Capitolo B18:
insiemi generati da procedure

Contenuti delle sezioni

54 pagine

B18:**0.01** All'inizio di questo capitolo mediante alcuni esempi ed alcune considerazioni sulla portata delle MSP e delle procedure, si motiva l'opportunità di introdurre la nozione di infinito a cominciare dalla sua accezione di infinito potenziale. Il punto di vista che si assume è quello di chi si preoccupa della organizzazione complessiva delle conoscenze matematiche.

Successivamente si introduce la nozione di insieme procedurale primariamente attraverso gli esempi delle sequenze illimitate di stringhe e di numeri interi naturali.

Come nei capitoli precedenti e in particolare in B15, si lascia la definizione delle macchine sequenziali programmabili generatrici sul piano della prospettiva delle soluzioni affidabili di una ampia varietà di problemi, lasciando ancora aperta la possibilità di definire nei dettagli vari modelli di questi automatismi.

Nella seconda parte si introducono gli insiemi ricorsivi, insiemi procedurali privilegiati in quanto meglio controllabili, e si procede all'esame delle relazioni e delle operazioni che li riguardano.

In questo capitolo, inoltre, vengono individuate alcune importanti questioni critiche riguardanti sia le procedure e le sequenze illimitate sia gli insiemi in generale e il loro ruolo per i fondamenti della matematica.

B18:a. elaborazioni con risorse illimitate e infinito potenziale

B18:**a.01** Vogliamo ora ampliare le considerazioni sulle macchine sequenziali programmabili, MSP, per chiarire come possono essere utilizzate.
Per questo serve esaminare coppie costituite da una di esse, che denotiamo con M, e da un insieme dei suoi possibili dati iniziali D_{M}; una tale coppia $\langle \mathsf{M}, D_{\mathsf{M}} \rangle$ la chiamiamo macchina da avviare.
In linea generale il linguaggio D_{M} è un sottoinsieme di un ambiente ASs^*, dove A denota l'alfabeto dei caratteri con i quali M prevede siano scritti i suoi dati di avvio.
Per alcune macchine come insieme dei possibili dati potrebbe essere risultare utile un intero monoide libero, per altre solo una sua parte ben definita.
Con alcune MSP l'insieme D_{M} costituisce invece una incognita: si tratta di capire quali dati di ingresso possono portare a evoluzioni utili della M e questo potrebbe richiedere indagini impegnative.

Una situazione chiara è quella che vede un preciso L sottoinsieme di A^* costituito da dati che provocano evoluzioni finite che si concludono con la emissione di una stringa risultato utile e che vede ogni stringa del complementare di L rifiutata e dichiarata "illegale" in quanto incapace di esprimere un dato sensato per la M.
Per queste macchine da avviare $\langle \mathsf{M}, D_{\mathsf{M}} \rangle$ tali che ogni dato iniziale $w \in D_{\mathsf{M}} \rangle$ provoca una evoluzione che si arresta dopo un numero finito di passi con l'emissione di una ben definita (spesso molto elaborata) stringa che denotiamo con $R_{\mathsf{M}}(w)$ e che chiamiamo stringa risultato o stringa trasformata della w da parte della M.
In questo caso si dice che M è una **macchina sequenziale programmabile trasformatrice**, in siglaMSPT.
Nel capitolo precedente ci siamo occupato sostanzialemte solo di queste.

Si possono poi avere situazioni di incertezza con una parte delle stringhe di A^* utili, un'altra di stringhe palesemente illegali e una parte costituita da stringhe che non si sa se possono portare a evoluzioni finite lunghe, a evoluzioni illimitate con emissioni finite, oppure a evoluzioni illimitate con emissioni illimitate.

B18:**a.02** Vi sono inoltre MSP che non richiedono un dato di avvio, o equivalentemente si servono di un insieme di dati ridotto a un solo elemento al quale diamo la orma "*start*"; le chiamiamo **MSP ad avvio unico**.
Tra di queste se ne possono trovare di quelle con evoluzione incerta, forse in grado di emettere un insieme finito di stringhe risultato, forse capaci di evolversi illimitatamente ma con emissioni da chiarire.
Tra le MSP ad avvio unico interessano soprattutto, e molto, quelle che si dimostrano in grado di procedere illimitatamente ad emettere una sequenza illimitata di stringhe risultato sempre diverse $\langle r_1, r_2, r_3, ..., r_k, ... \rangle$ che possono essere considerate utili: queste macchine le chiamiamo **macchine sequenziali programmabili generatrici**, in breve MSPG.

Vi sono invece coppie $\langle \mathsf{M}, D_{\mathsf{M}} \rangle$ per le quali solo alcune delle stringhe di D_{M} provocano una evoluzione finita con l'emissione di una stringa risultato o stringa trasformata; altre stringhe diD_{M} invece, sottoposte al controllo di un algoritmo preliminare sono segnalate come stringhe "inopportune" per l'evoluzione in quanto non in grado di fornire una stringa trasformata sensata o utile oppure capaci di portare a una evoluzione illimitata e improduttiva
mentre altre possono portare a evoluzioni illimitate o a evoluzioni con esito incerto.

Si individuano anche macchine con avvio unico della forma $\langle \mathsf{M}, \{start\} \rangle$ che hanno una evoluzione illimitata nel corso della quale procedono ad emettere .

Rispetto alla trasformazione da tentare le stringhe che conducono a una trasformata sono spesso dette, **stringhe legali per la macchina** M, mentre quelle che non forniscono una trasformata sono chiamate **stringhe illegali per la macchina**.

Tra le MSPT sono da collocare quelle che trasformano un qualsiasi numero intero nel valore fornito da una espressione costruita con le note operazioni aritmetiche; queste trasformazioni si possono considerare operazioni unarie concretamente eseguibili sui numeri interi.
Vi sono inoltre trasformazioni che agiscono su coppie di numeri interi e corrispondono alle varie operazioni binarie definibili sugli interi: somma, prodotto, minimo comune multiplo, massimo comun denominatore, numero dei divisori, ...).

B18:**a.03** La definizione di MSPT non garantisce che l'evoluzione di una tale macchina di fronte a una data stringa di ingresso giunga effettivamente a conclusione. Si possono definire MSPT che di fronte a certe stringhe procedono senza mai arrestarsi e altre che fatte procedere fino a un certo punto non hanno fornito alcun risultato e non fanno capire cosa accadrebbe se fossero fatte operare ulteriormente.

Serve quindi definire come **MSPT algoritmica**, in sigla MSPTA, una macchina MSPT che per ogni stringa di ingresso legale è in grado di fornire la corrispondente trasformata attraverso una sequenza finita di passi e che per ogni stringa illegale la denunci come tale dopo un numero finito di passi.

Nasce allora il problema di decidere se una data MSPT sia o meno una MSPTA; per ora ci limitiamo a segnalare che tale problema a elevati livelli di generalità si rivela assai impegnativo.

Poniamoci ora da un punto di vista strettamente finitistico e vediamo come, limitandosi rigorosamente a considerazioni sopra processi riguardanti strumenti e risorse finite, si incontrano complicazioni che rendono difficoltosa la individuazione di procedimenti che intendono essere utili per finalità di elevato interesse.

Prendiamo in considerazione una MSPTA che a partire da un numero naturale rappresentato da una sua notazione ricava un secondo numero naturale anch'esso fornito da una notazione dello stesso genere.

Essa si può considerare l'implementazione di una regola che fa corrispondere a un intero naturale, con il ruolo di dato di ingresso per la macchina, un secondo intero naturale costituente il corrispondente risultato che risulta quindi dipendente dal primo.

Ad esempio consideriamo il procedimento che dalla notazione decimale n_{10} di un intero naturale n ricava la sua rappresentazione binaria n_2 e da questa la somma delle sue cifre binarie che denotiamo con $\mathsf{bsum}(n)$: per chiarezza segnaliamo che i valori $\mathsf{bsum}(n)$ ottenuti dai primi naturali n sono dati da tabelle come la seguente:

$$\left\downarrow \begin{array}{cccccccccccccccccccc} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 & 13 & 14 & 15 & 16 & \cdots & 2047 & 2048 & \cdots \\ 0 & 1 & 1 & 2 & 1 & 2 & 2 & 3 & 1 & 2 & 2 & 3 & 2 & 3 & 3 & 4 & 1 & \cdots & 11 & 1 & \cdots \end{array} \right\downarrow .$$

Se ci si limita a considerare gli interi naturali n inferiori a un dato intero positivo N, la procedura individua una funzione finita del genere $\lceil\, \mathbb{N} \longmapsto \mathbb{N} \,\rceil$.

Nel caso concreto in cui si utilizza un computer che tratta n secondo la implementazione standard degli interi naturali che si serve di parole di memoria di 16 bits si ha $n < N = 32768$, mentre se si usa l'implementazione standard mediante 32 bits si ha $n < N = 2\,147\,483\,648$.

Per queste funzioni aventi come dominio un intervallo si trova abbastanza facilmente il codominio.
Se il dominio ha la forma $[0 : 2^k - 1]$, si ha come codominio $[0 : k]$, mentre a un dominio $[0 : N)$ con $N \in [2^k : 2^{k+1})$ corrisponde il codominio $[0 : k + 1]$.

B18:**a.04** Per un algoritmo che trasforma un numero naturale in un altro naturale ma che si serve di manovre più elaborate la precisazione del dominio e del codominio può essere molto più impegnativa. Ad esempio la funzione di Eulero [B26d03] applicata ai soli naturali che costituiscono un intervallo $\boldsymbol{I} = \mathbf{[}1 : N\mathbf{]}$ a questo dominio fa corrispondere come codominio un sottoinsieme di $\boldsymbol{I}$ non facile da precisare.

La determinazione precisa del dominio e del codominio di funzioni numeriche tende a complicarsi notevolmente per algoritmi che trasformano più interi naturali in numero non facilmente determinabile di numeri naturali.
Consideriamo la semplicissima somma di due interi naturali m ed n effettuata con un computer che si serve di numeri naturali implementati mediante 32 bits. In questo caso il codominio è l'intervallo $\mathbf{[}0 : 2^{31} - 1\mathbf{]}$, mentre il dominio è l'insieme, non del tutto banale, delle coppie di naturali $\langle m, n\rangle$ che soddisfano la disuguaglianza $m + n < 2^{31}$.

Si possono introdurre MSPTA che presentano limitazioni sul dominio dei dati molto più impegnative da individuare con precisione: basta pensare, ad esempio, alla valutazione di un'espressione un poco elaborata contenente più addizioni e moltiplicazioni sopra una dozzina di operandi i cui valori sono da cercare tra gli interi naturali.

Senza insistere troppo con gli esempi è sufficiente, ma anche necessario, segnalare che in molti campi applicativi vengono posti problemi, sia numerici che non numerici, che possiamo qualificare come **problemi parzialmente determinati** per i quali non viene definito con completa precisione la prestazione richiesta, oppure problemi per i quali sono individuati solo intenzionalmente gli insiemi ambiente dai quali è consentito ricavare i dati e gli insiemi ambiente nei quali si possono trovare i risultati.
In questi casi non sono conosciuti con precisione il dominio e il codominio delle funzioni che corrispondono alle possibili trasformazioni effettuabili.

Per ogni algoritmo proponibile per risolvere uno di questi problemi è naturale che si cerchi di portare chiarezza alla corrispondente coppia $\langle$ dominio , codominio $\rangle$, in quanto in linea di principio solo essa permette di definire completamente l'algoritmo stesso e conseguentemente permette di precisare la concreta realizzabilità della corrispondente MSPTA e la sua conseguente affidabilità.

Possiamo quindi affermare che per molti problemi computazionali la completa precisazione degli algoritmi risolutivi risulta tutt'altro che agevole e viene trascurata confidando nel buon senso dei responsabili delle applicazioni.

B18:**a.05** La definizione precisa degli algoritmi risolutivi e della loro portata è sicuramente determinante in molte circostanze applicative.
Ad esempio questo è il caso delle elaborazioni numeriche che devono essere effettuate da processori incorporati in apparecchiature costose che devono essere inpiegate per obiettivi giudicati cruciali e strategici. Si pensi ad esempio agli algoritmi da implementare in una sonda da impiegare in una esplorazione spaziale

Nei casi più impegnativi una migliore precisazione di dominio e codominio potrebbe essere ricercata ricorrendo a un secondo algoritmo "supervisore" che proceda a simulare un algoritmo ragionevolmente proponibile per ottenere indicazioni empiriche sopra il complessivo ambiente nel quale si deve collocare il dominio per avere garanzie "sufficientemente" complete di funzionamento attendibile.

Un tale supervisore in certe circostanze potrebbe risultare dispendioso o richiedere tempi sperimentali eccessivamente elevati; in questi casi potrebbe essere preferibile procedere probabilisticamente effettuando prove su opportuni campioni, cioè effettuando indagini non del tutto sicure, ma molto meno costose delle osservazioni esaurienti.

In effetti per la precisazione di dominio e codominio di sottoprogrammi di interesse pratico, strumenti che vanno considerati come importanti prodotti industriali, ci si basa su prove di collaudo da definire con una accuratezza "sufficiente", sia sul piano tecnico che sul piano legale.

Osserviamo ora che negli studi con finalità generali le distinzioni che abbiamo segnalate costituiscono elementi che disturbano una presentazione dei fattori cruciali che possa essere comunicata efficacemente e possa essere utilizzata per l'avanzamento della strumentazione.

Dobbiamo anche riconoscere che delle funzioni dati-risultati associate ad algoritmi spesso non risulta determinante la individuazione precisa del dominio e del codominio. Vi sono molte situazioni nelle quali ci si limita a confidare nella adeguatezza di quanto si va elaborando, basandosi semplicemente su pratiche consolidate da esperienze precedenti.

Le precedenti considerazioni, in effetti, intendevano principalmente segnalare situazioni nelle quali emergono differenze tra le esigenze delle applicazioni specifiche e le esigenze della presentazione di argomentazioni con finalità generali.

B18:**a.06** Riprendiamo ora ad affrontare l'alternativa tra finitezza e illimitatezza degli insiemi dei vari oggetti (in particolare di stringhe, di numeri interi e di entità ottenute combinando queste due) che possono costituire importanti terreni dei dati da sottoporre a elaborazioni procedurali e quindi che possono costituire argomenti di funzioni concretamente valutabili.

Va osservato che in alcune considerazioni precedenti abbiamo assunto tacitamente la "illimitata estendibilità" degli ambienti nei quali collocare il dominio e il codominio di molte funzioni.

Un primo sostegno alla opportunità di una tale assunzione proviene dalla osservazione che, come si può facilmente constatare, le definizioni di molte operazioni vengono semplificate in misura notevole.

Un esempio concreto si trova nel calcolo della somma di due interi naturali.
Ricordiamo che su tutti i computers sono disponibili dispositivi che effettuano la somma di due interi a ciascuno dei quali sono dedicati registri di memoria di estensione fissa, in particolare registri di 16, 32, 64 o 128 bits.

Alternativamente sono disponibili sistemi che permettono di eseguire la somma di due interi di estensione praticamente illimitata: se vengono loro proposti addendi interi forniti da centinaia o migliaia di cifre dedicano a questi numeri interi sequenze di celle standard di memoria e si preoccupano di organizzare le manovre che consentono di ottenere la precisa somma richiesta.

Anche se in concreto si possono incontrano dei limiti pratici, si può invece procedere ad avvicinare la illimitatezza ideale: nei nostri tempi di perduranti miglioramenti tecnici si può confidare che una richiesta relativa a interi superiori di quelli trattati fino a un certo momento possa essere soddisfatta con nuovi potenziamenti dei dispositivi esecutori.

Questa possibilità non è affatto garantita in tutte le circostanze: non si può escludere che una specifica richiesta di questo genere ad un determinato sistema di calcolo si riveli impossibile da soddisfare, ed in particolare sia impossibile da soddisfare in tempi accettabili dai committenti.

Nell'ambito delle presentazioni di considerazioni generali risulta invece opportuno ipotizzare questa possibilità con la semplice riserva di abbandonarla in circostanze nelle quali si dimostra insostenibile. Infatti una tale ipotesi consente di presentare vari sviluppi di strumenti numerici in forme molto più concise e più semplici da comunicare e da proporre come condivisibili con riserva.

Secondo questo approccio qualificabile come "atteggiamento empirico" diventa accettabile supporre che la somma può effettuarsi su "qualsiasi" coppia di interi, e dichiarare sbrigativamente che la somma di due numeri interi risulta "sempre possibile", limitandosi a segnalare a bassa voce la dovuta riserva.

B18:**a.07** Andando oltre la somma, si riesce ad individuare una grande varietà di operazioni numeriche per le quali si può proporre l'ipotesi della sufficienza delle risorse disponibili da far cadere solo quando si manifesti una insufficienza di fronte alla quale tuttavia si possa contare sulla reperibilità di risorse maggiori e sulla costruibilità di macchine più prestanti e più versatili che consentano di superare la limitazione riscontrata.

Questo atteggiamento di fiducia nelle possibilità di continuare a progredire viene robustamente sostenuto dalla consapevolezza dei progressi che si sono riscontati nei calcoli numerici negli ultimi decenni.

Questo atteggiamento di fiducia si può estendere anche ad elaborazioni che non si limitano ad operare sopra i numeri interi. Considerazioni analoghe si possono svolgere per le elaborazioni sulle stringhe e di conseguenza per le elaborazioni su tutti gli oggetti, della matematica e delle sue applicazioni, che sappiamo potersi esprimere adeguatamente mediante stringhe.

L'atteggiamento di fiducia si puùlteriormente allargare a comprendere la disponibilità di macchine progressivamente più evolute e adattive che consentano modalità di utilizzo più agevoli e che, conseguentemente consentano presentazioni delle possibilità di risoluzione di problemi più compatte e scorrevoli.
Anche in questo gli avanzamenti del recente passato sono di aiuto.

Evidentemente le esperienze del passato ci fanno prevedere che la costruzione di queste macchine più evolute possa risultare più impegnativa e che la loro descrizione dettagliata possa risultare più elaborata.

Ad esempio si possono considerare le operazioni sopra numeri razionali (che possono essere portate avanti in modo del tutto preciso quando si possono trattare numeratori e denominatori senza vincoli di grandezza) e dopo di queste le elaborazioni sui numeri reali in grado di garantire risultati approssimati con una tolleranza inferiore ad una soglia solitamente concordata, ad esempio con tolleranza inferiore a 10^{-10}.

L'ipotesi della illimitatezza attualmente si può assumere anche per le elaborazioni sopra testi e archivi amministrativi, finanziari, storici, ... Infatti la capacità degli odierni sistemi di registrazione digitale è in grado di assicurare la disponibilità di basi dati senza restrizioni tecniche, in quanto le limitazioni maggiori provengono dalla capacità limitata di trasferire le informazioni dai supporti tradizionali su dispositivi digitali gestibili razionalmente.

In generale possiamo affermare che si possono ottenere semplificazioni di utilizzo e di definizione conoscitiva facendo riferimento a macchine capaci di ampliare le proprie risorse nel caso queste si rivelino necessarie.
Conviene ribadire che sul piano espositivo la possibilità di ricorrere a termini come "risorsa illimitatamente estendibile" consente di semplificare molte definizioni e molte presentazioni di sviluppi costruttivi.

Da queste considerazioni emerge ancora che le esigenze di generalità spesso si trovano a confliggere con le esigenze delle precisazioni che consentono di raggiungere l'affidabilità dei risultati che si vogliono conseguire.

B18:**a.08** Nel procedere di questa *esposizione* ci proponiamo di mostrare che le esperienze matematiche plurisecolari e la attuale rilevante e perdurante crescita delle elaborazioni e del patrimonio di procedure affidabili consolidate conduce in modo ineluttabile ad adottare formalizzazioni e astrazioni.

Le entità, il linguaggio e le finalità della tradizione matematica sono riconducibili a esigenze profondamente motivate da esigenze applicative ed i risultati che si sono raggiunti e che continuano a essere conseguiti danno ragione ai metodi e allo stile elaborato da questa disciplina.

La propensione alla formalizzazione e alla astrazione delle attività scientifiche e tecnologiche richiede molteplici approfondimenti; nel seguito cercheremo di farla emergere progressivamente e a questo proposito rinviamo in particolare a B60, B61 e B65.

Uno degli atteggiamenti che va preso in maggiore considerazione riguarda l'opportunità di ricorrere sistematicamente alla nozione di **infinito** , a partire dalla sua accezione di **infinito potenziale discreto**.

Come si è detto, quando si parla di calcoli scientifici e tecnologici preoccupandosi anche della effettiva realizzabilità si dovrebbero descrivere procedure in grado di generare sequenze lunghe quanto si vuole, su nastri estesi quanto serve, procedendo attraverso un numero di passi elevato quanto lo richiede ciascuna particolare circostanza.
Si può tuttavia constatare facilmente che se si continuasse a seguire questo criterio si dovrebbero usare termini e giri di parole che condurrebbero ad esposizioni lunghe, pesanti alla lettura, onerose da ricordare e da comunicare e in definitiva scarsamente efficaci.

Un'alternativa più conveniente consiste nel parlare di procedure che producono "sequenze illimitate di stringhe", registrandole su "nastri illimitati", operando attraverso "successioni illimitate di passi".
Anche questo modo di esprimersi tuttavia si rivela poco agevole e conviene cercare di semplificarlo ulteriormente.

B18:**a.09** Cerchiamo un modo di andare oltre il finitismo stretto per gli insiemi di oggetti di interesse matematico e per le elaborazioni volte alle soluzioni affidabili dei problemi in scenari di attività computazionali per scopi tangibili che si evolvono nel tempo.

A questo scopo esaminiamo un paio di attività computazionali con obiettivi tangibili: con A_1 denotiamo la elencazione di numeri primi, mentre con A_2 denotiamo l'esame dettagliato di un fenomeno fisico $\mathfrak{F}$ valutato da una grandezza statistica y che si considera dipendere dai valori di un parametro n esprimente il numero di campioni sui quali si effettuano misurazioni della grandezza y.

L'elencazione dei numeri primi, notoriamente non ottenibile con le valutazioni di una formula finita e chiusa, può essere ridotta al procedere nella registrazione su un nastro di numeri primi che si individuano con successive indagini.
L'esame del fenomeno $\mathfrak{F}$ implica che si continui a registrare coppie di valori assunti, risp., da n e da y.

Si può pensare a una prima fase dello studio A_1 nella quale basta produrre un elenco di numeri su carta e a una prima fase di A_2 nella quale basta segnare dei punti sopra un foglio di carta millimetrata.

Approfondendo gli studi, per A_1 si cerca di aumentare l'estensione dell'elenco e per A_2 si devono segnare i risultati su un foglio più ampio in quanto si procede a esaminare sempre nuovi campioni e diventano più definiti i valori di y e quindi occorre assumere come unità di misura per la y successivi sottomultipli delle unità adottate in precedenza.

Proseguendo ulteriormente gli studi si impone l'adozione di tecniche di documentazione più elaborate attraverso l'utilizzo di strumenti più tecnologici e più prestanti.
La descrizione sintetica dei risultati per A_1 potrà cercare di coprire intervalli di numeri interi della forma $\mathbf{[}2 : M\mathbf{]}$ con M sempre più grande, mentre per A_2 potrà inquadrare prodotti cartesiani della forma $\mathbf{[}1 : N\mathbf{]} \times \mathbf{[} -Y : Y\mathbf{]}$ per numeri di campioni che si avvicinano a N e per valori di y individuati con precisione via via maggiore.

La descrizione di questa continua crescita della documentazione dei risultati (che si può effettivamente riscontrare nelle cronache di molti settori di ricerca e sviluppo) evidentemente se prosegue a servirsi di un linguagio strettamente finitistico è destinata a restare ripetitiva nei toni e a diventare sempre più pesante nei dettagli.

