rendere necessarie quando una manovra non conclusiva di una elaborazione articolata fornisce un risultato parziale che utilizzato da manovre successive porterebbe a manovre inutili, forse dannose e probabilmente rischiose; talora risulta conveniente sottoporre ciascuno dei risultati parziali sospetti a una validazione in grado di evitare le situazioni che portano verso manovre successive deprecabili.

In molte elaborazioni si raggiungono stadi intermedi nei quali risultano individuati oggetti intermedi tra i quali uno si rivelerà come il migliore da portare avanti per raggiungere il risultato finale.
La scelta di un tale oggetto intermedio più adatto a proseguire talora si può anticipare con una analisi delle caratteristiche degli oggetti intermedi in modo da evitare di effettuare elaborazioni sopra tutti gli oggetti intermedi candidati per servire alla scelta a posteriori della soluzione più adatta.

B02 c.24 Poniamo il problema di decidere se tutti i caratteri di una stringa v presentata alla MSM sul nastro T_v occorrono in una seconda stringa w presentata sul nastro T_w.

Si tratta di organizzare una iterazione primaria in ciascuno stadio della quale viene posto in un registro R uno dei successivi caratteri di v e di organizzare una sottoiterazione della precedente che scorre i successivi caratteri di w per stabilire se nessuno di essi coincide con quello in R.

Se si ha una coincidenza si interrompe la sottoiterazione e si passa ad un successivo carattere di v; se si trova la occorrenza in w di tutti i caratteri di v si conclude l'indagine positivamente.
Con il primo eventuale carattere di v che non occorre in w si conclude l'indagine negativamente interrompendo l'iterazione primaria.

Evidentemente un carattere compare in una generica w se e solo se compare nella ridotta nonripetitiva-c della w.

Quindi in talune circostanze per stabilire se tutti i caratteri della v compaiono nella w conviene sostituire preliminarmente la w con la sua ridotta nonripetitiva-c.

È facilmente definibile anche un algoritmo che decide se in due stringhe v e w sono presenti gli stessi caratteri. Questa prestazione è ottenibile applicando due volte l'algoritmo precedente per stabilire se ciascuna di esse ha tutti i caratteri occorrenti nell'altra.
Anche per questa soluzione si pone una alternativa: può essere coveniente sostituire preliminarmente entrambe le stringhe con le rispettive ridotte nonripetitive-c.

B02 c.25 È facilmente prevedibile anche un algoritmo che, date due stringhe v e w, consente di stabilire se contengono gli stessi caratteri o se viceversa vi sono caratteri che compaiono solo in una delle due stringhe.

Esso si serve di due nastri ausiliari leggibili e riscrivibili U_v e U_w sui quali vengono preliminarmente copiate la v a la w in previsione di modificarle. Serve anche un carattere ϵ che sicuramente non si incontra nelle stringhe da esaminare e da considerare elemento estraneo; si potrebbe rendere disponibile in $\mathbb{A}S$ questo ϵ da scegliere sapendo cosa ci si può aspettare delle possibili u e v.

Con una iterazione primaria si scorrono i caratteri della v e per ciascuno di essi a_i che sia diverso da ϵ si cerca con una sottoiterazione se compare nella w.

Se uno di questi a_i che sia diverso da ϵ non compare nella w resta stabilito che v e w non presentano gli stessi caratteri e si trasformano in ϵ tutte le eventuali restanti occorrenze di a_i in v.

Se a_i compare nella w si trasformano in ϵ tutte le sue occorrenze nella v e nella w. Idue fasi ulteriori si scorrono U_v esu U_w per eliminare tutte le occorrenze di ϵ.
Si conclude che in v e w occorrono gli stessi caratteri sse alla conclusione su U_v e su U_w non rimane alcun carattere.

In caso contrario rimane su U_v la stringa dei caratteri che occorrono in v e non in w e rimane su U_w la stringa dei caratteri che occorrono in w e non in v.

Con poche modifiche, ma utilizzando più caratteri diversi da quelli che si ammette possano occorrere nella v e nella w ma ciascuno posto in una corrispondenza riconoscibile con un carattere effettivamente trovato nelle due stringhe, si può ottenere un algoritmo che segnala le singole occorrenze di caratteri in v e non in w e le singole occorrenze di caratteri in w e non in v.

Testo fruibile in https://www.mi.imati.cnr.it/alberto/ e https://arm.mi.imati.cnr.it/Matexp/matexp_main.php