Si cerca quindi di fare riferimento a entità che sono in grado di contenere tutte gli insiemi finiti che possano essere concretamente individuati con le attività che si vanno svolgendo.
In particolare i numeri primi che si vanno individuando si collocano entro l'insieme $\mathbb{P}$ di "tutti" gli interi positivi, mentre i risultati di A_2 si possono inquadrare nel prodotto cartesiano $\mathbb{N} \times \mathbb{Z}$.

B18:**a.10** Nelle prossime sezioni riprendendo le macchine MSP e le procedure associate ci proponiamo di giustificare un linguaggio idealizzato al quale si permette di parlare di entità costruite su "nastri infiniti" o "memorie infinite" di altro genere, da procedure che operano attraverso una "successione infinita" di passi e nel complesso quindi servendos di "risorse infinite".
In tal modo si evitano le molte precisazioni che sarebbero richieste prima di ciascuna delle elaborazioni volte alla soluzione effettiva di specifiche istanze di problemi basandosi sulla ipotesi di poterle giustificare per ogni situazione che si deve affrontare effettivamente. In tal modo si ottengono vantaggi rilevanti facilmente constatabili, sia nell'impostazione dei problemi, che nella descrizione dei procedimenti risolutivi, che nelle considerazioni generali sui risultati.

Con questo approccio si vuole giustificare su basi utilitaristiche alle persone interessate esclusivamente a risultati tangibili l'introduzione di classiche nozioni astratte della matematica, a cominciare dalla nozione di infinito potenziale.

La nozione astratta più semplice, abbiamo visto essere l'insieme delle rappresentazioni unadiche degli interi naturali.
Questa entità si può introdurre mediante una **macchina sequenziale programmabile generatrice** che può procedere quanto serve nella generazione delle sequenze di un simbolo specifico, tipicamente la tacca "|".
Questa è la prima delle varianti delle MSP che non prevede di trasformare dati forniti di volta in volta e si pone l'obiettivo di emettere su un nastro di uscita una sequenza di risultati che potrebbe crescere illimitatamente.

L'esecuzione di una MSPG potrebbe concludersi dopo un numero finito di passi oppure procedere illimitatamente: nel primo caso la qualifichiamo come MSPGF, nel secondo come MSPGI.
È evidente che le MSPGI sono automatismi con finalità e comportamenti e molto diversi; essi tuttavia hanno in comune la ampia libertà per le istruzioni che le governano.
Va subito segnalato che il problema di decidere se una data MSPG sia una MSPGF oppure una MSPGI potrebbe costituire un problema di non facile soluzione e che per talune MSPG ci si trova incapaci di risolverlo; in effetti questo problema va collocato tra i problemi attualmente irrisolti.

B18:b. macchine sequenziali programmabili generatrici

B18:**b.01** Le macchine sequenziali programmabili generatrici, in sigla MSPG sono macchine formali che come le MSPT sono costituite da una unità di controllo in grado di utilizzare propri registri per l'esecuzion delle istruzioni, e di servirsi di supporti di memoria ad accesso diretto (tramite indirizzi) e ad accesso sequenziale (nastri da gestire tramite testine di lettura e scrittura.

Le MSPG differiscono dalle MSPT solo per la organizzazione complessiva delle loro prestazioni.
Anche una MSPG è in grado di avviare una sua evoluzione a partire da un complesso di dati iniziali che caratterizza una istanza del problema da affrontare.
Lo scopo delle sue prestazioni, però, non è l'ottenimento da un complesso di risultati, ma è quello di procedere alla costruzione di successivi dati di uscita che possiamo limitarci a vedere come generiche stringhe che vengono emesse in successive fasi operative sopra un nastro di uscita caratteristico della macchina stessa.

Una evoluzione di una MSPG può consistere nella emissione di una lista finita di stringhe risultato e di arrestarsi, ma può anche procedere ad una emissione può giungere ad una fase operativa con la quale ha emessa una sequenza di stringhe risultato ed ha la possibilità di procedere ulteriormente senza che un gestore della evoluzione sia in grado di stabilire se l'evoluzione stessa proseguirà illimitatamente (continuando ad emettere oppure senza ulteriori emissioni) o all'opposto fino a giungere ad una successiva fase di arresto, di conclusione della sua evoluzione.

Ogni evoluzione di una MSPG si può descrivere come una successione di "fasi" operative ciascuna delle quali aut si conclude con la emissione sopra il nastro di uscita di una nuova stringa e con la predisposizione alla esecuzione di una nuova fase, aut risulta procedere senza aver emesso (ancora) nulla.
Le stringhe che vengono emesse si dicono **stringhe generate dalla MSPG**.

B18:**b.02** Per formalizzare la descrizione dei comportamenti delle macchine generatrici pensiamo a una a MSPG M che, in conseguenza della lettura di una stringa y che si è constatata appartenere al linguaggio dai suoi dati iniziali ammissibili, ossia legali, effettua una evoluzione nella quale si distinguono fasi successive, ciascuna delle quali, la i-esima, ha portato alla emissione sul suo nastro di uscita U di una stringa w_i.
Quindi dopo una sequenza di n fasi sul nastro U di uscita si trova una sequenza di n stringhe che denotiamo con w_0, w_1, w_2, ..., w_n e che per semplicità espositiva consideriamo diverse tra di loro (se cosìì non fosse basterebbe aggiungere alla macchina una manovra selettiva che interviene prima della emissione e si serve del nastro di emissione).

A questo punto si danno le tre possibilità:

- M si si arresta concludendo la generazione di una lista finita di stringhe generate.
- Lo studio del comportamento precedente e prevedibile della M da parte di un supervisore della evoluzione gli consente di stabilire che la evoluzione della M proseguirà illimitatamente.
- Il comportamento della M e le conoscenze che il supervisore è in grado di utilizzare non gli consente di stabilire se la evoluzione della macchina proseguirà illimitatamento o giungerà ad un arresto.

Le caratteristiche di una evoluzione, oltre che dalla stringa y che ha avviata l'evoluzione stessa, dipende dal complesso delle istruzioni che governano la sua evoluzione, ovvero da programma del quale la macchina è dotata.

Se si è raggiunto l'arresto o se si dimostra che si giungerà ad un arresto siamo di fronte a un processo evolutivo finito che si può ricondurre ad una evoluzione algoritmica di una MSPT: basta trattare la lista $\langle w_0, w_1, w_2, ..., W_n \rangle$ come il risulatato della sua evoluzione.
Per queste situazioni potrebbe porsi la opportunità di distinguere se l'insieme delle stringhe generate costituisce un insieme esplicito facilmente trattabile, fa prevedere un insieme esplicitabile o addirittura un insieme finito ma intrattabile.

Se non si giunge ad un arresto e neppure si prevede una sua conclusione si pone quello che chiamiamo **problema del prodotto dell'evoluzione non finita** (in breve "problema-PENF") caratterizzato dalla MSPG M e dalla stringa iniziale y.
Nel seguito per ragioni pratiche trascuriamo la dipendenza da y, in quanto gran parte delle MSPG che rivestono interesse si evolvono senza partire da una specifica stringa iniziale; in alternativa si può inserire la y nella stessamacchin generatrice.

B18:**b.03** Per quanto riguarda il problema-PENF occorre iniziare con una presentazione della casistica per grandi linee.

Si hanno innanzi tutto macchine MSPG per le quali si dimostra che l'evoluzione non avrà fine, non potrà mai giungere a una configurazione di arresto, ma procederà con sempre nuove emissioni di stringhe che denotiamo con w_{n+1}, w_{n+2},...,w_{n+k},... .

Per alcune di queste macchine le stringhe emesse sono tutte diverse tra di loro; per altre accade che a un certo punto iniziano a replicarsi ciclicamente e per la loro sequenza si può scrivere:

$$w_m,\ w_{m+1},\ \ldots\ ,w_{m+p-1},\ w_{m+p} = w_m,\ w_{m+p+1} = w_{N+1},\ldots,\ w_{m+2\,p} = w_m,\ldots\ .$$

Accade tuttavia che si possono evitare tutte le ripetizioni tra le stringhe emesse da una MSPG arricchendo tale macchina con istruzioni che scorrono le stringhe precedentemente generate ed evitano di emettere stringhe ripetute.

Quindi l'evoluzione di una macchina che si viene a trovare in un ciclo di configurazioni che si ripetono ilimitatamente e quindi generano sequenze di stringhe che si ripetono e può essere interrotta.

Una macchina con un comportamento di questo genere ha la stessa portata di una macchina che si evolve per un numero finito di passi e genera una lista finita che può essere resa nonripetitiva.

Prendiamo poi in esame le MSPG che a un certo punto della loro evoluzione dopo aver generato le prime n stringhe, non danno indicazioni sulla loro possibile evoluzione successiva.
In una situazione di questo genere si può far proseguire l'evoluzione per un certo altro numero di fasi con la possibilità di giungere a un chiarimento, ma anche con la possibilità di ritrovarsi ancora in una situazione di incertezza.

Ogni proseguimento della evoluzione può rendere disponibili nuove stringhe generate ma richiede di consumare risorse (in termini di tempo, di memorie o di strumenti di supporto) e la decisione se procedere o meno oltre una situazione di incertezza dipende totalmente da come si valutano le circostanze concrete entro le quali si colloca la supervisione dell'evoluzione della MSPG in causa.

Si possono quindi prospettare diversi sottoscenari.

(1) Mancano le risorse e si interrompe.

(2) Si cercano nuove risorse, si trovano e si prosegue.

(3) Ci si limita ad aspettare che nuove risorse possano essere fornite dalla evoluzione della tecnologia e dal crescere dell'interesse nei confronti delle nuove stringhe che potranno essere generate.

(4) Ci si limita a dichiarare la possibilità di generazione illimitata, rinunciando ad accertare quali stringhe potrebbero essere effettivamente emesse.

Queste dichiarazioni di possibili generazioni illimitate possono essere giustificate solo come scelta metodologica nei confronti della redazione di considerazioni rivolte a fini generali sopra le proprietà delle MSPG .

Va inoltre rilevato che parlare in termini di effettiva disponibilità di una infinità di stringhe generate è assurdo sul piano delle osservazioni fisiche.

B18:**b.04** A questo punto siamo in grado di proporre un nuovo genere di entità, quello degli insiemi generati da MSPG che si dimostrano essere in grado di procedere illimitatamente alla generazione di stringhe sempre diverse dalle precedenti.

Queste entità vengono chiamate **insiemi-GI** e le macchine in grado di procedere a generarli le abbiamo denotate con l'acronimo MSPGI. [B18a10]
Si deve osservare che ciascuno di questi insiemi viene individuato prescindendo completamente dalla disponibilità delle risorse che la macchina generatrice effettivamente consuma e ignorando del tutto l'eventualità che essa incontri ostacoli alla sua evoluzione.

Per quanto riguarda la validità della dimostrazione della illimitatezza della evoluzione ci limitiamo ad invocare, ancora una volta, la ampia condivisibilità basata sulla semplicità degli esempi che si possono presentare a suo favore.

Ad una macchina MSPGI **M** risultano associate le liste delle stringhe che essa ha generato nel corso di successive fasi evolutive.

In particolare parliamo di una lista attuale ottenuta dopo una sequenza di n_1 fasi che scriviamo

$$L_{n_1} \;=\; \langle w_0, w_1, w_2, ..., w_{n_1} \rangle \;.$$

La illimitatezza della sua evoluzione, da dimostrare, comporta la possibilità realistica che essa, impegnando altre risorse possa proseguire a generare altre stringhe fino a ottenere una lista

$$L_{n_2} \;=\; \langle w_0, w_1, w_2, ..., w_{n_2} \rangle \;\text{ con } n_2 > n_1 \;.$$

Questo possibile ampliamento si può dichiarare ulteriormente estendibile, quante volte si vogliono.

B18:**b.05** L'insieme-GI generato dalla MSPGI **M** lo denotiamo con il simbolo $\mathbf{M}^{\mathcal{G}}$ e ad esso vanno attribuite come elementi tutte le stringhe w_j per la quali si possa constatare che compaiono in una delle liste L_r che la **M** si può enunciare in grado di generare.

Per ciascuna di tali stringhe w_i possiamo scrivere $w_i \in \mathbf{M}^{\mathcal{G}}$, notazione che, similmente a quanto fatto per gli insiemi finiti e per gli insiemi-E leggiamo

$$w_i \text{ è un elemento che appartiene all'insieme-G } \mathbf{M}^{\mathcal{G}} \;.$$

Ciascuno degli insiemi-G si può considerare un insieme-E, ovviamente, e un insieme-P, la proprietà che caratterizza i suoi elementi essendo la loro comparsa in una delle liste che la **M** è in grado di generare.

Equivalentemente invece che di stringa generata si può parlare di stringa che la macchina emette sul suo nastro di uscita.

Si osserva che la idealizzazione della evoluzione illimitata della MSPGI deve presumere che il nastro di uscita della macchina sia illimitatamente estendibile.

Non sarebbe invece realistico parlare di macchina dotata di un nastro di uscita illimitatamente esteso e tanto meno di nastro infinito.

Uno scenario puramente fantasioso è quello che vede la macchina dopo aver concluso la sua evoluzione di infiniti passi; questo evidentemente richiede che essa abbia consumato quantità infinite di risorse e sul suo nastro di uscita abbia registrate tutte le infinite stringhe che ha generate.

Queste descrizioni lontane dalla realtà sensatamente concepibile sono da considerare modi di dire immaginifici e metaforici che presentano il vantaggio di consentire enunciazioni delle proprietà che coinvolgono degli insiemi-G di notevole concisione.

Bisogna forse ammettere che per taluni discorsi risulta di qualche efficacia un linguaggio che presume scenari privi di giustificazioni fisiche.

In effetti è diffusa l'abitudine di parlare di stringhe facenti parte di linguaggi illimitatamente generabili e di insiemi potenzialmente infiniti; questi discorsi sono accettabili se sottintendono la possibilità di riformulare i relativi enunciati con discorsi più realistici ma che, purtroppo, risultano sensibilmente più elaborati e meno leggibili, ossia che presentano precisi svantaggi sul piano della organizzazione e della comunicazione delle conoscenze.

B18:**b.06** Per gli insiemi-G, in quanto particolari insiemi-P, si può porre la questione se possono essere considerati più specificamente degli insiemi-B.
Essi presentano il vantaggio di potersi collegare alla collezione delle macchine sequenziali programmabili, cioè alle macchine che si congettura siano in grado di effettuare tutte le elaborazioni procedurali.

Risulta conveniente presentare una consistente gamma di esempi significativi di insiemi-G con le corrispondenti MSPGI avviando in tal modo una analisi costruttiva delle loro proprietà.

Va subito segnalato che per gli insiemi-G risulta possibile definire costruzioni, operazioni e relazioni assimilabili a quelle viste per gli insiemi espliciti.
Va anche detto che lo studio degli insiemi-G presenta vari punti critici che richiedono di procedere con cautela: la cosa non può meravigliare, in quanto il loro genere è collegato a una idealizzazione che richiede la adozione di molto senso critico.
Dovremo quindi procedere individuando vari elementi di distinzione che conviene vedere come riguardanti una classificazione dei sottoinsiemi della totalità degli insiemi-G, totalità che denoteremo con **SetG**, simboli che possiamo qualificare come insieme-P.

B18:**b.07** I primi insiemi-G che conviene introdurre sono quelli che [B08f06] abbiamo chiamati **ambienti primari**.
Il più semplice di questi è l'insieme delle rappresentazioni unadiche degli interi naturali [B04a02]. Descriviamo quindi una MSPGI che può procedere quanto serve nella generazione delle sequenze di un simbolo di base, tipicamente la tacca "ǀ".

È semplice immaginare che una tale macchina riesca a emettere le rappresentazioni unadiche di un certo numero n di numeri naturali opportunamente separati da un segno separatore che scegliamo essere ",".
Nel caso $n = 6$, essa presenta all'inizio del nastro di uscita una stringa come la seguente:

$$\vdash , \text{ǀ}, \text{ǀǀ}, \text{ǀǀǀ}, \text{ǀǀǀǀ}, \text{ǀǀǀǀǀ} \dashv$$

Nella fase successiva essa riproduce nelle caselle del nastro di emissione alla destra dell'ultimo segno "ǀ" emesso i caratteri compresi tra l'ultima occorrenza del separatore "," e il segno $\dashv$ e a essi accoda una ulteriore "ǀ" e una replica del segno $\dashv$ in sostituzione di quella trascurata ottenendo

$$\vdash , \text{ǀ}, \text{ǀǀ}, \text{ǀǀǀ}, \text{ǀǀǀǀ}, \text{ǀǀǀǀǀ}, \text{ǀǀǀǀǀǀ} \dashv$$

cioè una rappresentazione dei primi $n + 1 = 7$ numeri naturali.

Agendo similmente, cioè sotto il controllo dello stesso sistema finito di istruzioni, nell'ipotesi di disporre di un nastro di adeguata lunghezza si può tranquillamente pensare di avere generata con una MSPGF la sequenza degli interi da 0 a $2 \cdot 10^9$; per questo basta un nastro di $2\,10^{18} + 10^9$ caselle, ovvero un sistema di poco più di due milioni di dispositivi di memoria da 1 Terabyte ciascuno (ormai familiari).

Volendo ampliare la sequenza degli interi concretamente disponibili sarebbe necessario modificare la macchina MSPGF rendendole disponibili più memorie e rendendola in grado di controllare questo nuovo dispositivo di registrazione. Anche questo potenziamento in linea di principio si può pensare fattibile.

Nella pratica questo ampliamento ed eventuali ulteriori successivi si rendono necessari quando si vogliono affrontare problemi più estesi dei precedenti; ad esempio quando si vogliono elencare tutti i numeri primi inferiori a $2 \cdot 10^9$. Ma questi sono solo 98 milioni e sono stati resi disponibili senza ricorrere all'intero intervallo $(1 : 2\,10^9)$ [e.g. `Lists of prime numbers (we)`].

Si giunge anche a pensare che si risparmia la memoria e il tempo di calcolo richiesti dall'effettivo elenco di un insieme della forma $(N]$ investendo in una migliore analisi delle proprietà di tale insieme e delle procedure che consentono di ottenere suoi sottoinsiemi o sue proprietà.

Questo porta a diminuire l'importanza di disporre di elenchi effettivi per questi insiemi finiti di grandissima ampiezza.
Non è indispensabile disporre di questi grandissimi insiemi di notazioni di numeri interi per stabilire se una stringa esprime una tale notazione, cioè un intero, e neppure per estrarre un suo ben definito sottoinsieme di estensione contenuta.
La possibilità di ampliare un tale insieme dovrebbe essere verificata solo quando se ne presentasse la concreta sostanziale necessità.

B18:**b.08** Descriviamo ora una seconda MSPGI che consente di procedere nella generazione di "tutte" le stringhe sopra un dato alfabeto finito A, cioè dell'insieme-G e viene denotato con A^*.

Dato che a questa macchina si chiede una portata maggiore di quella descritta in b07, ci si aspetta che si tratti di una apparecchiatura con prestazioni più diversificate e impegnative.

Vediamo ad esempio, le fasi della procedura per la generazione delle stringhe sull'alfabeto $\{\mathtt{a}, \mathtt{b}, \mathtt{c}\}$.
Per descriverla occorre trattare l'alfabeto come ordinato, cosa che è resa effettiva dalla semplice registrazione dei suoi caratteri sopra un apposito nastro predisposto per il mero confronto delle posizioni occupate dai caratteri; esprimiamo l'ordinamento scelto scrivendo $\{\mathtt{a} < \mathtt{b} < \mathtt{c}\}$.

Ricordiamo poi due manovre: quella che effettua il confronto lessicografico [B06c04] tra due stringhe della stessa lunghezza e quella che trasforma una stringa in quella della stessa lunghezza immediatamente successiva secondo l'ordine lessicografico [B08a02].

Diciamo che le stringhe vengano generate per successivi lotti, ciascuno dei quali costituito dalle stringhe caratterizzate da una delle successive possibili lunghezze; vogliamo inoltre che le stringhe di una data lunghezza (di un lotto) vengano generate seguendo l'ordine lessicografico.

Supponiamo che dopo una certa fase sul nastro di emissione si trovi la sequenza di stringhe (iniziante con la rappresentazione della stringa muta):

$$\vdash, \mathtt{a}, \mathtt{b}, \mathtt{c}, \mathtt{aa}, \mathtt{ab}, \mathtt{ac}, \mathtt{ba}, \mathtt{bb}, \mathtt{bc}, \mathtt{ca}, \mathtt{cb}, \mathtt{cc}, \mathtt{aaa}, \mathtt{aab} \dashv$$

Nella fase successiva si ricava dalla `aab` la successiva in ordine lessicografico e di lunghezza 3, cioè la `aac`; questa la si accoda al nastro preceduta dal separatore e seguita dal terminatore ottenendo:

$$\vdash \texttt{,a,b,c,aa,ab,ac,ba,bb,bc,ca,cb,cc,aaa,aab,aac} \dashv \quad ,$$

Si procede con fasi simili fino ad avere alla fine del nastro l'ultima stringa di lunghezza 3, cioè avendo sul nastro:

$$\vdash \texttt{,a,b,c,aa,ab,ac,ba,bb,bc,ca,cb,cc,aaa,...,ccb,ccc} \dashv \ .$$

A questo punto l'ultima stringa viene fatta seguire dal separatore e dalla `aaaa` , la prima delle stringhe la cui lunghezza supera di 1 quella della precedente.
In generale alla stringa della forma $\texttt{c}^s$ si fa seguire la $\texttt{a}^{s+1}$, operazione che si può ottenere senza difficoltà, ad esempio utilizzando la rappresentazione unadica della s.

Procedure analoghe consentono di generare le successive stringhe sopra un alfabeto qualsiasi seguendo un ordinamento lunghezza-lesssicografico, relazione che abbrevieremo con il termine **ordinamento-llx** .

B18:**b.09** Consideriamo la sequenza illimitata costituita dalle scritture binarie dei successivi interi naturali:

$$\texttt{0, 1, 10, 11, 100, 101, 110, 111, 1 000, 1 001, 1 010, 1 011, 1 100, 1 101, 1 110, 1 111, 10 000,} \ldots \ .$$

Si osserva che questa sequenza si può ottenere dalla procedura che genera nell'ordinamento-llx le stringhe sull'alfabeto $\{\texttt{0},\texttt{1}\}$ aggiungendole un dispositivo selettivo che consente l'effettiva emissione della stringa "`0`" e di tutte le stringhe che iniziano con la cifra `1`.

Si osserva che anche che i membri della sequenza precedente segue l'ordinamento-llx.
Più in generale si osserva che ogni sequenza di stringhe ottenuta con una riduzione da una sequenza di stringhe che osserva un ordinamento-llx (o qualsiasi altro ordine totale verificabile algoritmicamente, cioè basato sopra una routine di confronto tra stringhe facilmente precisabile) mantiene questo ordinamento.

(1) Eserc. Descrivere varianti della procedura precedente che consentono di generare le scritture posizionali degli interi positivi nelle basi 3, 4 e 10.

B18:**b.10** Si possono individuare molte altre sequenze di grande lunghezza arricchendo con opportuni meccanismi di selezione le MSPGI che procedono alla generazione di tutte le stringhe sopra un alfabeto ordinato.
In particolare con opportuni meccanismi selettivi si possono ottenere varie sequenze di numeri naturali che soddisfano proprietà specifiche che possono rivelarsi utili ad applicazioni facilmente immaginabili:
numeri naturali pari, numeri naturali divisibili per $d = 3, 4, 5, 11, ...$, numeri naturali palindromi, numeri positivi forniti da sequenze di cifre crescenti, ...

Ad esempio si possono generare le scritture decimali degli interi positivi divisibili per 3 o per 4 procedendo alla generazione di tutte le stringhe costituite da cifre decimali che non iniziano con 0 a applicando a ciascuna di esse un algoritmo che in un numero finito di passi stabilisce se la stringa soddisfa la richiesta di divisibilità che vedremo in B26.

Per la divisibilità per 3 si tratta di sommare le cifre e ottenere un intero inferiore che si vuole divisibile per 3; se il risultato m non è inferiore di 10 si effettua un'altra manovra consistente nella somma delle cifre di m; procedendo con queste manovre dopo un numero finito di passi si giunge a un intero compreso tra 1 e 9. Se tale intero coincide con 3, 6 o 9 è garantita la divisibilità per 3, se vale 1, 2, 4, 5, 7 o 8 è garantita la nondivisibilità per 3.

Per la divisibilità per 4 si richiede innanzi tutto che l'ultima cifra sia pari; in tal caso se essa è uguale a 0, 4 o 8 e la penultima cifra è pari oppure se l'ultima è uguale a 2 o 6 e la penultima è dispari è garantita la proprietà richiesta; negli altri casi il numero dato non è divisibile per 4.

Come conclusione di ogni fase di generazione di una nuova scrittura decimale seguita dalla manovra di selezione, se il requisito è soddisfatto si emette la notazione trovata sul nastro di uscita, in caso contrario la nuova scrittura viene trascurata.

(1) Eserc. Definire una procedura che genera secondo l'ordinamento-llx le stringhe dei caratteri `a` e `b` che in ciascuno dei suoi prefissi presenta il numero di occorrenze di `a` non inferiore al numero di occorrenze di `b`.

Più esplicitamente chiediamo di generare la successione delle stringhe

`a`, `aa`, `ab`, `aaa`, `aab`, `aba`, `aaaa`, `aaab`, `aaba`, `aabb`, `abaa`, `abab`, `aaaaa`, ...

B18:**b.11** Conviene osservare che due sequenze illimitate come le notazioni decimali degli interi positivi divisibili per 3 o per 4 possono essere generate più efficientemente da MSPGI che non generano stringhe che successivamente possono essere trascurate: ad esempio ogni sequenza illimitata di interi divisibili per un intero $d \geq 2$ si può generare con una macchina in grado di effettuare l'incremento di d di un qualsiasi intero naturale e di emettere ogni nuovo intero ottenuto in tal modo.

Lo schema organizzativo delle MSPG che con una manovra facile o comunque già nota procedono a generare successive stringhe e sottopongono ciascuna di esse ad un vaglio selettivo è uno schema per la generazione di sequenze specifiche (finite o illimitate) ampiamente adottato in quanto permette di ottenere tali sequenze con poca fatica del programmatore, anche se con elevato impegno per la macchina esecutrice.

Accade però che per qualche macchina che segue il suddetto schema organizzativo si riesca ad individuare un'altra macchina che genera la stessa sequenza di stringhe in modo sensibilmente più efficiente, cioè un'altra macchina equivalente più performante.

Conviene sottolineare questa situazione generale: tutte le MSPG, come tutte le MSP, ovvero come tutte le procedure, possiedono numerose varianti equivalenti e allontanandosi dagli schemi organizzativi generali, ossia limitando la gamma delle situazioni che si intendono controllare, si possono definire macchine più efficienti; queste tendenzialmente sono più articolate e in genere nell'evoluzione degli studi vengono precisate in momenti successivi; per molte coppie di tali varianti entrambe le procedure risultano vantaggiose in relazioe a diversi tipi di valutazioni.

B18:c. liste.G e insiemi-G

B18:**c.01** Ci proponiamo ora di giungere a una classificazione per grandi linee dei possibili comportamenti delle MSPG.

Ricordiamo che data una lista (finita) $L = \langle w_1, w_2, ..., w_n \rangle$, per ogni intero $i \in \{1, 2, ..., n\}$ si dice che $\langle w, i \rangle$ è la occorrenza i-esima entro la L.

Ricordiamo anche che da ogni lista L sappiamo eliminare i membri ripetuti, sappiamo ricavare l'insieme delle sue sottosequenze e quindi l'insieme dei componenti della L, insieme che denotiamo con $\mathsf{SetY}(L)$.

Consideriamo **M**, generica MSPG con il compito di emettere sul suo unico nastro di uscita, che denotiamo con U_{M}, stringhe sopra il suo alfabeto di uscita che denotiamo con A_{M}.
Per semplicità consideriamo che la **M** possa avere una unica evoluzione.

Ricordiamo che per **fase della evoluzione della macchina M** intendiamo la sottosequenza dei suoi passi che inizia nella sua configurazione iniziale o nella configurazione che segue un passo con il quale la macchina ha effettuata l'emissione di una stringa generata e si conclude con il passo nel quale ha concluso l'emissione di una nuova stringa risultato oppure ha effettuato il proprio arresto.

Analizziamo la situazione raggiunta della macchina **M** dopo che ha eseguiti m fasi della sua evoluzione giungendo nella sua configurazione che denotiamo con $\boldsymbol{C}_m$ e poniamoci il corrisponente problema-PENF [b02].

Può accadere che (1) l'evoluzione prosegua fino a concludersi con dell'istruzione di arresto, oppure (2) che prosegua con una sequenza di passi successivi che non conducono ad alcuna ulteriore emissione.

Nel caso (1), che denotiamo con (MG1) la macchina viene detta macchina sequenziale programmabile generatrice con evoluzione finita, in sigla **MSPGF**, e si dice che essa ha emesso la lista delle stringhe risultato che si denota con $\mathsf{M}^{\mathcal{L}}$. Inoltre per l'insieme delle stringhe che essa ha generate scriviamo $\mathsf{M}^{\mathcal{G}} := \mathsf{SetY}\left(\mathsf{M}^{\mathcal{L}}\right)$.

Nel caso (2) si possono avere tante scelte successive che attribuiamo a un supervisore della sua evoluzione il quale deve riferire ai committenti della elaborazione in corso.
Può accadere che il supervisore decida di proseguire l'elaborazione, oppure di effettuare un riesame dei possibili comportamenti successivi della **M** alla luce dei risultati ottenuti per cercare di individuare quali potranno essere le successive azioni della macchina.

Il proseguimento dell'evoluzione, ovviamente, richiede un ulteriore consumo di risorse e il supervisore deve deciderlo dopo aver valutato quanto può essere conveniente dedicare nuove risorse ai possibili nuovi risultati: dobbiamo dire, genericamente, che una decisione in proposito dipende da specifici elementi che il supervisore sa analizzare.

B18:**c.02** Il proseguimento della evoluzione della **M** con una secodo lotto di fasi e con il successivo esame dei nuovi risultati possono portare ad una configurazione di arresto e quindi a stabilire che **M** è una MSPGF con le conclusioni viste sopra.

In alternativa il nuovo lotto di fasi porta a un chiarimento delle previsioni con varie possibili prospettive; può invece accadere che rimangano sul tavolo tutte le precedenti incertezze o almeno una loro parte consistente.

Vediamo una classificazione che cerca di essere esauriente dei possibili scenari ai quali porta un attento esame che pensiamo a carico del supervisore.

(MG1a) Si accerta che l'evoluzione si concluderà con un futuro arresto e questo comporta la enunciazione che la **M** è una MSPGF e che l'evoluzione potrà proseguire fino alla sua conclusione; questo proseguimento dovrà essere deciso in base a una valutazione dei costi e dei benefici attesi.

(MG2) Si accerta che l'evoluzione proseguirà illimitatamente con emissione di sempre nuove stringhe risultato e quindi si assegna la **M** alla categoria delle macchine sequenziali programmabili generatrici che emettono illimitatamente nuove stringhe, in breve che si tratti di una MSPGI.

(MG3) Si accerta solo che l'evoluzione potrebbe procedere illimitatamente ma attraverso lotti di fasi con emissione di stringhe appartenenti solo a un insieme circoscritto, finito. In particolare si potrebbe accertare che l'evoluzione proseguirà illimitatamentema con sequenze di fasi che ciclicamente riguardano la stessa sequenza di configurazioni e quindi l'emissione di sequenze di stringhe uguali alle precedenti. In questo caso, lasciando la eventualità di altre incertezze ai due casi indicati di seguito, è possibile riuscire ad individuare l'insieme finito delle stringhe che **M** è in grado di emettere e la sequenza di passi, configurazioni e fasi che la macchina sarebbe destinata a ripetere ciclicamente.
Si può inoltre giudicare possibile la modifica della stessa **M** con l'aggiunta di opportune prestazioni di controllo per ottenere una macchina in grado di generare la lista finita prevista e di decidere successivamente di arrestarsi; in tale situazione la **M** può essere sostituita da una macchina equivalente del genere MSPGF.
Questa situazione conveniamo di caratterizzarla con il contrassegno (MG1b) per evidenziare la sua vicinanza al comportamento (MG1a).

(MG4) Si riesce ad accertare che l'evoluzione ha la possibilità di proseguire illimitatamente, ma non si dispone di alcuna garanzia che le nuove stringhe emesse risultino illimitatamente diverse da quelle emesse in precedenza, cioè si lascia aperto il dubbio tra l'emissione di una lista finita e il procedere illimitato di una lista in grado di rappresentare un insieme-G infinito, in breve un insieme-GI.

(MG5) Si procede con una nuova sequenza di passi con la quale aut si ottiene l'emissione di nuove stringhe, aut non si riscontra alcuna nuova emissione; in entrambi i casi si lascia il supervisore nella incertezza sopra l'esito che potrebbe avere la decisione di effettuare una ulteriore sequenza di passi evolutivi.

B18:**c.03** Dato che si sa decidere se una stringa che sta per essere emessa da una MSPG **M** compare nella lista delle stringhe emesse nelle fasi precedenti della elaborazione generativa, ciascuna macchina sequenziale programmabile generativa può essere modificata con l'aggiunta di una manovra di scorrimento delle stringhe emesse al fine di garantire che sul nastro di uscita vengano scritte solo stringhe non ripetute.

Si può quindi pensare che tutte le MSPG da esaminare per considerazioni generali (che prescindano dall'efficienza) siano **macchine sequenziali programmabili generatrici nonripetitive**, in sigla MSPGn

Confrontiamo l'evoluzione E_1 di una MSPG M_1 che emette stringhe ripetute con l'evoluzione E_2 della sua trasformata nonripetitiva M_2, nel corso della E_2 si aggiungono alle azioni della E_1 le manovre di controllo che precedono ogni emissione ed evitano le emissioni di stringhe replicate. Quindi a qualche fase della E_2 corrispondono più fasi della E_1, diverse dalla prima, le quali riguardano l'emissione di repliche di stringhe emesse in precedenza.

Le macchine MSPGn che hanno una evoluzione illimitata possono essere del tipo (MG2), cioè in grado di proseguire illimitatamente a emettere nuove stringhe: in tal caso le chiamiamo MSPGnI.

Si osserva che riducendosi alle sole macchine nonripetitive alcune delle situazioni (MG3) si possono ridurre a situazioni (MG1); in particolare le evoluzioni cicliche di una macchina M_1 possono essere

tenute sotto controllo per individuare una macchina con evoluzione che si conclude con l'arresto dopo aver emesso le stesse stringhe ottenibili dalla M_1.

Rimangono tuttavia le possibili situazioni di incertezza sui risultati che potranno essere ottenuti con un eventuale proseguimento dell'evoluzione successiva all'ottenimento dalla emissione della lista finita (nonripetitiva).
Dopo un certo numero di fasi evolutive e dopo una analisi dei risultati può permanere il dubbio se si possa proseguire fino a un arresto, se si potranno generare illimitatamente nuove stringhe, se si ha la possibilità di proseguire illimitatamente senza nuove emissioni.

Nel prosieguo ci occuperemo prevalentemente delle MSPGnI.

B18:**c.04** Consideriamo una MSPG M e usiamo l'espressione $\mathsf{M}^{\mathcal{L}}(t)$ per la lista dei membri che la macchina, dopo t fasi della sua evoluzione, ha emesso sul nastro di uscita U_{M}, inizialmente vuoto.
Ricordiamo inoltre che con $\mathsf{M}^{\mathcal{G}}(t)$ denotiamo l'insieme delle stringhe rappresentato dalla suddetta lista, ovvero la relazione $\mathsf{M}^{\mathcal{G}}(t) := \mathsf{SetY}\left(\mathsf{M}^{\mathcal{L}}(t)\right)$.
Chiaramente alla conclusione di ogni fase t si dispone di un insieme esplicito di stringhe generate.
Nel caso di macchina generatrici nonripetitive il numero dei membri della lista $\mathsf{M}^{\mathcal{L}}(t)$ è uguale al cardinale del corrispondente insieme, $|\mathsf{M}^{\mathcal{G}}(t)|$, cioè a t.

A ogni MSPG M si può associare il linguaggio $\mathsf{M}^{\mathcal{G}}$, che può essere definito come la totalità delle stringhe che occorrono in almeno una delle liste $\mathsf{M}^{\mathcal{L}}(t)$, ossia come la totalità delle stringhe che appartengono ad almeno uno degli insiemi espliciti $\mathsf{M}^{\mathcal{G}}(t)$.

L'appartenenza degli elementi di $\mathsf{M}^{\mathcal{G}}$ a uno degli insiemi $\mathsf{M}^{\mathcal{G}}(t)$ è la proprietà che fa dello stesso $\mathsf{M}^{\mathcal{G}}$ un insieme-P; questa proprietà tra gli elementi dell'ambiente $\mathsf{A}_{\mathsf{M}}{}^{*}$, che è un insieme-G, è goduta solo dalle stringhe di $\mathsf{M}^{\mathcal{G}}$.

Si osserva che se una stringa w appartiene a un insieme $\mathsf{M}^{\mathcal{G}}(t)$ per ciascuno dei $\bar{t} > t$ appartiene sicuramente all'insieme $\mathsf{M}^{\mathcal{G}}(\bar{t})$.

A questo proposito introduciamo la nozione di **successione nondecrescente di insiemi finiti**; si tratta di una successione di insiemi finiti $\langle S_1, S_2, ...S_n, ...\rangle$ tale che ogni componente S_i è sottoinsieme del successivo S_{i+1}.
Se tutti i componenti sono contenuti in senso stretto nei successivi si parla più specificamente di **successione crescente di insiemi finiti**.

Ad ogni successione nondecrescente si può associare un cosiddetto **insieme limite**, insieme-P costituito dalla totalità degli oggetti che sono elementi di almeno uno degli insiemi finiti suoi componenti.

Per ogni MSPGI M gli insiemi finiti $\mathsf{M}^{\mathcal{G}}(t)$ che essa sarebbe in grado di emettere costituisono una successione nondecrescente e se in particolate M è nonripetitiva si ha successione crescente in senso stretto.

L'insieme $\mathsf{L} := \mathsf{M}^{\mathcal{G}}(i)$ delle stringhe emesse dalla M si può considerare insieme limite di una successione nondecrescente di insiemi finiti.

Per ogni successione crescente illimitata l'insieme limite L è un insieme infinito; infatti per ogni intero positivo i l'elemento generato nella fase i si può porre in biiezione con l'elemento generato nella fase $i+1$ e in tal modo L si pone in biiezione con una sua parte.

B18:**c.05** Esaminiamo le sequenza illimitate generate da MSPGnI cominciando da esempi semplici e fondamentali.

Abbiamo visto il comportamento di una MSP che denotiamo con $\mathsf{M}_{\mathbb{N}}$ in grado di procedere a generare le stringhe che costituiscono le notazioni unadiche degli interi naturali.

Dopo l'emissione della rappresentazione dello zero $\langle\rangle$, essa organizza la successione delle fasi nelle quali una sequenza di repliche del segno | viene riprodotta e ampliata con un ulteriore occorrenza dello stesso segno.

Ciascuna di queste fasi si serve di un nastro ausiliario nel quale si mantiene una copia della precedente stringa aggiunta e prevede l'aggiunta a tale stringa di un ulteriore | e una iterazione per copiare la nuova stringa ausiliaria sul nastro di uscita.

Si osserva che la estensione della generazione può essere concretamente portata avanti fino a che risultano sufficienti i due nastri in progressiva crescita, il nastro di uscita (limite realistico vincolante per ogni lista emessa da una MSPG) e il meno impegnativo nastro ausiliario.

Se si volessero le rappresentazioni unadiche di ulteriori interi si dovrebbero estendere i nastri disponibili e la possibilità di questo dipende dalle circostanze in cui si volesse rendere effettivamente disponibile un elenco di interi naturali perché costituisca l'ambiente nel quale collocare ipotetici studi e ipotetiche costruzioni che richiedano numeri arbitratriamente illimitati di queste notazioni numeriche.

Considerazioni simili si possono svolgere per l'ambiente nel quale si possono collocare le costruzioni e gli studi sopra le stringhe su un qualsiasi alfabeto ordinato A fornito da una stringa della forma $\langle abc...z\rangle$

Denotiamo con M_{A} la MSPGI incaricata di questa generazione illimitata.
Essa, dopo aver generato il membro stringa muta, deve organizzare più livelli di iterazioni.
L'iterazione primaria si preoccupa di attuare lotti di fasi concernenti la generazione delle stringhe delle lunghezze successive $s = 1, 2, 3, ...$
Nell's-esimo di questi lotti si organizza la emissione in ordine lessicografico delle stringhe di lunghezza s, cosa che si ottiene con una iterazione di emissione della stringa a^s, la prima di A^s e con una iterazione di trasformazione di una stringa di lunghezza s nella successiva; questa manovra è stata descritta in B08a02.

Anche per una costruzione concreta di questa successione di stringhe si pongono problemi di limitazione dei nastri disponibili e per un ipotetico contesto realistico si possono descrivere operazioni di ampliamento ulteriore la cui decisione, evidentemente, non può che dipendere dalle circostanze.

B18:**c.06** Ad ogni MSPGnI M che emette liste di stringhe prive di ripetizioni associamo una cosiddetta **successione di liste linearmente crescente**, in sigla **successione-LLC**, entità che a ogni t intero positivo associa la lista delle prime t stringhe generate, cioè $\mathsf{M}^{\mathcal{L}}(t)$.
A tale M risulta associata anche la cosiddetta successione nondecrescente di insiemi finiti, in sigla **successione-IIF**, entità che a ogni n intero positivo associa l'insieme i cui elementi provengono dalle prime n stringhe emesse, cioè l'insieme $\mathsf{SetY}\left(\mathsf{M}^{\mathcal{L}}(n)\right) =: \mathsf{M}^{\mathcal{G}}(n)$.

Si dice **lista-Pi emessa dalla MSPG M** la funzione che a ogni intero naturale associa una stringa che appartiene ad almeno una lista emessa dalla macchina. Il codominio di tale funzione è evidentemente $\mathsf{M}^{\mathcal{G}}$.

Si osserva ora che le successioni-LLC prodotte da $\mathsf{M}_{\mathbb{N}}$ e da M_{A} sono formate da liste nonripetitive e quindi forniscono direttamente successioni di insiemi linearmente crescenti, in sigla successioni-ILC, ove il termine crescita lineare esprime il fatto che l'insieme emesso nella fase t contiene t elementi.

Questi insiemi-G costituiscono ambienti di fondamentale importanza e vengono individuati con simboli ampiamente utilizzati:

$$\mathbb{N} := \mathsf{M}_{\mathbb{N}}{}^{\mathcal{G}} \qquad \text{e} \qquad \mathsf{A}^* := \mathsf{M}_{\mathsf{A}}{}^{\mathcal{G}} \ .$$

$\mathbb{N}$ viene detto **insieme dei numeri naturali** e viene identificato anche con espressioni come $\{0,1,2,3,...\}$ e $\{0,1,2,3,...,n,...\}$. Quando si rende opportuno precisare le notazioni che esprimono questi numeri si usano le espressioni I^* per le notazioni unadiche e $\{0\} \dot\cup (\{1,...,B-1\}, \{0,1,...,B-1\}^*$ per le notazioni posizionali in base B, dove B denota un intero maggiore di 1.

A^* viene detto **linguaggio delle stringhe sull'alfabeto** A o anche, un po' impropriamente, monoide libero sull'alfabeto A.

Si osserva anche che per ciascuna delle liste.GI $\mathsf{M}_{\mathbb{N}}{}^{\mathcal{L}}(t)$ e $\mathsf{M}_{\mathsf{A}}{}^{\mathcal{L}}(t)$, dati due diversi componenti si può decidere algoritmicamente quale dei due precede l'altro.
Le liste.G che godono di questa proprietà sono dette **liste.G di componenti confrontabili**.

Diciamo inoltre **lista.G sequenziabile** una lista.G per la quale si conosce un algoritmo in grado di trasformare (in un numero finito di passi) ogni componente della lista nel successivo.

Evidentemente ogni lista.G finita è sia sequenziabile che confrontabile.
La situazione è più problematica per le liste.GI.
Una tale lista sequenziabile è anche confrontabile.
Si può pensare di poter ottenere il confronto e la trasformazione nel successivo con una qualche attività finita, ossia eseguibile in un numero finito di passi. Quindi la distinzione dei componenti confrontabili e la distinzione della sequenzibilità risultano problematiche solo per le liste.GI.

Evidentemente ogni lista.GI sequenziabile è una lista.GI di componenti confrontabili: infatti date due stringhe w_a e w_b di una L lista.GI sequenziabile si possono portare avanti il parallelo le due manovre di passaggio al componente successivi che iniziano, risp., con w_a con w_b alternando un passo su una e un passo sull'altra; Dato che sia w_a che w_b sono state ottenute con un numero finito di passiè garantito che aut con la sottolista della L che inizia con w_a si raggiunge w_b stabilenco che w_a precede w_b, aut con la sottolista che inizia con w_b si raggiunge w_a per concludere che w_b precede w_a.

Non è invece garantito che una lista.GI di componenti confrontabili secondo una relazione d'ordine totale $\preceq$, sia anche una lista di componenti sequenziabili, nel senso che, date due stringhe w_a e w_b con la prima che precede la seconda, pur sapendo passare da ogni stringa alla successiva secondo $\preceq$, non si conosce una sequenza finita di stringhe una successiva della precedente che conduca dalla w_a alla w_b.

Un esempio è costituito dall'insieme delle stringhe di A^* con A alfabeto con almeno due caratteri. Date due stringhe w_a e w_b la prima precedente la seconda, diversa dalla successiva diella prima, non esiste sequenza finita di stinghe l'una successiva della precedente che porti dalla w_a alla w_b.

B18:**c.07** Si osserva che le precisazioni sulla generazione di liste.GI da parte di macchine MSPGI specifiche richiedono di occuparsi di tanti dettagli specifici e si osserva che queste preoccupazioni ci allontanano dall'affrontare questioni di portata generale che possano contribuire in misura rilevante ad operazioni per la crescita dell'*apparato* matematico-informatico.

Cerchiamo allora di definire delle entità formali che nelle argomentazioni per lo sviluppo dell'*apparato*, nella organizzazione e nella comunicazione delle conoscenze matematiche, algoritmiche e modellistiche possano sostituire vantaggiosamente, in termini di concisione e di leggibilità, le liste $\mathsf{M}^{\mathcal{L}}(n)$.

Data una MSPGI M abbiamo che la stringa $w \in \mathsf{A}_{\mathsf{M}}{}^*$ appartiene a $\mathsf{M}^G cl$ sse occorre in una delle liste $\mathsf{M}^L cl(t)$ (e quindi occorre in tutte le liste $\mathsf{M}^L cl(\bar t)$ con $\bar t > t$.

Altre sequenze emesse da MSPGI sono quelle usate per introdurre gli insiemi delle notazioni posizionali degli interi naturali $\mathbb{N}_{[B]}$ con $B = 2, 3, ...$ e il più astratto corrispondente insieme degli interi naturali $\mathbb{N}$.

Altre sequenze da MSPGI si ottengono con macchine che sono in grado di avere disponibili le stringhe di un insieme ambiente ma emettono sul nastro di uscita solo quelle che superano il vaglio di uno specifico algoritmo con il ruolo di selettore, ruolo da considerare caso particolare del ruolo di accettatore.

B18:**c.08** Ricordiamo che date due liste L_1 ed L_2 si dice che la prima è sottolista propria della seconda sse si può ottenere dalla seconda eliminando da essa alcune componenti. Diciamo invece che L_1 è **sottolista** di L_2 sse è sua sottolista propria o se coincide con essa.
Ricordiamo anche che per ogni coppia di liste la relazione "essere sottolista" è decidibile algoritmicamente.

Consideriamo ora due MSPG M_1 e M_2 che generano stringhe sullo stesso alfabeto A, le liste generate da tali macchine, dopo t fasi ${\mathsf{M}_1}^{\mathcal{L}}(t)$ e ${\mathsf{M}_2}^{\mathcal{L}}(t)$ ed i linguaggi procedurali da loro generati $S_1 := {\mathsf{M}_1}^{\mathcal{G}}$ ed $S_2 := {\mathsf{M}_2}^{\mathcal{G}}$.

Si dice che S_1 è un **sottolinguaggio**, ossia un sottoinsieme, di S_2 sse ogni stringa appartenente ad S_1, cioè a qualche ${\mathsf{M}_1}^{\mathcal{L}}(t_1)$, appartiene ad S_2, cioè a qualche ${\mathsf{M}_2}^{\mathcal{L}}(t_2)$.

Osserviamo esplicitamente che tra i sottoinsiemi di S_2 si può considerare lo stesso S_2; i restanti sottoinsiemi si dicono **sottoinsiemi propri**.

Va anche segnalato che si incontrano coppie di MSPG $\langle \mathsf{M}_1, \mathsf{M}_2 \rangle$ tali che risulta difficile stabilire la relazione insiemistica che intercorre tra ${\mathsf{M}_1}^{\mathcal{G}}$ e ${\mathsf{M}_2}^{\mathcal{G}}$.

Può essere difficile decidere se ${\mathsf{M}_2}^{\mathcal{L}}$ sia sottolista.G di ${\mathsf{M}_1}^{\mathcal{L}}$ o viceversa; stabilire se ogni stringa che occorre nella lista.G generata da una macchina sia occorrenza anche della lista.G emessa dall'altra; oppure trovare occorrenze di una lista.G che non siano occorrenze dell'altra; o anche stabilire se le due liste.G abbiano qualche o componente in comune o meno.

Data una lista.G L, finita o illimitata, generata da una MSPG M, ogni lista.G E_1 ottenibile da una sua variante arricchita da un qualche algoritmo selettore S è sottolista.G della L.
Vi sono macchine M che generano liste.GI $\mathsf{L} = \mathsf{M}^{\mathcal{L}}$ (illimitate) per le quali si individuano selettori che portano a sottoliste.GI (illimitate) e selettori che portano a liste finite).

Va segnalato esplicitamente che si incontrano coppie $\langle \mathsf{M}, \mathsf{S} \rangle$ del genere $\langle$ *macchina , selettore* $\rangle$ per le quali in una data fase di studio non si è in grado di stabilire se la sottolista.G che può essere selezionata dal selettore sia finita o illimitata.

B18:**c.09** Tutte le successioni ambiente o liste.GI ambiente fornite da MSPGI viste in precedenza sono nonripetitive e sono state generate secondo un ordine totale ben definito, cioè in un ordine verificabile con un ben definito algoritmo di confronto.

Questo accade anche per le liste.GI ottenute per selezione delle liste.GI ambiente; anch'esse quindi sono nonripetitive.

Ancor più in generale sono nonripetitive le sottosequenze delle sequenze generate da MSPG nonripetitive.

Si trovano invece facilmente sequenze ripetitive ottenute applicando un opportuno trasduttore (cioè un algoritmo che trasforma stringhe in stringhe) a ciascuna delle stringhe rese disponibili da una MSPG che genera una lista.GI nonripetitiva, in particolare a ciascuna delle stringhe di una lista.GI ambiente.

Ad esempio la procedura che rende disponibili i successivi interi naturali e che trasforma ciascuno di essi, lo denotiamo con n, nella sua scrittura binaria che denotiamo con n_2 e successivamente calcola ed emette la somma $\mathsf{bsum}(n)$ delle cifre di tale rappresentazione.
Questa procedura, come visto in B17e01, genera una lista illimitata di occorrenze di interi naturali che presenta numerose ripetizioni: ad esempio si ha $\mathsf{bsum}(3) = \mathsf{bsum}(5) = \mathsf{bsum}(6) = \mathsf{bsum}(9) = \mathsf{bsum}(10) = \mathsf{bsum}(12) = \mathsf{bsum}(17) = 2$.

Un esempio ancor più evidente è dato dalla MSPGI che fa corrispondere a ogni intero naturale la lunghezza della sua notazione posizionale in una qualche base B.

B18:**c.10** Si può però ridurre ogni sequenza da MSPG ripetitiva a una sequenza (finita o illimitata) nonripetitiva.
Si consideri la MSPG M e $\mathsf{L} := \mathsf{M}^{\mathcal{L}}$, la lista.G che essa emette. Se si modifica la M nella MSPG N aggiungendole una manovra riduttiva che prima della emissione di ciascuna stringa procede a controllare se una replica di tale stringa sia già stata emessa sul nastro di uscita e qualora la trovi rinunci a registrare in uscita la nuova copia.
La lista.G nonripetitiva così ottenuta, che va denotata con $\mathsf{N}^{\mathcal{L}}$, evidentemente è una sottolista.G della $\mathsf{M}^{\mathcal{L}}$.

Osserviamo anche che l'ottenimento di una parte iniziale nonripetitiva della $\mathsf{M}^{\mathcal{L}}$ risulta decisamente più costoso in termini di tempo della corrispondente sovrasequenza ripetitiva. Questo fa sì che anche l'emissione di sequenze ripetitive possa avere qualche interesse.

Consideriamo due macchine MSPG M_1 e M_2, le liste.G che procedono a emettere ${\mathsf{M}_1}^{\mathcal{L}}$ e ${\mathsf{M}_2}^{\mathcal{L}}$ e le corrispondenti sequenze nonripetitive che scriviamo ${\mathsf{N}_1}^{\mathcal{L}}$ e ${\mathsf{N}_2}^{\mathcal{L}}$.
Le due macchine si dicono equivalenti-list sse ${\mathsf{M}_1}^{\mathcal{L}} = {\mathsf{M}_2}^{\mathcal{L}}$; si dicono invece equivalenti-set sse ${\mathsf{M}_1}^{\mathcal{G}} = {\mathsf{N}_2}^{\mathcal{L}}$; le sequenze nonripetitive che emettono, ${\mathsf{N}_1}^{\mathcal{L}}$ e ${\mathsf{N}_2}^{\mathcal{L}}$ presentano, in genere secondo ordinamenti diversi, le stesse stringhe, cioè sono tali che per ogni stringa v che occorre nella ${\mathsf{N}_1}^{\mathcal{L}}$ si trova che occorre anche nella ${\mathsf{N}_2}^{\mathcal{L}}$, in genere in una posizione diversa.
La classe di equivalenza per permutazione di liste.G si dice insieme procedurale o **insieme-G**.
Ogni macchina MSPG M consente dunque di individuare un insieme procedurale.

Da molte sequenze ripetitive illimitate mediante l'eliminazione delle ripetizioni si ottengono liste nonripetitive illimitate: come esempi si considerino la lista delle lunghezze delle rappresentazioni in una qualche base $B = 2, 3, ...$ degli interi positivi e la lista dei prodotti $m \cdot n$ per la sequenza delle coppie $\langle m, n\rangle \in \mathbb{P} \times \mathbb{P}$ ottenuta scorrendo questo quadrato cartesiano con procedimento diagonale alla Cantor [B30a07].

Si trovano però anche liste da MSPGI illimitate ripetitive dalle quali si ottengono corrispondenti lista nonripetitive finite: un evidente esempio è dato da una MSPGI che procede a rendere disponibili le stringhe di $\mathbb{N}_{[10]}$ e per ciascuna di esse emette solo le ultime due cifre decimali: la corrispondente lista nonripetitiva contiene solo le 100 stringhe formate dalle coppie di cifre decimali (abbiamo trascurato di distinguere 0 da 00, 1 da 01, ... , 9 da 09) .

B18:**c.11** In genere, data una MSPG M non è difficile ottenere una diversa MSPG che genera la stessa lista.G $\mathsf{M}^{\mathcal{L}}$.

Ad esempio la successione degli interi pari si può generare con una macchina che inizia ponendo 0 sul nastro di uscita e quindi procede ad aggiungere dopo ogni numero pari $2\,k$ l'intero $2\,k + 2$ facilmente ottenibile.

Tale lista.G può essere fornita anche da una MSPG che procede a considerare tutti gli interi naturali e per ciascuno di essi stabilisce se è multiplo di 2 e in solo in caso positivo procede a registrarlo sul nastro di uscita.

Due MSPG che generano la stessa lista.G si dicono **MSPG equivalenti-G**.

Due liste.G si dicono **liste.G equivalenti-set** sse ogni stringa che occorre nella prima occorre anche nella seconda e viceversa.
Casi evidenti di liste.G equivalenti-set sono dati da liste.G ripetitive e dalle corrispondenti ottenute eliminando le stringhe ripetute.

Due MSPG si dicono **MSPG equivalenti-set** sse generano due liste.G equivalenti-set.
Due liste.GI equivalenti-set sono la lista $\mathsf{L}_{\mathtt{a,b,c}}$ delle stringhe sui caratteri `a`, `b` e `c` generate con la procedura descritta in `c08` e la lista $\mathsf{L}_{\mathtt{b,c,a}}$ generata dalla variante della precedente procedura che si basa sull'ordinamento rappresentabile con la scrittura $\{\mathtt{b} < \mathtt{c} < \mathtt{a}\}$.

Altre due liste.GI equivalenti-set degne di nota riguardano le coppie costituenti $\mathbb{N} \times \mathbb{N}$: la prima ottenuta con la generazione che corrisponde alla visita di $\mathbb{N} \times \mathbb{N}$ per segmenti codiagonali, la seconda ottenuta con la generazione che procede per ganci.

B18:**c.12** Introduciamo ora un genere di entità che si possono qualificare come insiemi-P e come insiemi-G ciascuna delle quali ha come elementi items di una lista.GI.
Come un insieme finito si ottiene per soggettificazione di una classe di liste (finite) equivalenti per occorrenza, uno degli insiemi che stiamo per introdurre si ottiene soggettificando una classe di liste.G equivalenti-set.

Consideriamo una MSPG M, il suo nastro di uscita che denotiamo con U_{M}, il suo alfabeto di uscita che scriviamo A_{M} e la lista.G che essa genera $\mathsf{M}^{\mathcal{L}}$.
Si dice **insieme generato da una MSPG M** la classe di equivalenza-set delle liste.G della quale fa parte $\mathsf{M}^{\mathcal{L}}$; tale entità si denota con $\mathsf{M}^{\mathcal{G}}$.
Se M_1 denota un'altra MSPG che genera una lista.G equivalente-set a $\mathsf{M}^{\mathcal{L}}$, cioè una lista.G appartenente alla stessa classe di equivalenza-set di cui fa parte $\mathsf{M}^{\mathcal{L}}$, vale l'uguaglianza $\mathsf{M}^{\mathcal{L}} = {\mathsf{M}_1}^{\mathcal{L}}$.

Ogni insieme generato da una MSPG verrà detto insieme procedurale, oppure in forma sincopata **insieme-G**, oppure **insieme ricorsivamente enumerabile**, o anche **linguaggio ricorsivamente enumerabile** o più concisamente anche **linguaggio REn**.

L'insieme generato da M è caratterizzato dagli enunciati della forma $w \in \mathsf{M}^{\mathcal{G}}$ validi per tutte e sole le stringhe w su A_{M} che vengono emesse su U_{M}. La formula $w \in \mathsf{M}^{\mathcal{G}}$ si traduce verbalmente dicendo che la stringa w appartiene all'insieme generato da M, oppure dicendo che la w è un **elemento** dell'insieme $\mathsf{M}^{\mathcal{G}}$, oppure dicendo che l'insieme $\mathsf{M}^{\mathcal{G}}$ contiene la stringa w.

B18:**c.13** Non è difficile descrivere alcune MSPG in grado di procedere nella emissione di coppie di numeri interi naturali, oppure di coppie di stringhe sopra uno stesso alfabeto A, oppure di coppie di entità appartenenti a uno stesso ambiente-G o appartenenti a due ambienti-G.

Consideriamo il caso più generale riguardante due MSPG: M che genera la lista.G $\mathsf{M}^{\mathcal{L}}$ di stringhe sull'alfabeto A e N che genera la lista.G $\mathsf{N}^{\mathcal{L}}$ di stringhe sull'alfabeto B e precisiamo la $MPSG$ in grado di generare la lista.G delle coppie $\left\langle \langle w_i, y_j \rangle \mid : w_i \in \mathsf{M}^{\mathcal{L}}, y_j \in \mathsf{N}^{\mathcal{L}} \right\rangle$.
La nuova macchina si pone in grado di utilizzare M e N i loro nastri di uscita e di procedere con una iterazione primaria illimitata.
Nella fase n-sima di questa iterazione chiede alle due macchine subordinate di generare sui loro nastri w_n e y_n e successivamente servendosi delle stringhe disponibili sui nastri procede ad emettere nuove coppie

$\langle w_i, y_j \rangle$ organizzando visite di segmenti codiagonali o visite di ganci di $2\,n+1$ coppie appartenenti a $\mathbb{N} \times \mathbb{N}$.
In tal modo si procede a generare "tutte le coppie della lista.G richiesta.

Consideriamo due MSPIA MSd_1 e MSd_2, le due liste.G ${MSd_1}^{\mathcal{L}} = \langle i \in \mathbb{N} :| \; x_i \rangle$ e ${MSd_2}^{\mathcal{L}} = \langle j \in \mathbb{N} :| \; y_j \rangle$ e e i corrispondenti insiemi-G ${MSd_1}^{\mathcal{G}} = \{i \in \mathbb{N} :| \; x_i\}$ e ${MSd_2}^{\mathcal{G}} == \{j \in \mathbb{N} :| \; y_j\}$.
Si dice prodotto cartesiano delle due liste.G la successione delle coppie $\langle x_i, y_j raa$ ordinate secondo lo scorrimento diagonale di $\mathbb{N} \times \mathbb{N}$; essa si denota con ${MSd_1}^{\mathcal{L}} \times {MSd_2}^{\mathcal{L}}$.
Questa successione costituisce una lista.G e la MSPIA che la genera si dice prodotto cartesiano di M_1 e M_2 e si denota con $\mathsf{M}_1 \times \mathsf{M}_2$.
L'insieme-G $\mathsf{SetY}\left({MSd_1}^{\mathcal{L}} \times {MSd_2}^{\mathcal{L}}\right)$ è il prodotto cartesiano ${MSd_1}^{\mathcal{G}} \times {MSd_2}^{\mathcal{G}}$ dei due insiemi-G ed evidentemente à un insieme-G.

Ugni sottoinsieme di ${MSd_1}^{\mathcal{G}} \times {MSd_2}^{\mathcal{G}}$ è un insieme-P e si dice relazione binaria tra i due insiemi-G, o anche **relazione [binaria] ricorsivamente enumerabile**, oppure concisamente **relazione-REn**.
Questa relazione può essere o meno un insieme-G; essa è tale sse si trova una MSPAI che genera una lista delle coppie che lo costituiscono.

Si precisa facilmente una MSPG in grado di generare la lista.G di coppie

$$\mathsf{D} = \left\langle \langle w_0, y_0 \rangle, \langle w_1, y_1 \rangle, \langle w_2, y_2 \rangle, ..., \langle w_n, y_n \rangle, ... \right\rangle \; = \; \left\langle n \in \mathbb{N} :| \; \langle w_n, y_n \rangle \right\rangle \; .$$

Tale lista.G si dice sottolista diagonale di ${MSd_1}^{\mathcal{L}} \times {MSd_2}^{\mathcal{L}}$

È semplice ricavare dalla D o della $\mathsf{R} := \mathsf{M}_1 \times \mathsf{M}_2$ le descrizioni delle due MSPG le quali procedono a generare, risp., la sequenza delle prime componenti delle coppie che occorrono nella D o nella R $\langle a_0, a_1, a_2, ..., a_n, ... \rangle$ e la sequenza delle seconde componenti della D o della R $\langle b_0, b_1, b_2, ..., b_n, ... \rangle$.

La prima delle due liste.G viene detta **prima proiezione** della R e si denota con $\boldsymbol{Prj}_1(\mathsf{R})$, mentre la seconda delle due sequenze viene detta **seconda proiezione** della R e si denota con $\boldsymbol{Prj}_2(\mathsf{R})$

Va osservato che, anche se la sequenza $\mathsf{R}^{\mathcal{L}}$ è nonripetitiva, ciascuna delle due sequenze $\boldsymbol{Prj}_i(\mathsf{R}^{\mathcal{L}})$ per $i = 1, 2$ può essere ripetitiva. e per certe esigenze può risultare necessario eliminare le loro ripetizioni.

Come per le relazioni finite, gli insiemi individuati dalle due suddette proiezioni sono detti, risp., **dominio dellarelazione** e **codominio della relazi** e si denotano con $\mathsf{dom}(\mathsf{R}^{\mathcal{G}}) := \mathsf{SetY}(\boldsymbol{Prj}_1(\mathsf{R}^{\mathcal{L}}))$ e con $\mathsf{cod}(\mathsf{R}^{\mathcal{G}}) := \mathsf{SetY}(\boldsymbol{Prj}_2(\mathsf{R}^{\mathcal{L}}))$.

Come per le relazioni finite, può accadere che questi due insiemi-G coincidano, oppure non coincidano e in particolare siano disgiunti.

In molte situazioni interessanti essi si possono convenientemente collocare in un unico insieme ambiente; questo è il caso di prodotto cartesiano di due insiemi-G costituiti da stringhe sopra lo stesso alfabeto.

In altre situazioni invece conviene collocare i due insiemi-G in due insiemi ambiente ben distinti.

Un esempio di questa seconda situazione si ricava dalla lista.G delle coppie costituite dalle scritture decimali dei successivi numeri naturali accompagnate dalle scritture degli stessi numeri in una lingua naturale. In questo caso il dominio è l'insieme dei numeri naturali e il codominio è un particolare insieme illimitato di stringhe sopra l'alfabeto della lingua naturale.

B18:**c.14** Una relazione procedurale si dice funzione procedurale, o **funzione-G**, sse la sua prima proiezione è una lista.G nonripetitiva, ossia sse ogni elemento del suo dominio è in relazione con un solo elemento del suo codominio.

Esempi di funzioni-G sono gli insiemi di coppie della forma

$$\left\langle n \in \mathbb{N} :| \ \langle n, f(n)\rangle \right\rangle \ ,$$

dove $f(n)$ denota un valore numerico fornito da un qualche algoritmo (in particolare da una espressione aritmetica) a partire dall'intero naturale n.

Una funzione-G f viene detta in particolare **funzione calcolabile** sse ciascuno dei suoi valori $f(n)$ è ottenibile con un algoritmo.

Esempi di tali funzioni calcolabili sono la $\mathsf{bsum}(n)$ e la riduzione alle ultime due cifre incontrate in a02.

Una relazione-G si dice più precisamente di biiezione procedurale o **biiezione-G** sse sono sequenze nonripetitive sia la sua prima proiezione che la sua seconda proiezione.
In altre parole una relazione-G $\mathsf{R} := \mathsf{R}^{\mathcal{G}}$ è una biiezione-G sse per ciascuna delle sue coppie $\left\langle w_i, y_j \right\rangle$ w_i compare come unica occorrenza nella $\boldsymbol{Prj}_1(\mathsf{R}^{\mathcal{G}})$) e y_j compare come unica occorrenza nella $\boldsymbol{Prj}_2(\mathsf{R}^{\mathcal{G}})$).

Ancora più in particolare si dice Ns permutazione procedurale o **permutazione-G** una biiezione procedurale le cui due proiezioni sono liste.G nonripetitive ed equivalenti-set.

Particolari permutazioni procedurali sono quelle ottenute da una lista.G nonripetitiva $\mathsf{M}^{\mathcal{L}}$ e da una delle liste.G ottenibili sottoponendo a una permutazione solo i primi membri di una sua sottolista.G finita e mantenendo invariate le sue altre coppie.

Anche queste costruzioni possono essere considerate generalizzazioni di costruzioni su insiemi finiti aventi caratteristiche simili.
Ci limitiamo a segnalare che con richieste molto vicine a quelle utilizzate per i tipi particolari di relazioni finite (simmetriche, antisimmetriche, transitive, ...) e di funzioni finite (endofunzioni, permutazioni, involuzioni, ...) si possono definire i corrispondenti tipi di relazioni procedurali e di funzioni procedurali.

B18:**c.15** Tornando a considerare la MSPG M di b02, è semplice descrivere la sua variante che invece di generare la lista.GI $\mathsf{M}^{\mathcal{L}} = \left\langle w_0, w_1, w_2, ..., w_n, ... \right\rangle$, genera la lista.GI di coppie

$$\mathsf{M}^{\mathcal{F}} := \left\langle \left\langle 0, w_0 \right\rangle, \left\langle 1, w_1 \right\rangle, \left\langle 2, w_2 \right\rangle, ..., \left\langle n, w_n \right\rangle, ... \right\rangle \ .$$

Questa lista.G individua una funzione procedurale avente come dominio $\mathbb{N}$ e come codominio $\mathsf{M}^{\mathcal{G}}$, un sottoinsieme dell'insieme ambiente $\mathsf{A}_{\mathsf{M}}{}^{*}$.
Le liste.G $\mathsf{M}^{\mathcal{L}}$ e $\mathsf{M}^{\mathcal{F}}$ in molti contesti possono essere identificate; in ogni caso si può affermare $\mathsf{cod}(\mathsf{M}^{\mathcal{F}}) = \mathsf{M}^{\mathcal{L}}$, mentre evidentemente $\mathsf{dom}(\mathsf{M}^{\mathcal{F}}) = \mathbb{N}$.

Se M è una MSPGF, $\mathsf{M}^{\mathcal{G}}$ denota l'insieme (finito) delle stringhe che sono registrate sul nastro di uscita quando la macchina si arresta oppure quando un algoritmo o una argomentazione stringente, ossia una dimostrazione, hanno garantito che la macchina non potrà emettere stringhe diverse da quelle già correntemente registrate e quindi consentono di interrompere la evoluzione della M con la certezza di avere ottenute tutte le stringhe che essa può generare.

Nel caso in cui M sia la MSPGI che genera I^{*}, viene generato l'insieme delle rappresentazioni unadiche degli interi naturali e si può scrivere $w \in \mathsf{M}^{\mathcal{G}}$ per ogni w stringa costituita da un numero qualsiasi di segni "I".
Nel caso in cui A denota un insieme finito di simboli ed M è la MSPGI che genera A^{*}, viene generato l'insieme delle stringhe sull'alfabeto A e si può scrivere $w \in \mathsf{M}^{\mathcal{G}}$ per ogni w stringa ottenuta giustapponendo una qualche sequenza di caratteri di A.

B18:**c.16** Nel caso in cui M sia la MSPGI che genera $\mathbb{N}_{[B]}$ per qualche intero $B = 2, 3, ...$ con il ruolo della base , la corrispondente lista.G elenca l'insieme delle notazioni in base B degli interi naturali e si può scrivere $w \in \mathsf{M}^{\mathcal{G}}$ per ogni w scrittura in base B di un numero intero naturale.

Abbiamo osservato che le elaborazioni sui numeri interi naturali si possono effettuare servendosi di ciascuna delle sue rappresentazioni $\mathbb{N}_{[B]}$ senza differenze sostanziali, ossia con differenze concernenti solo singoli dettagli operativi.
Le diverse notazioni posizionali si dicono **equivalenti operativamente**, per significare che le elaborazioni condotte su notazioni in due diverse basi si possono porre in collegamento servendosi di opportune trascodifiche e per significare che due elaborazioni su diverse notazioni conducono a risultati trasformabili l'uno nell'altro e quindi da considerare equivalenti in tutte le loro possibili conseguenze.

È dunque lecito individuare come **insieme degli interi naturali** la soggettificazione delle diverse rappresentazioni $\mathbb{N}_{[B]}$ per $B = 2, 3, ...$, nonché della notazione unadica (non avendo peso nelle attuali considerazioni le inefficienze di quest'ultima).
Questo insieme si denota con $\mathbb{N}$ e si può porre in biiezione con tutti gli insiemi-G $\mathbb{N}_{[B]}$ o simili, insiemi che chiamiamo anche **codifiche degli interi naturali**.

Gli insiemi che si possono porre in biiezione con $\mathbb{N}$ sono chiamati **insiemi numerabili**.
Sono quindi numerabili tutti gli insiemi che costituiscono codifiche dei numeri naturali.

D'ora in avanti in linea di massima trascureremo di distinguere tra un insieme come $\mathbb{N}$ e le sue diverse codifiche.
I numeri naturali ottenuti da definizioni o costruzioni specifiche in genere, seguendo la tradizione prevalente, saranno presentati attraverso la loro notazione decimale.

B18:**c.17** Consideriamo due MSPG M e N e i due corrispondenti insiemi procedurali $\mathsf{S}_M := \mathsf{M}^{\mathcal{G}}$ e $\mathsf{S}_N := \mathsf{N}^{\mathcal{G}}$.
Si dice che il primo è **sottoinsieme** del secondo sse ogni $w \in \mathsf{S}_M$ appartiene anche a S_N; in tal caso si scrive $\mathsf{S}_M \subseteq \mathsf{S}_N$.

Si dice che S_M è **sottoinsieme proprio** di S_N e si scrive $\mathsf{S}_M \subset \mathsf{S}_N$ sse $\mathsf{S}_M \subseteq \mathsf{S}_N$ e $\mathsf{S}_M \neq \mathsf{S}_N$.

Evidentemente un insieme procedurale fornito da una sottolista.G propria della $\mathsf{N}^{\mathcal{L}}$ è sottoinsieme proprio dell'insieme $\mathsf{N}^{\mathcal{G}}$.

Definizioni e notazioni prevedibili riguardano le relazioni trasposte di sovrainsieme e sovrainsieme proprio.

Può accadere che si incontrino coppie di insiemi procedurali per le quali in una data fase degli studi che li riguardano non si riesca a decidere se uno dei due è sottoinsieme o sottoinsieme proprio dell'altro. Osserviamo che per una coppia di insiemi finiti questa indecisione, almeno in linea di principio, è evitabile.

Si possono avere MSPG che procedono senza mai fermarsi, ma che generano linguaggi finiti, vuoi perché da un certo passo in poi non generano altre stringhe, vuoi perché da un certo passo in poi generano solo stringhe già generate in precedenza.
Per ciascuna di queste macchine in genere risulta conveniente sostituirla con una MSPGF che sia una sua equivalente-set. La effettiva attuazione di una tale sostituzione potrebbe incontrare difficoltà in quanto dipende dalle caratteristiche peculiari di ciascuna macchina ovvero dalle conoscenze riguardanti le stringhe che essa può o non può generare.

B18:**c.18** Abbiamo definito insieme infinito un insieme che si può porre in corrispondenza biunivoca con un suo sottoinsieme proprio.

Si osserva che attualmente questa è una definizione di prospettiva, in quanto non si è ancora data una definizione generale rigorosa di insieme e tanto meno quella di corrispondenza biunivoca tra insiemi qualsiasi.

In attesa di definizioni più rigorose dovremo servircene solo quando accade di analizzare una coppia di insiemi-G $\langle M, N\rangle$ e una relazione-G R tra M ed N e si possa decidere con argomentazioni condivisibili il carattere della R; in particolare può essere rilevante stabilire se R sia una funzione, una iniezione, una suriezione, una biiezione, una relazione finita, ...

(1) Prop.: Gli insiemi associati a liste.G da MSPGI nonripetitive sono insiemi infiniti.

Dim.: Da una MSPGI M che genera la lista.G potenzialmente illimitata e nonripetitiva $\mathsf{L} = \mathsf{M}^{\mathcal{L}}$ mediante la semplice aggiunta di opportuni meccanismi selettivi si ottengono MSPGI che generano sottoelenchi di $\mathsf{M}^{\mathcal{L}}$ illimitati (e ovviamente nonripetitivi) ∎

In particolare da una lista.GI $\langle a_1, a_2, a_3, ...\rangle$ si può eliminare il prefisso di lunghezza n per un qualsiasi n intero positivo e considerare la corrispondenza biunivoca

$$\{\langle a_1, a_{1+n}\rangle, \langle a_2, a_{2+n}\rangle, \langle a_3, a_{3+n}\rangle, ...\}\ .$$

Si possono proporre vari tipi di macchine derivate aggiungendo selettori a MSPG precedentemente studiate che denotiamo con M. Vediamo alcuni di questi tipi.

(1) Macchine che evitano di emettere solo un numero finito di stringhe della $\mathsf{M}^{\mathcal{L}}$ smettendo dopo un certo numero finito di passi di esercitare la selezione.

(2) Macchine che trascurano infinite stringhe emettendo solo quelle che nella lista $\mathsf{M}^{\mathcal{L}}$ occupano posizioni dispari o posizioni multiple di un intero $k = 2, 3, ...$ (grazie a un uso facilmente immaginabile di un contatore ciclico) o, in generale, che occupano posizioni riconoscibili da un algoritmo.

(3) Macchine che alternativamente interrompono e riprendono le operazioni in successive fasi determinate da una lista.G di interi positivi crescenti preliminarmente fissata $\langle i_1, r_1, i_2, r_2, i_3, r_3, ...\rangle$.

Consideriamo due MSPGI M ed N e scriviamo $\langle a_1, a_2, ..., a_n, ...\rangle := \mathsf{M}^{\mathcal{L}}$ e $\langle b_1, b_2, ..., b_n, ...\rangle := \mathsf{N}^{\mathcal{L}}$; supponiamo anche che $\mathsf{N}^{\mathcal{L}}$ sia una sottolista.G illimitata della $\mathsf{M}^{\mathcal{L}}$.
È facile definire una macchina $\mathbf{K}$ che chiede alternatamente alla M e alla N di procedere alla emissione di una propria nuova stringa; se denotiamo, risp., con a_n e b_n le stringhe emesse nelle due corrispondenti fasi emissive caratterizzate dal progressivo n, chiediamo che dopo aver ottenuta la b_n la $\mathbf{K}$ emetta la coppia $\langle a_n, b_n\rangle$ sul proprio nastro di uscita.
Dunque $\mathbf{K}^{\mathcal{L}} = \left\langle \langle a_1, b_1\rangle, \langle a_2, b_2\rangle, ..., \langle a_n, b_n\rangle, ...\right\rangle$ e la nonripetitività di $\mathsf{M}^{\mathcal{L}}$ e di $\mathsf{N}^{\mathcal{L}}$ implica che questa successione di coppie individui una biiezione-G tra $\mathsf{M}^{\mathcal{G}}$ e $\mathsf{N}^{\mathcal{G}}$.
Questa biiezione se $\mathsf{N} = \mathsf{M}$ è più precisamente una endofunzione-G entro $\mathsf{M}^{\mathcal{G}}$.

B18:**c.19** **(1) Prop.**: Ogni insieme finito F sopra un dato alfabeto A può considerarsi un insieme procedurale.

Dim.: Si tratta di individuare una MSPG che emette una qualsiasi lista degli elementi di F (non importa in quale ordine). Una tale lista può essere registrata sopra un nastro T in un tempo finito (anche se potrebbe non essere facilmente prevedibile) e questo può essere messo a disposizione di una MSPG che non deve far altro che trasferire sul proprio nastro di uscita il contenuto di T ∎

Dunque le nozioni di elemento e di appartenenza per gli insiemi procedurali estendono quelle introdotte per gli insiemi finiti.

A questo punto si pone il problema di stabilire fino a che punto si possono estendere agli insiemi procedurali le altre costruzioni e le altre relazioni esaminate per gli insiemi finiti.

Per cominciare a muoversi in questa direzione ora procederemo costruttivamente a introdurre composizioni che utilizzando due MSPG definiscono nuove macchine dello stesso genere per ciascuna delle

quali l'insieme procedurale che genera va considerato come una estensione di una nota composizione binaria di insiemi finiti (unione, intersezione, eliminazione, prodotto cartesiano, ...).

In tal modo saremo in grado di definire un complesso costituito da operazioni binarie e da relazioni riguardanti insiemi procedurali che permetterà di tenere sotto controllo **SetG**, la collezione-P degli insiemi-G mediante espressioni di genere algebrico e relazionale.

B18:**c.20** Per gli sviluppi sopra annunciati ci serviremo di alcune MSPG che denotiamo con M_j per $j = 1, 2, ..., \mu$ e delle liste.G da esse generate $\mathsf{L}_j := \mathsf{M}_j^{\mathcal{L}}$. Per talune argomentazioni risulta vantaggioso chiedere che queste macchine generatrici siano nonripetitive.

Per tenere sotto controllo un linguaggio procedurale potrebbe essere utile disporre di una MSPG che genera le sue stringhe secondo un ordine che possa essere concretamente tenuto sotto controllo.

Molti sviluppi sopra gli insiemi procedurali fanno riferimento a insiemi ambiente forniti da liste.G che seguono un ben definito ordinamento totale; queste liste le chiamiamo **liste.GT**. In particolare risulta utile che ogni lista delle notazioni posizionali degli interi naturali segua l'ordinamento $\leq$ e che per ogni alfabeto ordinato

$$\mathsf{A} := \langle a_1 \prec a_2 \prec ... \prec a_{n-1} \prec a_n \rangle \ ,$$

la lista delle sue stringhe segua l'ordinamento-llx determinato dall'ordinamento totale $\prec$.

Per ciascuno di questi ambienti totalmente ordinati si dispone di un cosiddetto **algoritmo di confronto** Kcl che per ogni duetto di sue stringhe consente di scegliere quale delle due sia da considerare la precedente.

In generale chiamiamo **lista.GK** ogni lista.G L dotata di un algoritmo di confronto $\mathcal{K}$ e di un ordinamento totale determinato dallo stesso $\mathcal{K}$. Un insieme-G associato a una lista.GK viene chiamato **insieme-GK**.

Un algoritmo di confronto $\mathcal{K}$ per una lista L e il conseguente suo ordinamento totale possono essere attribuiti anche a ciascuna delle MSPG che generano L.
Una MSPG che genera una lista.GK si può chiamare **MSPGK**.

Per qualcuna di queste liste.GK L può risultare conveniente, sul piano lessicale, privilegiare un particolare algoritmo $\mathcal{K}$ da chiamare algoritmo canonico e chiamare ordinamento canonico anche il conseguente ordinamento totale della L.

Si osserva che un algoritmo di confronto e un ordinamento totale algoritmico per una lista.GK L quando sono ridotti alle coppie di elementi di una L_1 sottolista.G delle L diventano algoritmo di confronto e ordinamento totale algoritmico anche per tale L_1; quindi anche questa sottolista risulta essere una lista.GK e costituisce una rappresentazione di un insieme-GK.

Un ordinamento totale di una lista.G viene detto anche **ordinamento sequenziale**, mentre la manovra che definisce un tale ordinamento viene chiamata dice **sequenzializzazione**. Con questo termine si designa anche la regola che la determina.

Un ordinamento totale di una lista.G si dice **ordinamento progressivo** sse si conosce un algoritmo che data una sua stringa w fornisce la stringa che la segue immediatamente.

Data una MSPGK, la si può modificare in modo che in ogni stadio della sua evoluzione presenti le stringhe sul suo nastro di uscita U secondo l'ordinamento algoritmico stabilito da un suo algoritmo di confronto (e di conseguenza non presenti stringhe ripetute).

Basta infatti aggiungere a qualsiasi MSPGK un dispositivo che, dopo che essa ha reso disponibile una nuova stringa, stabilisce mediante l'algoritmo di confronto se è già presente su U e solo nel caso si sia

verificata la sua assenza la aggiunge collocandola in modo da mantenere la lista registrata sopra U nell'ordine richiesto.
Anche per queste manovre non ci preoccupiamo della loro efficienza, tendenzialmente molto bassa.

B18:**c.21** Occorre osservare che la possibilità di mantenere totalmente ordinato in ogni stadio evolutivo di una MSPG il nastro di uscita (nastro finitamente registrato e concretamente disponibile) è cosa ben diversa della disponibilità di una MSPG che generi stringhe secondo un ordine sequenziale algoritmico, cioè che presenti le stringhe che sta producendo sopra un nastro di uscita che si modifica solo per l'aggiunta di ogni nuova stringa che colloca dopo quelle presenti, le stringhe generate in precedenza.

Questi due comportamenti sono sostanzialmente equivalenti per le liste e per gli insiemi finiti, mentre per le generazioni illimitate la possibilità di generare stringhe che vengono collocate prima di qualche stringa generata precedentemente può portare a difficoltà nell'utilizzare la loro equivalenza-set.

Consideriamo due MSPG M_1 ed M_2 che procedono a emettere stringhe sullo stesso alfabeto A secondo due ordini diversi ω_1 e ω_2, entrambe evitando ripetizioni e procedendo per fasi alternate, ovver in modo che dopo l'emissione di una stringa da parte di una macchina questa stia inattiva fino alla emissione di una stringa da parte dell'altra.
Per esse può essere importante porsi il problema se sono equivalenti-set, che denotiamo concisamente con PES.

Supponiamo che a un certo stadio evolutivo M_2 abbia emesso solo stringhe già emesse da M_1. A questo punto per chiarire PES dobbiamo chiedere se in una fase successiv M_2 emetterà un'altra stringa già emessa da M_1, oppure una stringa che potrà essere emessa da M_1 solo successivamente, oppure una stringa che M_1 non potrà emettere, situazione che potrebbe essere dimostrata o solo ipotizzata.

A questo trilemma si può cercare di rispondere solo analizzando i dettagli delle due macchine e potrebbe capitare di trovarsi nella incapacità di dare una risposta attendibile: potrebbe accadere che M_2 non emetta alcuna altra stringa, oppure che emetta stringhe che M_1 non ha emesse e che non si sappia decidere se mai le emetterà.

Un modo per cercare di chiarire la situazione in un caso specifico consiste nel far proseguire le due evoluzioni (ma consumando risorse) nella speranza che le produzioni successive delle due macchine forniscano indicazioni capaci di sciogliere i dubbi.

Va segnalato che la libertà che la definizione delle MSPG concede alle possibili loro evoluzioni in generale rende possibile che non sia garantita la capacità di raggiungere indicazioni dirimenti in seguito a ulteriori fasi evolutive.

Di fronte alla evoluzione di una MSPG M che emette stringhe su un alfabeto A possono presentarsi altre difficoltà per le quali a livello generale non risulta garantita la possibilità di decidere i dubbi che seguono.

Come sapere se una generica stringa w su A sarà emessa da una macchina MSPG?
Come sapere se una macchina MSPG con la sua evoluzione è in grado di generare una lista illimitata, oppure che generarà una lista finita, oppure (quel che sarebbe peggio) che non sarà in grado di emettere alcuna stringa?

Vedremo inoltre che il sopra accennato tipo di difficoltà e la sopra accennata impossibilità equivalgono a quelle sulla possibilità di trovare una MSPG in grado di generare la propria lista.G secondo un determinato ordinamento algoritmico.
In effetti giungeremo a individuare delle MSPG per le quali si dimostra impossibile rispondere alle accennate domande.

B18:**c.22** Abbiamo vista l'opportunità di servirsi di termini come nastro di lunghezza infinita, risorse che crescono all'infinito ed insieme infinito.
Vedremo più oltre [B19f] che due insiemi che si possono mettere in biiezione si dicono avere lo stesso cardinale e che la relazione di equicardinalità è una equivalenza.
Vedremo poi come si possono individuare collezioni di insiemi che costituiscono classi di equicardinalità e troveremo coppie di insiemi che non si possono mettere in biiezione ma sono tali che uno di essi può essere posto in biiezione solo con qualche sottoinsieme proprio dell'altro e per questo si dice avere cardinale inferiore a quello dell'altro.
Vedremo quindi che si rende necessario definire una gerarchia tra le collezioni di insiemi infiniti equicardinali ovvero, in altre parole, che risulta necessaria una gerarchia tra i livelli di infinitezza da attribuire agli insiemi.

Altri infiniti si incontreranno con la definizione degli interi negativi [B20a] e dei primi oggetti geometrici [B21, B22, B23, B30, ...) e con l'introduzione delle varie accezioni della nozione di limite [B35, I12, I13, I16, ...).
Si prevede quindi l'opportunità di introdurre un sistema di notazioni per i diversi livelli di infinitezza degli insiemi.
Un primo simbolo è $+\infty$ e a esso cominciamo a dare due compiti: esso si può definire come cardinale (numero) delle delle caselle di un nastro illimitato e può contribuire alla notazione $\mathbf{[}k : +\infty\mathbf{)}$ con la quale vogliamo individuare l'insieme-GI di tutti gli interi positivi maggiori di un dato intero positivo k.

B18:d. composizioni di insiemi-G

B18:**d.01** Ci proponiamo di definire, sulle tracce di quanto fatto per gli insiemi finiti, le composizioni booleane degli insiemi-G. Giungeremo a dimostrare che queste composizioni, come quelle riguardanti gli insiemi finiti, sono operazioni binarie e unarie che portano ad insiemi dello stesso genere, cioè ad altri insiemi-G e che per esse valgono proprietà cheestendono quelle riguardanti gli insiemi finiti.

Ancora consideriamo, per $i \in \{1, 2, \mu u\}$, le MSPG M_i ed i corrispondenti linguaggi procedurali generati $\mathsf{L}_i := \mathsf{M}_i{}^{\mathcal{G}}$; denotiamo inoltre con A_i l'alfabeto che la M_i usa per il suo nastro di emissione e l'alfabeto loro unione con $\overline{A} := \mathsf{A}_1 \cup \mathsf{A}_2$.

Precisiamo che con μu, segno che chiamiamo **metamu**, denotiamo la stringa muta da utilizzare per espressioni usate per proprietà di stringhe e linguaggi oggetto dello studio attuale.

Diciamo **unione** di L_1 ed L_2 il linguaggio-P su $\overline{A}$ le cui stringhe appartengono o ad L_1 o ad L_2 (o a entrambi) e che denotiamo con $\mathsf{L}_1 \cup \mathsf{L}_2$.

Diciamo **intersezione** di L_1 ed L_2 il linguaggio-P su $\overline{A}$ le cui stringhe appartengono sia ad L_1 che ad L_2 e che denotiamo con $\mathsf{L}_1 \cap \mathsf{L}_2$.

Diciamo **eliminazione** di L_2 da L_1 il linguaggio-P su A_1 le cui stringhe appartengono ad L_1 ma non ad L_2 e che denotiamo con $\mathsf{L}_1 \setminus \mathsf{L}_2$.

Diciamo **complementazione** di L_i entro $\mathsf{A}_i{}^*$ il linguaggio-P su A_i le cui stringhe non appartengono ad L_i, cioè il lingaggio $\mathsf{A}^* \setminus \mathsf{L}_i$, denotato anche con $\mathsf{L}_i{}^{\complement}$.

Diciamo **differenza simmetrica** di L_1 ed L_2 il linguaggio-P su $\overline{A}$ costituito dalle stringhe che appartengono ad L_1 ma non a L_2 e dalle stringhe che appartengono ad L_2 ma non a L_1, linguaggio che denotiamo con $\mathsf{L}_1 \ominus \mathsf{L}_2$.

Diciamo **prodotto cartesiano** di L_1 ed L_2 il linguaggio-P costituito dalle coppie di stringhe

$$\langle w_1, w_2 \rangle \in \mathsf{L}_1 \times \mathsf{L}_2$$

e che denotiamo con $\mathsf{L}_1 \times \mathsf{L}_2$; evidentemente esso è un sottoinsieme di $\mathsf{A}_1{}^* \times \mathsf{A}_2{}^*$.

B18:**d.02** Introduciamo una MSPG $\mathsf{M}_\cup$ che si serve di$\overline{A}$ come alfabeto di uscita e del nastro di uscita $U_\cup$ e che si avvale delle due MSPG M_1 e M_2; la chiamiamo **MSPG composizione per unione** delle due suddette macchine.

$\mathsf{M}_\cup$ procede per passi successivi, in ciascuno dei quali chiede alternatamente ad M_1 e ad M_2 di effettuare un passo evolutivo e quando una di esse rende disponibile una stringa la emette sul nastro $U_\cup$.
Se una delle due macchine, la M_i, arresta la sua evoluzione, la $\mathsf{M}_\cup$ procede a registrare le stringhe che l'altra, la $\mathsf{M}3-i$, procede a fornire stringhe da emettere.
Se entrambe si arrestano, anche la $\mathsf{M}_\cup$ si arresta con l'emissione di una lista finita.

È evidente che ogni stringa che compare su $U_\cup$ deve appartenere ad L_1 o ad L_2 (o a entrambi questi insiemi procedurali), come è evidente che, per $i = 1, 2$, ogni stringa emessa da M_i dopo n_i passi deve comparire su $U_\cup$ dopo $2\,n_i - 1$ o dopo $2\,n_i$ passi al più.

Dunque l'insieme-G generato dalla $\mathsf{M}_\cup$ è evidentemente l'unione di $\mathsf{L}_1 := \mathsf{M}_1{}^{\mathcal{G}}$ ed $\mathsf{L}_2 := \mathsf{M}_2{}^{\mathcal{G}}$.

Osserviamo in particolare che entrambe le M_i emettono un numero finito di stringhe sse la $\mathsf{M}_\cup$ genera un insieme finito di stringhe.
Quindi la costruzione della $\mathsf{M}_\cup$ è un'estensione della costruzione della macchina che produce l'unione di due insiemi finiti.

A questo punto è lecito affermare che l'unione di due insiemi-G è un insieme-G.

Evidentemente, anche se M_1 ed M_2 sono nonripetitive, la macchina $\mathsf{M}_\cup$ potrebbe essere ripetitiva; tuttavia, come si è già osservato, non è difficile dotare $\mathsf{M}_\cup$ di un dispositivo che evita di emettere sul nastro $U_\cup$ una stringa già presente.

B18:**d.03** Introduciamo una MSPG $\mathsf{M}_\cap$ che si serve dell'alfabeto di uscita $\mathsf{A}_\cap := \mathsf{A}_1 \cap \mathsf{A}_2$ e del nastro di uscita $U_\cap$ e che utilizza M_1 e M_2 come sue sottomacchine; la chiamiamo **MSPG composizione per intersezione** delle suddette macchine.

Anche la MSPG $\mathsf{M}_\cap$ procede per passi uccessivi in ciascuno dei quali chiede alternatamente ad M_1 e ad M_2 di effettuare un passo evolutivo.
Entrambe le sottomacchine, che conviene pensare nonripetitive, mantengono il proprio nastro di emissione, ma in esso registrano solo le stringhe sull'alfabeto $\mathsf{A}_1 \cap \mathsf{A}_2$.

La $\mathsf{M}_\cap$ consultando questi nastri risulta in grado di registrar su $U_\cap$ le sole stringhe che vengono generate da entrambe le macchine: infatti dopo che la M_i (i=1,2) ha emesso una nuova stringa l'esame del nastro di uscita dell'altra macchina M_{3-i} consente di decidere se presentare o meno tale stringa su $U_\cap$.

L'insieme-G che la $\mathsf{M}_\cap$ procede a generare è evidentemente l'intersezione di $\mathsf{L}_1 := {\mathsf{M}_1}^{\mathcal{G}}$ e $\mathsf{L}_2 := {\mathsf{M}_2}^{\mathcal{G}}$.

Se una delle due macchine, la M_i, emette una lista finita e si arresta, si può garantire che l'intersezione $\mathsf{L}_1 \cap \mathsf{L}_2$ è un linguaggio finito. Non è invece lecito arrestare anche la M_{3-i}, in quanto ci si deve chiedere se questa potrà emettere stringhe che sono state registrate sul nastro di uscita della M_i e che devono essere emesse su $U_\cap$ per essere assegnate a $\mathsf{L}_1 \cap \mathsf{L}_2$.

Solo l'eventuale esaurimento del nastro di uscita della M_{3-i} per arresto di tale macchina o in seguito a una dimostrazione dell'esaurimento dell'ampliamento del nastro di uscita della M_i comporta l'arresto della $\mathsf{M}_\cap$; in caso contrario risulta necessario proseguire illimitatamente l'evoluzione della M_{3-i}.

A questo punto possiamo comunque affermare che l'intersezione di due insiemi-G è un insieme-G.

Si osserva tuttavia che può rimanere problematico stabilire se una particolare stringa $w \in {\mathsf{A}_\cap}^*$ appartiene o meno a $\mathsf{L}_\cap$. Se a un certo stadio dell'evoluzione della $\mathsf{M}_\cap$ osserviamo che w è stata emessa solo da M_1, rimane il dubbio se M_2 la emetterà successivamente (ed in tal caso $w \in \mathsf{L}_1 \cap \mathsf{L}_2$, oppure non la emetterà, e in tal caso $w \notin \mathsf{L}_1 \cap \mathsf{L}_2$.
Le decisioni che chiedono di proseguire illimitatamente l'evoluzione di almeno una MSPG risultano problematiche.

Con questa riserva si può affermare che anche la costruzione della intersezione di due insiemi-G sia un'estensione della costruzione omologa per l'intersezione di insiemi finiti.

Consideriamo il caso in cui sia M_1 che M_2 emettono stringhe seguendo due loro rispettivi ordini algoritmici Ω_1 e Ω_2 e che interessi sapere se si può individuare un ordine algoritmico per $\mathsf{S}_\cap := {\mathsf{M}_1}^{\mathcal{G}} \cap {\mathsf{M}_2}^{\mathcal{G}}$, ordine che denotiamo con $\Omega_\mathcal{K}$, dove con $\mathcal{K}$ si denota un opportuno algoritmo di confronto.

Si osserva che fissato un qualsiasi algoritmo di confronto $\mathcal{K}$ si può ottenere un ordine sequenziale algoritmico $\Omega_\mathcal{K}$ basato sopra tale $\mathcal{K}$ munendo $\mathsf{M}_\cap$ di un opportuno dispositivo di mantenimento di tale ordine per le stringhe che si vanno emettendo, sia sui nastri che riproducono le emissioni di L_1 ed L_2, sia sul nastro $U_\cap$ nel quale si procede ad emettere $\mathsf{L}_\cap$.
Evidentemente conviene che $\Omega_\mathcal{K}$ sia il più possibile coerente con Ω_1 ed Ω_2, in quanto i controlli sui nastri per le singole macchine possono essere evitati o ridotti.

B18:**d.04** Introduciamo una MSPG $\mathsf{M}_\setminus$ che si serve dell'alfabeto di uscita A_1 e di un suo nastro di uscita $U_\setminus$ e che utilizza come sue sottomacchine due MSPG M_1 e M_2 che per semplicità supponiamo nonripetitive; questa macchina la chiamiamo **MSPG composizione per eliminazione** tra due insiemi-G.

Anche la $\mathsf{M}_\setminus$ procede per manovre successive in ciascuna delle quali chiede alternatamente ad M_1 e ad M_2 di effettuare un passo evolutivo. Supponiamo inoltre, per semplicitàm che le due macchine adottino lo stesso alfabeto (in caso contrario serve la loro unione).

La $\mathsf{M}_\setminus$ si serve di M_1 ed M_2 come sottomacchine, adotta un nastro di uscita $U_\setminus$ e si serve dei nastri U_1 e U_2 di uscita per le due sottomacchine, riservandosi della possibilità di cancellare le stringhe che vi saranno registrate sostituendole con segni neutri.

Alla fine di una fase nella quale la M_2 ha individuato una nuova stringa w_2 la $\mathsf{M}_\setminus$ controlla se essa compare o meno su $\mathsf{U}_\setminus$; se la trova la cancella, se non la trova la registra su U_2

Simmetricamente alla fine di una fase nella quale la M_1 ha individuato una nuova stringa w_1 la $\mathsf{M}_\setminus$ controlla se essa compare o meno su U_2; se la trovala cancella da U_2, se non la trova la registra su $\mathsf{U}_\setminus$.

In tal modo ciascuna delle stringhe che sono individuate da M_1 ma non da M_2 viene a trovarsi su $U_\setminus$, mentre le stringhe non individuate da MSd_1 non hanno modo di venire registrate su $U_\setminus$, le stringhe individuate prima da M_1 poi da M_2 dopo essere state registrate su $\mathsf{U}_\setminus$ vengono cancellate e quelle individuate prima da M_2 e poi da M_1 non possono comparire in $\mathsf{U}_\setminus$.

L'insieme procedurale generato dalla $\mathsf{M}_\setminus$, evidentemente, è l'eliminazione da $\mathsf{S}_1 := {\mathsf{M}_1}^{\mathcal{G}}$ di $\mathsf{S}_2 := {\mathsf{M}_2}^{\mathcal{G}}$.

Se la macchina M_1 produce una lista finita o se comunque si può garantire che a un certo punto non ne emetterà altre, non si può arrestare l'evoluzione della M_2 in quanto essa potrebbe emettere qualche stringa non generata da M_1 che andrebbe sicuramente emessa su $\mathsf{U}_\setminus$ e quindi si deve procedere, forse illimitatamente, alla generazione da parte di M_2.
Simmetricamente se la macchina M_2 produce una lista finita o se comunque si può garantire che a un certo stadio evolutivo non emetterà altro, si deve far proseguire solo la M_1.

Si ha l'arresto della $\mathsf{M}_\setminus$ solo se entrambe le sottomacchine concludono la loro generazione e in tal caso e solo in quasto, viene generata una lista finita.

A questo punto possiamo affermare che l'eliminazione tra due insiemi-G conduce a un insieme-G.

Come le due precedenti costruzioni, quindi, anche la costruzione per l'eliminazione tra due insiemi procedurali può essere considerata un'estensione della operazione per l'eliminazione tra due insiemi finiti.

Anche per questa composizione di insiemi-G può avere interesse la appartenenza al linguaggio risultato di una particolare stringa w.
Evidentemente se si riconosce che w appartiene a L_1, per la sua appartenenza ad L_2 si può avere una situazione di attesa con una incertezza simile a quella che si riscontra per la macchina adottata per l'intersezione di liste.G.

Se a un certo stadio dell'evoluzione della $\mathsf{M}_s mi$ osserviamo che l'ha emessa solo M_1, rimane il dubbio se M_2 la emetterà successivamente (ed in tal caso $w \notin (\mathsf{L}_1 \setminus \mathsf{L}_2)$, oppure non la emetterà, e in tal caso $w \in \mathsf{L}_1 \setminus \mathsf{L}_2$.
Se invece a un certo stadio dell'evoluzione la w non è stata emessa dalla M_1 e dalla M_2 rimane il dubbio se essa verrà emessa da una o da entrambe queste macchine generatrici.

Si osserva che nel caso M_1 emetta stringhe seguendo un definito ordine sequenziale algoritmico questa proprietà viene mantenuta dalla $\mathsf{M}_\setminus$. Tuttavia la richiesta dell'appartenenza alla lista.G generata di

una w può incontrare incertezze. Se si constata che l'evoluzione della M_1ha raggiunto la fase di possibile generazioe della w; se questa non si è verificata si decide che $w \notin \mathsf{M}_{\setminus}{}^{\mathcal{L}}$; se w è stata registrata su $\mathsf{U}_{\setminus}$ si deve attendere il momonto in cui la M_2 l'ha generata o in cui si può constatare che non la possa più generare, possibilità che dipende dall'eventuale conoscenza dell'ordine di generazione della M_2 stessa.

B18:**d.05** Introduciamo una MSPG $\mathsf{M}_{\ominus}$ che utilizza come sottomacchine due MSPG M_1 e M_2 per generare $MSd_1{}^{\mathcal{L}} \ominus MSd_2{}^{\mathcal{L}}$ e che chiamiamo **MSPG composizione per differenza simmetrica** delle macchine generatrici MSd_1 e MSd_1. Il suo nastro di uscita lo denotiamo con $U_{\ominus}$.

Per semplicità supponiamo ancora che ciascuna delle due sottomacchine da comporre proceda a emettere sul proprio nastro una lista nonripetitiva e che entrambe le macchine producano stringhe sullo stesso alfabeto A.

Anche la MSPG $\mathsf{M}_{\ominus}$ procede per manovre successive in ciascuna delle quali chiede alternatamente ad M_1 e ad M_2 di effettuare un passo evolutivo.
Queste due sottomacchine mantengono i rispettivi nastri di uscita U_1 e U_2.

Le due sottomacchine registrano ogni nuova stringa emessa sul proprio nastro e sul nastro $\mathsf{U}_{\ominus}$ La $\mathsf{M}_{\ominus}$ a ogni nuova emissione di una sottomacchina tenta un aggiornamento di $\mathsf{U}_{\ominus}$ che può consistere sia in una aggiunta che in una cancellazione.

Quando M_i emette una nuova stringa y_j, se essa non si trova in U_{3-i} viene emessa su U_i e su $\mathsf{U}_{\ominus}$, mentre se si trova su U_{3-i} viene trascurata, ossia non viene registrata su $\mathsf{U}_{\ominus}$ ed eventualmente viene cancellata da tale nastro.

Ciascuna delle stringhe prodotte da M_1 ma non da M_2 e ciascuna delle stringhe prodotte da M_2 ma non da M_1 verrà quindi a trovarsi su $\mathsf{U}_{\ominus}$.

Il linguaggio procedurale generato dalla $\mathsf{M}_{\ominus}$ evidentemente è la differenza simmetrica di $\mathsf{S}_1 := \mathsf{M}_1{}^{\mathcal{G}}$ e $\mathsf{S}_2 := \mathsf{M}_2{}^{\mathcal{G}}$.

Se una delle due macchine M_i produce una lista finita si deve continuare a far evolvere M_{3-i} per garantire che possano essere cancellate stringhe che M_i potrebbe avere registrate su $U_{\ominus}$.

Come per le precedenti costruzioni si trova che la costruzione della differenza simmetrica tra due insiemi procedurali va considerata una estensione della costruzione della differenza simmetrica di due insiemi finiti.

Valgono poi considerazioni analoghe sulla possibilità di avere sul nastro sul quale si costruisce $\mathsf{L}_1 \ominus \mathsf{L}_2$ una lista che in ogni fase della sua costruzione presenta un ordinamento algoritmico e sui problemi che si incontrano per stabilire se una data stringa w appartiene o meno ad $\mathsf{M}_{\ominus}{}^{\mathcal{G}}$.

B18:**d.06** Consideriamo una MSPG M e la lista.G L da essa generata e introduciamo una seconda MSPG $\mathsf{M}_{\complement}$ che chiamiamo MSPG ottenuta dalla M mediante **MSPG ottenuta per complementazione**. Si tratta di una macchina che usa M come sottomacchina e il cui nastro di uscita denotiamo con $USs_{\complement}$.

Può accadere che si voglia il complementare in un dato monoide libero A^*, o in mancanza di tale richiesta si assume si cerca il complementare in A^* ove A è un alfabeto del cui monoide libero $\mathsf{M}^{\mathcal{L}}$ fa parte.
Occorre far riferimento a un ordine di A e al corrispondente ordine-llx di A^*, relazione che in seguito chiamiamo "precedenza-llx", adottando anche i termini stringa massima-llx e stringa che segue-llx.

Anche la MSPG $\mathsf{M}_{\complement}$ procede per manovre successive in ciascuna delle quali o chiede alla M di effettuare un passo evolutivo, oppure effettua un passo per la produzione delle stringhe di A^* secondo la precedenza-llx.

Inizialmente il nastro $\mathsf{U}_{\lceil}$ è vuoto e rimane tale fino a che viene generata una prima stringa y_1 di L. Con questa stringa vengono inserite su $\mathsf{U}_{\lceil}$ tutte le stringhe di A^* che precedono-llx la y_1 (escludendo la y_1).

In seguito quando viene generata una nuova stringa $y_2 \in \mathsf{L}$, se questa precede-llx la massima-llx presente in $\mathsf{U}_{\lceil}$, che denotiamo con l_U, allora viene cancellata dal nastro di uscita.

Se invece y_2 segue-llx la l_U, si procede aggiungere a $\mathsf{U}_{\lceil}$ tutte le stringhe di A^* che seguono-llx la l_U precedono-llx la y_2.

A questo punto occorre distinguere se M genera una lista.G illimitata oppure una lista finita.

Nel primo caso si constata che ogni stringa generata da M o non viene mai registrata su $\mathsf{U}_{\lceil}$, o qualora fosse stata registrata viene sicuramente cancellata; si constata anche che ogni stringa di A^* che non viene generata da M_2 viene prima o poi scritta su $\mathsf{U}_{\lceil}$ senza la possibilità di esserne successivamente cancellata.

Nel caso in cui M_2 genera un insieme finito di stringhe, al momento in cui è accertato che non ne genererà altre la $\mathsf{M}_{\lceil}$ deve proseguire illimitatamente la produzione e l'emissione delle stringhe di A^* che seguono-llx le stringhe fino a qual moment presenti su $\mathsf{U}_{\lceil}$.

L'insieme generato da tale $\mathsf{M}_{\lceil}$ viene qualificato come **insieme-G cofinito**.

L'insieme procedurale generato da $\mathsf{M}_{\lceil}$, evidentemente, è l'insieme complementare entro A^* di L_2; esso viene denotato con $\mathsf{A}^* \setminus \mathsf{L}_2$ o anche, quando A può essere sottinteso, con ${\mathsf{L}_2}^{\complement}$.

Si osserva che il passaggio al complementare è una costruzione che si può considerare un caso particolare della eliminazione vista in a17: il ruolo di una macchina che genera L_1 è stato implicitamente assegnato a una sottomacchina che genera l'intero monoide libero A^*.

Si osserva che la macchina sopra prospettata genera liste nonripetitive che l'ordine-llx. Va però osservato che la possibilità che una lista registrata su $\mathsf{U}_{\lceil}$ contenga elementi di L che non sono ancora stati generati, potrebbe rendere problematico decidere se una data stringa $w \in \mathsf{A}^*$ appartenga o meno ad ${\mathsf{L}}^{\complement}$.

Vogliamo infine osservare esplicitamente che la differenza simmetrica di due linguaggi procedurali si può ottenere con le operazioni di unione, intersezione ed eliminazione, cioè che anche per gli insiemi procedurali vale l'uguaglianza

$$\mathsf{L}_1 \ominus \mathsf{L}_2 \;=\; (\mathsf{L}_1 \cup \mathsf{L}_2) \setminus \mathsf{L}_1 \cap \mathsf{L}_2) \;.$$

Questa si dimostra senza difficoltà osservando che una stringa viene emessa da una delle due macchine dichiarate equivalenti sse viene emessa anche dall'altra macchina.

B18:**d.07** Consideriamo due MSPG M_1 e M_2 in grado di produrre stringhe su due alfabeti non necessariamente uguali.

Ci proponiamo di introdurre una MSPG che denotiamo con $\mathsf{M}_1 \times \mathsf{M}_2$, che identifichiamo anche con $\mathsf{M}_\times$ e che chiamiamo **MSPG composizione per prodotto cartesiano** delle macchine date.

$\mathsf{M}_\times$ si serve come sottomacchine delle due date e dei loro nastri di uscita; sul suo nastro di emissione che denotiamo con $\mathsf{U}_\times$ emette la lista.G costituente il prodotto cartesiano delle liste.G $\mathsf{L}_1 := {\mathsf{M}_1}^{\mathcal{L}}$ ed $\mathsf{L}_2 := {\mathsf{M}_2}^{\mathcal{L}}$.

Supponiamo per semplicità che ciascuna delle M_i sul proprio nastro di uscita U_i produca una lista nonripetitiva.

La MSPG $\mathsf{M}_\times$ si serve anche di un nastro ausiliario; come le altre MSPG ottenute per composizione di due macchine che risultano più semplici $\mathsf{M}_\times$ procede per manovre successive in ciascuna delle quali chiede alternatamente ad M_1 e ad M_2 di effettuare un passo evolutivo.

Se inoltre le due sottomacchine in ciascuna delle loro fasi riescono a presentare una lista che segue un ordinamento sequenziale algoritmico, si può fare in modo che la $\mathsf{M}_\times$ proceda a costruire una lista nonripetitiva e che segue un ordinamento sequenziale algoritmico.

Consideriamo la fase dell'evoluzione nella quale M_1 ha generato e registrato su U_1 la lista $\langle x_1, x_2, ...x_h\rangle$ e nella quale sul nastro U_2 si trova la lista $\langle y_1, y_2, ...y_k\rangle$; sul nastro ausiliario, inizialmente vuoto si trova registrato il prodotto cartesiano di queste due liste finite, ossia

$$\Big\langle i = 1, ..., h; j = 1, ..., k clpp\langle x_i, y_j\rangle\Big\rangle \ .$$

Conviene pensare che le coppie di questo insieme siano disposte in una matrice a presentazione cartesiana con h righe e k colonne che chiamiamo matrice ausiliaria. Ad esempio se $h = 4$ 4 $k = 3$ abbiamo la matrice

$$\begin{array}{cccc} \langle x_1, y_3\rangle & \langle x_2, y_3\rangle & \langle x_3, y_3\rangle & \langle x_4, y_3\rangle \\ \langle x_1, y_2\rangle & \langle x_2, y_2\rangle & \langle x_3, y_2\rangle & \langle x_4, y_2\rangle \\ \langle x_1, y_1\rangle & \langle x_2, y_1\rangle & \langle x_3, y_1\rangle & \langle x_4, y_1\rangle \end{array} \ .$$

Nel nastro di uscita si deve invece avere la lista delle coppie $\langle x_i, y_j\rangle$ ottenuta scorrendo diagonalmente il triangolo inferiore a sinistra della matrice ausiliaria; nel caso $h = 5, k = 4$ su $\mathsf{U}_\times$ abbiamo:

$\langle x_1, y_1\rangle$, $\langle x_1, y_2\rangle$, $\langle x_2, y_1\rangle$,$\langle x_1, y_3\rangle$, $\langle x_2, y_2\rangle$, $\langle x_3, y_1\rangle$,$\langle x_1, y_4\rangle$, $\langle x_2, y_3\rangle$, $\langle x_3, y_2\rangle$, $\langle x_4, y_1\rangle$.

In generale abbiamo una lista di $\dfrac{m(m+1}{2}$ componenti, con $m := \min(h, k)$.

Nella fase nella quale la sottomacchina M_1 genera x_{h+1} lo si accoda si U_1 e si aggiunge alla matrice ausiliaria la colonna delle coppie $\langle x_{h+1}, y_j\rangle$; nella fase nella quale la sottomacchina M_2 genera y_{k+1} lo si accoda su U_2 e si aggiunge sulla matrice ausiliaria la riga delle coppie $\langle x_i, y_{k+1}\rangle$; in entrambi i casi possono rendersi necessari rimaneggiamenti atti a mantenere la riconoscibilità della struttura della matrice ausiliaria .

Se in una delle suddette fasi risulta aumentato m, il minimo tra numero di righe e numero di colonne della matrice sul nastro ausiliario la $\mathsf{M}_\times$ accoda su $\mathsf{U}_\times$ la sequnza delle coppie

$$\Big\langle\langle x_1, y_m\rangle, \langle x_2, , y_{m-1}2\rangle, ..., \langle x_m, y_1\rangle\Big\rangle \ ,$$

sequenza ricavabile senza difficoltà dalla matrice ausiliaria. JP Con queste manovre si riesce a generare la lista.G prodotto cartesiano.

I componenti di questa lista seguono l'ordine lessicografico determinato dai due ordinamenti stabiliti dalla L_1 e dalla L_2.

Se si volesse un ordinamento diverso si dovrebbe aggiungere un dispositivo che riordina il contenuto del nastro di uscita dopo ogni suo ampliamento con m nuovi componenti, lavoro che potrebbe essere molto dispendioso.

B18:**d.08** Le considerazioni dei precedenti paragrafi consentono il seguente enunciato.

(1) Teorema La collezione degli insiemi-G è chiusa rispetto alle operazioni di unione, intersezione, eliminazione, differenza simmetrica, complementazione e prodotto cartesiano ∎

Mediante gli operatori insiemistici visti in precedenza ed altri costrutti che vedremo in seguito, mediante simboli rappresentanti insiemi procedurali ottenibili da MSPG conosciute o con altre opportune notazioni e mediante coppie di parentesi coniugate che svolgono il ruolo dei delimitatori di sottoespressioni valutabili autonomamente, si possono comporre espressioni insiemistiche chiaramente interpretabili in termini costruttivi che estendono quelle viste in B11d14 in grado di individuare insiemi finiti.

A tali espressioni si aggiunge la qualifica di **formule ben formate**, termine che potrà essere meglio definito in seguito [C14e], quando disporremo di ulteriori strumenti per l'analisi di taluni linguaggi formali.

Osserviamo esplicitamente che ogni espressione ben formata $\mathcal{F}$ ottenuta con insiemi-G e con gli operatori insiemistici visti in precedenza (o con loro ragionevoli varianti) individua un insieme-G.

Questo fatto discende dalla considerazione che una espressione come la $\mathcal{F}$ riguarda un complesso Σ di operazioni binarie o unarie (organizzate ad arborescenza distesa come si dice in [C14f]) che agiscono sopra insiemi-G che fungono da operandi nell'espressione iniziale e che quindi sono riconducibili a ben definite MSPG, oppure a insiemi introdotti per individuare risultati di composizioni precedenti e quindi determinati da MSPG composite ottenute con le costruzioni trattate nei paragrafi precedenti.

Alla fine della esecuzione delle operazioni del complesso Σ, quindi, si riesce sempre a ottenere una MSPG che individua un insieme procedurale risultato, per quanto possa essere articolata la descrizione di tale macchina.

B18:**d.09** A questo punto quindi possiamo disporre di un genere di espressioni in grado di individuare una ampia gamma di insiemi-G.

Vogliamo sottolineare che le espressioni precedenti sono costituite da insiemi procedurali specifici, macchine MSPG specifiche e loro composizioni caratterizzate da operatori booleani.

Queste composizioni in effetti non sono completamente determinate dagli operatori booleani che le caratterizzano, in quanto devono essere dotate di dispositivi che richiedono scorrimenti, confronti e riordinamenti che possono essere piuttosto laboriosi. Tuttavia tutte queste composizioni si possono ricondurre a operazioni elementari riguardanti stringhe, nastri, contatori e informazioni su oggetti discreti sostanzialmente semplici.

Osserviamo anche che si incontrano molte altre collezione di espressioni ciascuna delle quali, come quelle viste in questa sezione, si basa su un numero finito di oggetti che possono considerarsi elementari e su un numero finito di composizioni degli oggetti individuati dalle espressioni considerate in precedenza, composizioni utilizzabili per ampliare quanto si vuole la collezione stessa.

Tra queste espressioni che abbiamo già incontrate segnaliamo le espressioni per stringhe e linguaggi, le espressioni su numeri interi, le espressioni riguardnti liste e le espressioni per insiemi finiti.

Molte altre collezioni di espressioni con caratteristiche simili le incontreremo nel seguito: espressioni per polinomi, espressioni per costruzioni di elementi di strutture algebriche, espressioni per funzioni di variabili reali e complesse, serie formali di potenze, espressioni per funzioni analitiche, per grafi, per automi, espressioni per calcoli su enunciati afferenti alla logica, ...

Queste collezioni di espressioni si riescono a trattare con strumenti della teoria dei linguaggi formali [v. in particolare C14] che nelle loro linee principali si possono considerare ben conosciuti a partire dagli anni 1970.

Con tali strumenti si riescono a esercitare vari controlli sulle espressioni dei vari generi e a gestire le relative collezioni.
Per molte di esse si riescono ad organizzare elaborazioni automatiche che consentono di utilizzarle con elevata efficienza sotto il controllo di programmi per computers e per altre apparecchiature automatiche che hanno ormai assunto lo status dei prodotti industriali.

B18:e. insiemi ricorsivi

B18:**e.01** Riprendiamo una importante proprietà che puèssere goduta o meno dagli insiemi infiniti, quella della risolvibilità del problema dell'appartenenza, proprietà che caratterizza gli insiemi che abbiamo chiamato insiemi-B [B08f11].

Consideriamo per questo un alfabeto finito A, il corrispondente monoide A^*, una MSPG **M** in grado di emettere sul proprio nastro di uscita U stringhe su A da assegnare al linguaggio-G che denotiamo con $\mathsf{L} := \mathbf{M}^{\mathcal{G}}$.

Poniamoci il **problema di appartenenza per una MSPG M**, che denotiamo con PRBmember[**M**, A], //Ne01 ossia il problema che per ogni $w \in \mathsf{A}^*$ chiede di decidere se $w \in \mathbf{M}^{\mathcal{G}}$ o meno. Ricordiamo che in inglese questo problema è detto "*membership problem*".

Facendo riferimento a questa precisa richiesta si parla di istanza del problema PRBmember[**M**, A] corrispondente alla stringa w, entità che denotiamo con PRBmember[**M**, A](w).

Un particolare tentativo di risolvere questa singola istanza di problema consiste nel verificare se la stringa w compare come membro della lista $\mathbf{M}^{\mathcal{L}}(t)$ che la **M** ha prodotto sul nastro U al termine della fase t della sua evoluzione.
A livello più generale il problema si può risolvere individuando un algoritmo in grado di decidere l'appartenenza a $\mathbf{M}^{\mathcal{G}}$ per qualsiasi stringa $w \in \mathsf{A}^*$.
Se si individua tale algoritmo si può enunciare che l'insieme-G $\mathbf{M}^{\mathcal{G}}$ è anche un insieme-B.

Il riconoscimente per ciascuna $w \in \mathsf{A}^*$ della sua occorrenza o meno nella lista.G $\mathbf{M}^{\mathcal{G}}$ si può chiamare **decisione del problema di occorrenza**.

Il problem di appartenenza può essere posto a elevati livelli di generalità per intere collezioni di linguaggi e per gruppi di alfabeti diversi; si può cercare di risolverlo anche per qualsiasi alfabeto.

Vedremo che i problemi di appartenenza si collegano sia alla possibilità di generare liste.G che seguono un determinato ordinamento algoritmico $\Omega_{\mathcal{K}}$ basato sopra un algoritmo di confronto $\mathcal{K}$, sia alla possibilità di generare liste.G ottenibili mediante sequenzializzazione.

Si constata che il problema di appartenenza si risolve facilmemnte per l'insieme-G dei numeri naturali $\mathbb{N}$ e per ogni insieme-G della forma A^*, quale che sia alfabeto A.
Per questo basta effettuare lo scorrimento della stringa da esaminare e per ciascuno dei caratteri che la compongono verificare che coincide con ı o che appartiene ad A.

Gli insiemi-G per i quali risulta decidibile il problema dell'appartenenza si dicono anche **insiemi-GB**, **insiemi ricorsivi** o anche **linguaggi ricorsivi**.

B18:**e.02** Gli insiemi-GB ambienti basilari $\mathbb{N}$ e A^* possono presentare le liste.GI dei loro elementi secondo ordinamenti sequenziali ben riconoscibili: l'ordinamento $\leq$ per gli interi naturali e l'ordinamento len-lxg per l'insieme di tutte le stringhe sopra l'alfabeto A.

Più in generale un qualsiasi insieme-G per il quale si individua un algoritmo che trasforma un suo qualsiasi elemento in un successivo si dice che gode della **sequenziabilità**.

Questa proprietà degli insiemi-G equivale alla decidibilità del problema dell'appartenenza; questa equivalenza viene garantita dai due enunciati che seguono.

(1) Prop.: Ogni lista.GI di stringhe sull'alfabeto A per la quale si conosca un ordinamento algoritmico di A^* definisce un insieme-GB.

Dim.: Denotiamo con M una MSPGI che può generare la lista.GI in causa che quindi si può scrivere $\mathsf{L} = \mathsf{M}^{\mathcal{G}}$; denotiamo inoltre con $\mathcal{K}$ l'algoritmo di confronto e con $\preceq_{\mathcal{K}}$ l'ordinamento algoritmico derivato da questo meccanismo di confronto.
Un tipico ordinamento di questo genere è un ordinamento len-lex di A associato da un ordinamento totale dei caratteri di A.

Si tratta di individuare un algoritmo che in un numero finito di passi data una qualsiasi $w \in \mathsf{A}^*$ decide se compare nella L.

Questo algoritmo procede a generare tutte le stringhe di A^* secondo $\preceq_K$, trascura tutte le stringhe che precedono la w e sa riconoscere se viene generata la w, caso in cui dichiara la sua appartenenza a L, oppure si accorge di aver generata una stringa successiva alla w, caso in cui dichiara la sua nonappartenenza a L ∎

Vale anche una proprietà reciproca.

Prop. (2) Ogni lista.GI L di stringhe su A che rappresenta un insieme-B, lista.GI per la quale è decidibile il problema dell'occorrenza, può essere emessa secondo un ordinamento algoritmico .

Dim.: Basta precisare la MSPGI che procede a scorrere le stringhe sull'alfabeto A secondo l'ordinamento len-lxg ed emette solo quelle che l'algoritmo dell'occorrenza dice occorrere nella L ∎

B18:**e.03** La risoluzione di una istanza di problema di appartenenza si può vedere, in completa generalità, come la individuazione di una MSPA che denotiamo con $\mathbf{D}$ e che appartenga al genere delle cosiddette **MSP dicotomiche** .

Una tale macchina ha il compito di eseguire un cosiddetto **algoritmo dicotomico**: essa è caratterizzata da avere un solo nastro di uscita U ridotto a un unico registro binario sul quale alla fine di una elaborazione ci si aspetta uno di due segni: un segno con significato positivo (potrebbe essere `true`, `yes` o 1) ed un segno con significato negativo (potrebbe essere , rispetivamente, `false`, `no` o 0) al quale va attribuito il valore opposto a quello del precedente.

Nell'ambito di una attività per la soluzione di un problema ogni segno che viene fornito da un algoritmo dicotomico può essere usato come fattore determinante per la decisione da effettuare di fronte a una scelta tra due manovre successive; le scelte di questo genere sono qualificate come **scelte dicotomiche**.

Una MSP dicotomica M_D utilizzata per risolvere un problema di appartenenza per le liste.G generati dalle macchine costituenti un insieme M deve disporre di due nastri di ingresso: il primo nastro è in grado di accogliere una descrizione di una generica MSPG $\mathbf{P} \in \mathsf{M}$, che è opportuno presumere sia una MSPGI il cui alfabeto di uscita denotiamo con $\mathsf{A}_{\mathbf{P}}$; il secondo nastro ha il compito di presentare una $w \in \mathsf{A}_{\mathbf{P}}{}^*$ che può essere il frutto di una ipotesi o provenire da qualche elaborazione preliminare.

L'algoritmo che decide l'appartenenza $w \in \mathbf{P}^{\mathcal{G}}$? ha il compito di emettere sul suo nastro/registro di uscita U, dopo una elaborazione finita, il segno con significato positivo sse se ha riscontrato l'appartenenza e il segno con significato opposto nel caso contrario.

Conviene rilevare esplicitamente che si è avanzata la richiesta di utilizzare la descrizione di una MSP sulla quale non si dichiara alcuna restrizione per definire il funzionamento di una MSPA che svolge una **attività simulatrice**. Questa richiesta richiede delle giustificazioni che a questo punto ci limitiamo a presentare a un livello intuitivo e fiduciario.
Delle MSP non abbiamo dato una definizione che in sostanza consiste solo nella richiesta, fiduciaria, della possibilità di dare per ciascuna di tali macchine una descrizione che la rendesse nonambigua e pienamente utilizzabile, cioè tale che un opportuno esecutore sia in grado di ricavare una adeguata

parte iniziale per ciascuna delle sue possibili evoluzioni (non essendo sensato affermare di disporre effettivamente di una sua completa evoluzione).

A questo punto potremmo chiedere più formalmente, che ogni MSP sia definita esaurientemente con una stringa sopra un opportuno alfabeto A_M in grado di esprimere precise regole di comportamento per ogni automatismo proposto come affidabile.

Tuttavia qui ci limitiamo a segnalare alcuni tipi di stringhe che definiscono MSP che incontreremo tra poco:
– le descrizioni dei trasduttori a stati finiti [C13];
– le descrizioni mediante grammatiche dei meccanismi generativi di linguaggi formali [C14];
– le definizioni di macchine di Turing [C21].

Ricordiamo inoltre i testi sorgente degli odierni programmi per computers scritti in uno dei moltissimi linguaggio di programmazione (C++, Java, PHP, linguaggi per elaborazioni simboliche, ...).

Dobbiamo inoltre promettere che verranno definiti solo modelli M di MSP tali che a ciascuno di essi, si possa associare una MSPA in grado di stabilire se una qualsiasi stringa sull'alfabeto $\mathsf{A}_{\mathcal{M}}$ utilizzabile per definire le caratteristiche di una qualche MSP descrive esaurientemente e correttamente una macchina del genere M o meno.
Si viene quindi a prospettare una MSPGI in grado di procedere alla generazione di tutte le descrizioni utilizzabili da opportuni esecutori delle MSP di un certo genere M e quindi la disponibilità di ciascuna delle macchine di tale genere attraverso la stringa che la definisce e che determina il suo comportamento.

B18:**e.04** Prima di procedere è opportuno dire di più sulle nozioni di problema e delle sue istanze, focalizzando l'attenzione sui problemi di appartenenza.

Una **istanza del problema di appartenenza** consiste nella decisione per una specifica MSPG M che emette stringhe su un alfabeto A_{M} e per una particolare stringa $w \in \mathsf{A}_{\mathsf{M}}{}^*$ se w è una occorrenza della lista.GI $\mathsf{M}^{\mathcal{L}}$ o meno.
Il problema della appartenenza in una forma più generale richiede una decisione per una qualsiasi procedura $\mathbf{P}$ facente parte di una data collezione di procedure $\mathsf{MSPG}\mathfrak{X}$ e per qualsiasi stringa sul corrispondente alfabeto di uscita, $\mathsf{A}_{\mathbf{P}}$.

Dunque consideriamo che un problema di appartenenza risulti definito con chiarezza quando sono dati un esemplare $\mathsf{MSPG}\mathbf{X}$, o più operativamente una MSPA $\mathsf{M}_{\mathfrak{X}}$ in grado di decidere se una descrizione di macchine MSPG riguarda o meno un membro di $\mathsf{MSPG}\mathbf{X}$, e un alfabeto A adatto a esprimere le stringhe generate da membri di $\mathsf{MSPG}\mathbf{X}$; tale problema si può individuare con la scrittura

$$\mathsf{PRBmember}[\mathsf{MSPG}\mathbf{X}, \mathsf{A}] \ .$$

Spesso l'alfabeto A riveste importanza secondaria, in quanto la fondamentale intercambiabilità dei segni adottati nelle formalizzazioni consente di avere formulazioni con alfabeti diversi di problemi essenzialmente equivalenti.

Inoltre l'alfabeto di uscita di una particolare MSPG può essere incluso nella stringa articolata che la definisce e le analisi delle formulazioni delle macchine di questo tipo possono essere condotte in modo marcatamente parametrico.

In genere alcune istanze di un problema di appartenenza sono facili da risolvere, talora tanto banali da consentire presentazioni intuitive che non si soffermano su una meticolosa formalizzazione.
In particolare molti dei problemi di appartenenza concernenti MSPG che generano insiemi finiti sono poco impegnativi e poco interessanti.

Al contrario le considerazioni sulle situazioni difficili da tenere sotto controllo per le composizioni di procedure MSPGI esposte in :a inducono ad aspettarsi che i problemi di appartenenza riguardanti alcune classi di MSPGI possano presentare serie difficoltà.

A questo proposito conviene far presente che, a causa della grande varietà delle prestazioni che richiediamo alle MSPGI, risulta interessante riuscire ad esprimersi sopra una grande varietà di problemi di appartenenza.
Va anche segnalato che molti problemi che si incontrano nella matematica discreta e nelle sue applicazioni si possono presentare come problemi di appartenenza.

Consideriamo un insieme di MSPG $\mathsf{MSPGI}\mathfrak{Y}$ che sia sottoinsieme proprio di $\mathsf{MSPGI}\mathfrak{X}$; si osserva che questo insieme di macchine si può confondere con il linguaggio delle loro definizioni, linguaggio che può essere sottoposto a regole precise e quindi si può trattare come un insieme procedurale.
Il relativo problema di appartenenza $\mathsf{PRBmember}[\mathsf{MSPG}\mathfrak{Y}]$ è lecito chiamarlo **sottoproblema** del precedente $\mathsf{PRBmember}[\mathsf{MSPG}\mathfrak{X}]$.

Evidentemente anche nello studio sistematico dei problemi di appartenenza è opportuno e significativo cercare di risolvere problemi di ampia portata.
Si trova tuttavia che questa aspettativa incontra varie difficoltà ed anche, come accennato, limiti invalicabili.
In effetti incontreremo problemi di appartenenza della forma $\mathsf{PRBmember}[\mathsf{MSPG}\mathfrak{Z}, \mathsf{A}]$ per i quali si dimostra non esistere alcun algoritmo dicotomico in grado di decidere l'appartenenza $w \in \mathsf{M}^{\mathcal{G}}$ per tutte le $w \in \mathsf{A}^*$.
In questi casi può essere conveniente porsi la questione di trovare suoi sottoproblemi i cui relativi problemi di appartenenza siano decidibili.
B18:**e.05** Abbiamo visto che sono insiemi numerabili l'insieme delle notazioni unadiche degli interi naturali $|^*$ e più in generale gli insiemi A^* delle stringhe sopra un qualsiasi alfabeto finito A.

Conviene segnalare che risulta conveniente considerare solo insiemi numerabili che si possono collocare come sottoinsiemi di un monoide libero su un alfabeto finito ben esplicitato o concordemente giudicato come ben esplicitabile.
In molte considerazioni si fa riferimento ai caratteri di uno solo di questi alfabeti e tali segni nel proseguire delle argomentazioni si possono tranquillamente chiamare caratteri in uso. Abbiamo anche visto che altri insiemi ambiente rilevanti si ottengono con composizioni insiemistiche di insiemi ambiente primari ed eventualmente con l'intervento di algoritmi selettivi, molti dei quali sono stati ampiamente adottati, taluni da secoli.

Inoltre sono insiemi numerabili che si rivelano utili vari insiemi-GI generati secondo un ordinamento sequenziale algoritmico. Questi li chiamiamo insiemi generabili illimitatamente e progressivamente o, in breve, **insiemi-GIp**.

Si dice **insieme progressivamente generato** un insieme-G per il quale si può precisare, oltre alla MSPG che lo genera, una MSPA in grado di decide il problema del confronto delle sue stringhe.
Viceversa un insieme-GI generato secondo un ordinamento sequenziale algoritmico è generabile progressivamente.

Ricordiamo inoltre che viene spesso usato il termine **insieme contabile** per caratterizzare un insieme che può essere sia finito che numerabile.

B18:**e.06** Si possono individuare numerosi insiemi ricorsivi costituiti da interi naturali che presentano rilevante interesse.

Questi insiemi si possono presentare come successioni della forma $\langle n \in \mathbb{N} :| \mathcal{E}_n \rangle$, dove $\mathcal{E}_n$ denota un'espressione che per ogni naturale n si sa valutare senza difficoltà ricavando un intero naturale che quindi risulta dipendente dallo stesso n.
In particolare si hanno espressioni nelle quali possono comparire come operandi, oltre ad n, altri numeri naturali specifici forniti da scritture decimali o scritture loro equivalenti, e operatori come somma, prodotto, differenza, quoziente, resto, massimo, minimo, MCD, mcm, e altre funzioni definite in modo da garantire che possano effettivamente fornire degli interi naturali.
Alcune di queste funzioni sono state già introdotte, ad esempio le funzioni $\mathsf{bsum}(n)$ [a01)] e $\mathbb{N}_{[B]}$ per ogni $B = 2, 3, 4...,$.
Altre saranno definite in seguito, come la funzione totient di Eulero definita in B26d03 e le varie successioni presentate in D20: successione dei numeri di Fibonacci $\mathsf{Fib}(n)$, successione dei numeri di Catalan Ctln_n, successione dei numeri di Lucas $\mathsf{Luc}(n)$, ...
Accanto a queste saranno definite funzioni su due variabili intere come i coefficienti binomiali e le successioni doppie di Lah $\mathsf{nlah}(n, k)$ [D20h02].

Anche l'espressione $\mathcal{E}_n$ deve essere "ben formata", cioè deve seguire regole sintattiche che esamineremo più avanti [D30, C14], le quali sono in grado di renderla interpretabile dagli appositi algoritmi chiamati algoritmi di `parsing (we)`, meccanismi che riescono a garantire con efficienza e accuratezza se una espressione è ben formata e in tal caso consentono di calcolarla.

Esempi di presentazioni di queste successioni sono

$$\Big\langle n \in \mathbb{N} :| \; 5 + 3\,n^2 + n^3 \Big\rangle \quad , \quad \Big\langle n \in \mathbb{N} :| \; n! + \min\big(8\,n, \mathsf{nlah}(2\,n, n)\,7\,\mathsf{bsum}(3\,n)\big) + n\%3 \Big\rangle \; .$$

$$\Big\langle n \in \mathbb{N} :| \; \max\big(\mathsf{Fib}_{2\,n}, 7 + 2\,n\%3\big) \Big\rangle \; .$$

Più in generale si possono presentare espressioni della forma $\Big\langle n \in \mathbb{N} :| \; \mathsf{T}(n) \Big\rangle$, nella quale T rappresenta una MSP con compiti di trasduttore da $\mathbb{N}$ in $\mathbb{N}$, cioè una MSPTA che costituisca una rappresentazione algoritmica di una funzione calcolabile del genere $\lceil\, \mathbb{N} \longmapsto \mathbb{N} \,\rceil$.

Come si è accennato, ciascuna delle espressioni nelle presentazioni precedenti conduce a una MSPA; in molti casi l'espressione è più comprensibile della definizione della MSPA.

Altre importanti successioni di interi naturali si possono definire mediante sistemi di uguaglianze di ricorrenza come quelli con i quali vengono introdotti i numeri di Fibonacci [D20d] e i numeri di Catalan [D20e].

Va inoltre segnalato fin d'ora che risulta molto utile servirsi di espressioni riguardanti altri insiemi numerabili di numeri, come i numeri interi relativi [B20a], i numeri razionali [B30], i numeri algebrici B38, i numeri reali calcolabili [B42] e i numeri complessi calcolabili [B50].
Per questi insiemi numerici, che abbiamo segnalati in ordine di estensione crescente, si possono introdurre molte utili operazioni e si possono assumere come domini di una ampia varietà di funzioni speciali.

Per le successioni numeriche specifiche è quindi disponibile un'ampia varietà di espressioni e di algoritmi che consentono molteplici calcoli e svariate dimostrazioni di proprietà.

B18:**e.07** Come abbiamo accennato, tra gli insiemi numerabili vanno considerati anche i linguaggi numerabili che possono essere utilizzati per definire le stesse MSP dei vari generi.

Più in generale ogni genere di struttura matematica introdotta per finalità costruttive deve essere definita in modo preciso ed esauriente; quindi anche per ciascuna struttura specifica di qualsiasi genere

dovrebbe potersi individuare una MSPA in grado di decidere se una stringa su un opportuno alfabeto descrive una MSP di una data specie $\mathbb{S}$ o meno.

Di conseguenza si riesce a individuare anche una MSPGI in grado di procedere nella generazione dell'insieme numerabile delle strutture di ogni ben definita specie $\mathbb{S}$.

Ad esempio non è difficile individuare un linguaggio che consenta la presentazione di ciascuno dei numerosi gruppi finitidi largo uso [B41b, T22] mediante le rispettive tavole di moltiplicazione e dei linguaggi più stringenti per le tavole di moltiplicazione dei gruppi finiti abeliani, dei gruppi finiti ciclici e anche per i gruppi finiti semplici (va anche detto che un linguaggio con tale finalità deve saper esprimere una popolazione di oggetti assai complessa).

Similmente si possono individuare linguaggi per la presentazione mediante matrici delle adiacenze dei digrafi in genere [B14, D27] e in particolare per i digrafi aciclici, per i digrafi connessi, per le arborescenze, ... Altri linguaggi sono pensabili per configurazioni come poliedri, disegni a blocchi, codici, nodi, ...

B18:**e.08** Ci proponiamo ora di dimostrare costruttivamente che alcune composizioni di due insiemi numerabili (unione, intersezione, eliminazione, differenza simmetrica, complementazione) forniscono nuovi insiemi numerabili.

Per ogni composizione prendiamo in considerazione due MSPGI, M_1 e M_2, l'alfabeto A sul quale sono costruite le stringhe dei due rispetivi insiemi numerabili generati $\mathsf{N}_1 := {\mathsf{M}_1}^{\mathcal{G}}$ e $\mathsf{N}_2 := {\mathsf{M}_2}^{\mathcal{G}}$, e le macchine MSPA $\mathbf{D}_1$ e $\mathbf{D}_2$ in grado di decidere i rispettivi problemi di appartenenza.

Ci avvarremo anche, per $i = 1, 2$, di due ordinamenti sequenziali algoritmici per gli insiemi N_i che denotiamo con $\preceq_i$; ci serviremo inoltre dei corrispondenti ordinamenti riflessi $succeq_i$ e delle loro riduzioni nonriflessive $\prec_i$ e $\succ_i$.

(1) Prop.: L'unione dei due insiemi numerabili è un insieme numerabile.

Dim.: Si tratta di descrivere una MSPA $\mathbf{D}$ in grado di decidere in un numero finito di passi se una qualsiasi $w \in \mathsf{A}^*$ appartiene o meno ad $\mathsf{N} := NSs_1 \cup \mathsf{N}_2$.

Per questa $\mathbf{D}$, come per le MSPA decisionali che vedremo per le altre composizioni di insiemi numerabili, assumiamo che si tratti di una macchina in grado di usare come sottomacchine sia la M_1 che la M_2.

La decisione sull'appartenenza della generica w ad N si ottiene in una o due manovre.
Nella prima si simula $\mathbf{D}_1$ per stabilire se $w \in \mathsf{N}_1$; in caso affermativo si decide per l'appartenenza $w \in \mathsf{N}$ e si conclude l'esecuzione.

In caso di risposta negativa si simula $\mathbf{D}_2$ per stabilire se $w \in \mathsf{N}_2$ e dopo un numero finito di passi si ottiene la decisione della dalla M_2 e la si applica come risposta al quesito $w \in \mathsf{N}$? ∎

B18:**e.09** In qualche attività pratica potrebbe porsi il problema di disporre di tutti gli elementi di $\mathsf{N}_1 \cup \mathsf{N}_2$ di lunghezza inferiore o uguale a uno specifico intero positivo $\overline{n}$.
Chiaramente è garantito che per ogni $\overline{n} \in \mathbb{P}$ si riesce a ottenere questo elenco finito mediante una sequenza finita di passi, in quanto è possibile controllare quando sia M_1 che M_2 hanno esaurito la generazione delle stringhe con lunghezza minore o uguale a $\overline{n}$, anche se i valori emessi dalle due macchine crescono molto diversamente.

A questo proposito può essere utile osservare l'esempio delle seguenti successive istantanee del nastro di uscita di una macchina che fornisce l'unione dell'insieme dei numeri primi dispari e dell'insieme dei multipli positivi di 6:

3 6 , 3 5 6 , 3 5 6 12 , 3 5 6 7 12 , 3 5 6 7 12 18 , 3 5 6 7 11 12 18

, 3 5 6 7 11 12 18 24 , 3 5 6 7 11 12 13 18 24 , 3 5 6 7 11 12 13 18 24 36
, 3 5 6 7 11 12 13 17 18 24 36 , 3 5 6 7 11 12 13 17 18 24 36 42 , . . .

Si può quindi concludere che ogni unione di due insiemi numerabili, della forma $\mathsf{N}_1 \cup \mathsf{N}_2$, può essere generato secondo un ordinamento sequenziale (come l'ordinamento len-lex nell'ambito di un A^*).

Accanto alla proposizione e08(1) conviene considerare le seguenti sue generalizzazioni.

(1) Prop.: L'unione di una collezione finita di insiemi numerabili $\cup_{i=1}^{m} \mathsf{N}_i$ è un insieme numerabile.

Dim.: Per decidere l'appartenenza all'unione di una stringa w si fa uso di una macchina decisionale D in grado di servirsi in succesione di tutte le macchine decisionali parziali che identifichiamo, risp., con D_1, D_2, ... e D_m.

Questo lavoro procede fino a quando si trova una D_i che accetta la w in modo da concludere che $w \in \mathsf{N}_i$, oppure quando risulta accertato che tutte le m macchine rifiutano w e di conseguenza si decide che $w \notin \mathsf{N}$ ∎

(2) Coroll.: L'unione disgiunta di due insiemi numerabili ${\mathsf{M}_1}^{\mathcal{G}}$ e ${\mathsf{M}_2}^{\mathcal{G}}$ è un insieme numerabile.

Dim.: Basta arricchire la costruzione descritta per b7(1) adottando un nastro di uscita U in grado di registrare coppie della forma $\langle w, j \rangle$ dove w esprime ogni stringa prodotta da una delle due macchine e $j(= 1, 2)$ è l'indice della macchina da cui proviene la w attuale. In tal modo due stringhe coincidenti ma provenienti da due sottomacchine diverse possono essere considerate diverse ∎

Supposto che le due liste.GI da comporre seguano lo stesso ordine algoritmico il nastro di uscita può essere riorganizzato dopo l'aggiunta di ogni nuova coppia $\langle w, j \rangle$ per garantire l'ordine sequenziale che vede prevalere l'ordine dei primi membri e nel caso di uguaglianza di questi decide, ad esempio, che l'indice di macchina 1 precede l'indice 2.

Accanto alla proposizione b08(1) conviene considerare le seguenti come sue generalizzazioni.

(3) Prop.: L'unione di un insieme numerabile e di un insieme finito è un insieme numerabile.

Dim.: Si fa uso di una macchina che all'inizio simula la macchina che genera l'insieme finito e che dopo aver esaurito questo insieme simula la macchina MSPGI che genera l'insieme numerabile ∎

Anche per una tale composizione quando l'insieme numerabile viene generato secondo un ordinamento sequenziale si ha la possibilità di ottenere un ordinamento sequenziale per l'insieme unione.

B18:**e.10 Prop.** L'intersezione di due insiemi numerabili N_1 ed N_2 è un insieme contabile, ossia numerabile o finito.

Dim.: Si tratta di descrivere una MSPA $\mathbf{D}$ in grado di decidere in un numero finito di passi se una qualsiasi $w \in \mathsf{A}^*$ appartiene a entrambi gli insiemi numerabili NSs_1 ed N_2.

Anche per questa $\mathbf{D}$ si chiede che si tratti di una macchina in grado di servirsi sia della M_1 che della M_2.

La decisione sull'appartenenza della generica $w \in \mathsf{A}^*$ ad N si ottiene in una o due manovre.

Nella prima si richiama $\mathbf{D}_1$ per stabilire se $w \in \mathsf{N}_1$; in caso negativo si decide per la non appartenenza $w \notin \mathsf{N}$ e si conclude l'esecuzione.

In caso di risposta positiva occorre richiamare $\mathbf{D}_2$ per stabilire se $w \in \mathsf{N}_2$ e dopo un numero finito di passi si conclude con la stessa decisione alla quale perviene la M_2 ∎

Se uno dei due insiemi da intersecare, chiamiamolo N_1, è finito, evidentemente anche l'intersezione deve essere un insieme finito.

Per quanto riguarda la generazione di $\mathsf{N}_1 \cap \mathsf{N}_2$ In termini operativi occorre dire che solo se a un certo punto dell'elaborazione si riesce a stabilire che non si avranno altre emissioni della macchina N_1 si può decidere la fine delle operazioni della M.
Come si possa stabilire l'esaurimento delle emissioni utili della N_1 dipende dalle particolarità della N_1 e dall'andamento delle esecuzioni.

B18:**e.11** **(1) Prop.**: L'eliminazione di un insieme numerabile da un insieme numerabile è un insieme contabile.

Dim.: Si tratta di descrivere una MSPA $\mathbf{D}$ in grado di decidere in un numero finito di passi se una qualsiasi $w \in \mathsf{A}^{\mathcal{G}\mathrm{en}}$ appartiene a $\mathsf{N} := \mathsf{N}_1 \setminus \mathsf{N}_2$, cioè appartiene a NSs_1 e non fa parte di N_2.

Anche per questa $\mathbf{D}$ si chiede che si tratti di una macchina in grado di servirsi sia della M_1 che della M_2.

La decisione sull'appartenenza della generica w ad N si ottiene con una o due manovre.

Nella prima si richiama $\mathbf{D}_1$ per stabilire se $w \in \mathsf{N}_1$; in caso negativo si decide per la non appartenenza $w \notin \mathsf{N}$ e si conclude l'esecuzione.

In caso di risposta positiva occorre invocare la $\mathbf{D}_2$ per stabilire se $w \in \mathsf{N}_2$ e dopo un numero finito di passi si può decidere: se w appartiene anche a N_2 viene rifiutata per N, se w non fa parte di N_2 la si dichiara elemento di N ∎

B18:**e.12** **(1) Prop.**: Il complementare di un insieme numerabile $\mathsf{M}^{\mathcal{G}}$ è un insieme contabile.

Dim.: Si tratta di descrivere una MSPA $\overline{\mathbf{D}}$ in grado di decidere in un numero finito di passi se una qualsiasi $w \in \mathsf{A}^*$ non fa parte di $\mathsf{N} := \mathsf{M}^{\mathcal{G}}$, cioè se $w \in \mathsf{N}^{\complement}$.

A questa $\overline{\mathbf{D}}$ si chiede di sapere invocare la $\mathbf{D}$.

La decisione sull'appartenenza della generica w ad N si ottiene chiedendo alla $\mathbf{D}$ di stabilire se $w \in \mathsf{N}$.

In caso di esito negativo si decide di accettare w in $\overline{\mathsf{N}}$; se all'opposto si trova che w appartiene a N, la stringa viene considerata estranea a $\overline{\mathsf{N}}$ ∎

(2) Prop.: La differenza simmetrica di due insiemi numerabili $\mathsf{N}_1 \ominus \mathsf{N}_2$ è un insieme contabile (cioè numerabile o finito).

Dim.: Si tratta di descrivere una MSPA $\mathbf{D}$ in grado di decidere in un numero finito di passi se una qualsiasi $w \in \mathsf{A}^*$ appartiene a uno e uno solo dei linguaggi NSs_1 ed N_2, cioè se $w \in (\mathsf{N}_1 \ominus \mathsf{N}_2)$.

Anche per questa $\mathbf{D}$s si chiede che sia una macchina in grado di invocare sia la $\mathbf{D}_1$ che la $\mathbf{D}_2$.
La decisione sull'appartenenza della generica w ad N si ottiene con tre manovre. Nella prima si invoca $\mathbf{D}_1$ per stabilire se $w \in \mathsf{N}_1$ e per entrambi gli esiti si invoca $\mathbf{D}_2$ per stabilire se $w \in \mathsf{N}_2$.
Con la mamovra conclusiva si decide che $w \in \mathsf{N}$ sse si è trovato che w appartiene a uno solo dei due linguaggi N_1 ed N_2, mentre si decide che $w \notin \mathsf{N}$ sse w appartiene sia a N_1 che a N_2, oppure non appartiene ad alcuno dei due linguaggi ∎

B18:**e.13** **(1) Prop.**: Il prodotto cartesiano di due insiemi numerabili è un insieme numerabile.

Dim.: Consideriamo due MSPGI M_1 e M_2 che generano gli insiemi numerabili N_1 e N_2 facenti parte, risp., del monoide ${\mathsf{A}_1}^*$ e del monoide ${\mathsf{A}_2}^*$ e rappresentiamo con $\mathbf{D}_1$ e $\mathbf{D}_2$ i corrispondenti algoritmi in grado di decidere le rispettive appartenenze.

Per la MSPGI M che genera l'insieme prodotto cartesiano chiediamo di servirsi, oltre che del proprio nastro di uscita U, delle macchine M_1 e M_2 compresi i rispettivi nastri di uscita U_1 e U_2 e di un nastro ausiliario .

La generazione di $\mathsf{N}_1 \times \mathsf{N}_2$ procede come in d07.

Si tratta poi di descrivere una MSPA $\mathbf{D}$ in grado di decidere in un numero finito di passi se una qualsiasi coppia $\mathsf{w} := \langle w_1, w_2 \rangle$ che appartiene a $\mathsf{A}_1^* \times \mathsf{A}_1^*$ presenta come primo membro un elemento di N_1 e come secondo un elemento di N_2.

Anche per questa $\mathbf{D}$s si chiede che si tratti di una macchina in grado di invocare sia la $\mathbf{D}_1$ che la $\mathbf{D}_2$.

La decisione sull'appartenenza della generica coppia $\mathsf{w} = \langle w_1, w_2 \rangle$ ad $\mathsf{N} := \mathsf{N}_1 \times \mathsf{N}_2$ si ottiene con due manovre.

Con la prima si invoca $\mathbf{D}_1$ per stabilire se $w_1 \in \mathsf{N}_1$. In caso negativo si conclude che w non appartiene al prodotto cartesiano N.

In caso positivo si invoca $\mathbf{D}_2$ per stabilire se $w_2 \in \mathsf{N}_2$ e in seguito a risposta positiva si decide $\mathsf{w} \in \mathsf{N}$, mentre in seguito a risposta negativa si conclude con la decisione opposta ∎

Il risultato precedente si generalizza senza difficoltà ricorrendo più volte all'enunciato (1).

(2) Prop.: Il prodotto cartesiano di una sequenza finita di insiemi numerabili è un insieme numerabile ∎

B18:**e.14** Si pone il problema di definire liste.GI di macchine o altre entità con strutture articolate definite a partire da prestazioni loro richieste.
Viste queste richieste occorre individuare un alfabeto $\boldsymbol{A}$ tale che le sue stringhe siano in grado di esprimere completamente le entità desiderate.
Occorre inoltre individuare un algoritmo decisionale $\boldsymbol{D}$ in grado di stabilire se una stringa su $\boldsymbol{A}$ esprime una entità desiderata o meno.

La MSPG per le entità desiderate deve procedere a generare le stringhe su $\boldsymbol{A}$ e applicare a ciascuna di esse l'algoritmo $\boldsymbol{D}$ che decide se ciascuna stringa esprime una entità desiderata.

Se si possono individuare $\boldsymbol{A}$ e $\boldsymbol{D}$ è lecito parlare dell'insieme numerabile delle entità desiderate. Si osserva che si possono individuare MSPG che generano liste.G di stringhe esprimenti macchine che presentano equivalenze-set e che non è facile individuare macchine che evitino queste equivalenze.

Si può porre anche il problema dell'essere numerabile un insieme unione di una sequenza numerabile di insiemi ricorsivi ottenuta da una sequenza numerabile di MSPG esprese in forma riconoscibile in un numero finito di passi.
Anche questo si ottiene con una costruzione diagonale alla Cantor [B30a07].

B18:f. relazioni-G, funzioni-G, relazioni ricorsive e funzioni ricorsive

B18:**f.01** Si possono definire molte MSPGI che generano liste.G di coppie e quindi definiscono insiemi-G di coppie di grande interesse. Questi insiemi di coppie sono qualificabili sia come relazioni procedurali [a05] che come insiemi ricorsivi e quindi si possono chiamare **relazioni [binarie] ricorsive**.

Per concretezza è opportuno segnalare alcune macchine MSPGI facilmente comprensibili che portano a relazioni ricorsive prevedibilmente interessanti.

(1) Macchina che emette le coppie $\langle n, n\rangle$ per i successivi $n = 0, 1, 2, ...$

(2) Macchina che emette le coppie costituite da un numero naturale $n \in \mathbb{N}$ e dal corrispondente n-esimo numero primo che denotiamo con p_n; più precisamente definiamo la macchina che procede a emettere la lista.GI

$$\Big\langle \langle 0, 1\rangle, \langle 1, 2\rangle, \langle 2, 3\rangle, \langle 3, 5\rangle, ..., \langle 4, 7\rangle, ..., \langle n, p_n\rangle, ... \Big\rangle \ .$$

(3) Macchina che genera le successive coppie $\langle 2, 1\rangle$, $\langle 3, 1\rangle$, $\langle 4, 2\rangle$, $\langle 4, 1\rangle$, $\langle 5, 1\rangle$, $\langle 6, 3\rangle$ $\langle 6, 2\rangle$ $\langle 6, 1\rangle$, $\langle 7, 1\rangle$, $\langle 8, 4\rangle$, $\langle 8, 2\rangle$, $\langle 8, 1\rangle$, $\langle 9, 3\rangle$, $\langle 9, 1\rangle$, ..., cioè le coppie costituite da ciascuno degli interi positivi e da ciascuno dei suoi divisori propri.

(4) Macchina che fa riferimento all'alfabeto $\mathsf{A} = \{\mathtt{a}, \mathtt{b}\}$ ed emette le coppie $\langle \mathtt{aa}, \mathtt{a}\rangle$, $\langle \mathtt{ab}, \mathtt{a}\rangle$, $\langle \mathtt{ba}, \mathtt{b}\rangle$, $\langle \mathtt{bb}, \mathtt{b}\rangle$, $\langle \mathtt{aaa}, \mathtt{a}\rangle$, $\langle \mathtt{aaa}, \mathtt{aa}\rangle$, $\langle \mathtt{aab}, \mathtt{a}\rangle$, $\langle \mathtt{aab}, \mathtt{aa}\rangle$, ...

Le prime tre liste.G precedenti sono relazioni tra interi positivi (con una presenza dello 0), la quarta è una relazione tra stringhe di $\{\mathtt{a}, \mathtt{b}\}^*$.

Le relazioni generate dalle macchine precedenti possono leggersi, risp.: (1) relazione identità tra numeri naturali, (2) successione dei numeri primi, (3) relazione "essere diviso propriamente da", (4) relazione "avere come sprefisso proprio non muto".

B18:**f.02** Riprendendo le considerazioni in B17e05, si può dimostrare facilmente che il dominio e il codominio di una relazione ricorsiva sono insiemi ricorsivi.

Si può inoltre dimostrare ricorrendo ad argomentazioni diagonali alla Cantor che i prodotti cartesiani di insiemi ricorsivi quali $\mathbb{N} \times \mathbb{N}$, $\mathbb{N} \times \mathsf{A}^*$, $\mathbb{Z} \times \mathbb{Z}$, $\mathsf{A}^* \times \mathsf{B}^*$, $\mathsf{A}^{\times 3}$ e altri simili sono insiemi ricorsivi e si possono considerare ambienti ricorsivi nei quali è agevole collocare relazioni-G binarie o di arietà superiore.
Le particolari relazioni introdotte in f01 sono evidentemente sottoinsiemi ricorsivi degli ambienti ricorsivi $\mathbb{N} \times \mathbb{N}$ e $\{\mathtt{a}, \mathtt{b}\}^+ \times \{\mathtt{a}, \mathtt{b}\}+$.

In generale si possono introdurre relazioni binarie ricorsive che sono sottoinsiemi di prodotti cartesiani della forma $\mathsf{U} \times \mathsf{V}$ o in particolare della forma $\mathsf{U} \times \mathsf{U}$ dove con U e V denotiamo ambienti ricorsivi.

Varie relazioni ricorsive si possono ottenere con MSPGIp che scorrono secondo un ordine algoritmico un prodotto cartesiano della forma $\mathsf{U} \times \mathsf{V}$ e a ciascuna delle coppie di questo ambiente applicano una MSPA $\mathbf{S}$ con il compito di selezionare ed emettere solo le coppie che giudica accettabili.

Esempi di relazioni ricorsive di questo genere sono, per ogni $k \in \mathbb{P}$, gli insiemi

$$\{\langle i, j\rangle \in \mathbb{N} \times \mathbb{N} \ \S \ |i - j| < 2\} \ \text{ e } \ \{\langle v, w\rangle \in \mathsf{A}^* \times \mathsf{A}^* \ \S \ v \in \mathbf{Pfx}(w)\} \ .$$

Si possono inoltre introdurre ambienti esprimibili come prodotti cartesiani della forma $\mathsf{U} \times \mathsf{V} \times \mathsf{W}$, dove con U, V e W denotiamo ambienti primari o ambienti ricorsivi e procedendo in questa direzione si possono introdurre ambienti esprimibili come prodotti cartesiani di un numero qualsiasi di insiemi ricorsivi introdotti in precedenza.

Scorrendo questi ambienti evidentemente sequenzializzabili e applicando a ciascuna terna, quaterna o s-upla visitata una MSPA selettrice si possono individuare le cosiddette relazioni ricorsive ternarie, quaternarie o dette della arietà s.

B18:**f.03** Anche delle relazioni ricorsive si possono considerare svariate collezioni caratterizzate da proprietà e relazioni espresse in modo simile a quelle esaminate per le relazioni finite in B14.

Si possono quindi prendere in considerazioni relazioni ricorsive riflessive, antiriflessive, simmetriche, antisimmetriche, transitive, di equivalenza, di ordine, di ordine totale, di ordine reticolare, ...

Alle relazioni ricorsive, evidentemente, si possono applicare composizioni di natura insiemistica come unione, intersezione, complementazione e prodotto cartesiano; queste, in conseguenza di quanto dimostrato in :c, portano ad altre relazioni ricorsive.

Inoltre tra queste relazioni ricorsive si possono stabilire ulteriori relazioni: ad esempio può interessare che una relazione R_1 sia meno estesa di una seconda R_2: in tale caso l'affermazione $\mathsf{R}_1 \subseteq \mathsf{R}_2$ si esprime anche dicendo che la R_1 implica la R_2, oppure scrivendo $\mathsf{R}_1 \Longrightarrow \mathsf{R}_2$.

Si possono applicare anche altre composizioni e trasformazioni che portano ad altre relazioni ricorsive; a tali costruzioni sono dedicati i paragrafi che seguono.

B18:**f.04** **(1) Prop.**: Il prodotto di composizione di due relazioni ricorsive è una relazione ricorsiva.

Dim.: Consideriamo le due relazioni ricorsive $\mathsf{R} \in \mathsf{U} \times \mathsf{V}$ e $\mathsf{S} \in \mathsf{V} \times \mathsf{W}$ (con U, V e W ambienti ricorsivi) e due MSPGIp M ed N tali che $\mathsf{R} =: \mathsf{M}^{\mathcal{G}}$ e $\mathsf{S} =: \mathsf{N}^{\mathcal{G}}$. Si tratta di individuare una MSPGIp $\mathbf{P}$ che genera $\mathsf{R} \circ_{lr} \mathsf{S}$.

Essa primariamente organizza una corsa sulle coppie $\langle u, w\rangle$ di $\mathsf{U} \times \mathsf{W}$ secondo un ordinamento algoritmico $\preceq$ per stabilire per ciascuna delle coppie $\langle x, y\rangle$ tali che $\langle x, y\rangle \preceq \langle u, w\rangle$ se si trova nella relazione $\mathsf{R} \circ \mathsf{S}$ grazie a una coppia di coppie $\big\langle\langle u, v\rangle, \langle v, w\rangle\big\rangle$ entrambe precedenti o uguali alla $\langle u, w\rangle$.

Queste decisioni si possono ottenere in un numero finito di passi, in quanto riguardano l'insieme finito delle coppie di $\mathsf{U} \times \mathsf{U}$ che non sono successive della $\langle u, w\rangle$.

Dopo la fase relativa a una coppia $\langle u, w\rangle$ sul nastro di uscita $\boldsymbol{U}$ della M sono elencate le coppie per le quali le coppie sue nonsuccessive garantiscono l'appartenenza ad $\mathsf{R} \circ_{lr} \mathsf{S}$.

Alla fine della fase relativa alla coppia successiva la lista su $\boldsymbol{U}$ potrebbe essere arricchita non solo della nuova coppia ma anche di coppie precedenti grazie all'azione di intermediaria di tale nuova coppia.
Con questo processo vengono individuate tutte e sole le coppie che costituiscono $\mathsf{R} \circ_{lr} \mathsf{S}$ ∎

(2) Prop.: La chiusura transitiva di una relazione ricorsiva entro un ambiente ricorsivo è una relazione ricorsiva.

Dim.: Si considera una relazione ricorsiva $\mathsf{R} \in \mathsf{U} \times \mathsf{U}$ e si tratta di individuare una MSPGIp $\mathbf{T}$ che genera $\mathsf{R}^{\oplus}$.

Similmente alla macchina individuata per la dimostrazione precedente $\mathbf{T}$ organizza una successione di fasi associate alle coppie $\langle u, w\rangle$ di $\mathsf{U} \times \mathsf{V}$ successive secondo un ordine $\preceq$ e in ciascuna fase stabilisce quali coppie precedenti o uguali alla suddetta si trovano in $\mathsf{R}^{\oplus}$ grazie all'intermediazione delle sole coppie suddette.

Ancora si tratta di fatti che si possono decidere in un numero finito di passi, in quanto riducibili all'esame di insiemi finiti di coppie. Anche con questo processo vengono individuate tutte le coppie di $\mathsf{R}^{\oplus}$ ed esse sole ∎

(3) Prop.: La chiusura di equivalenza di una relazione ricorsiva entro un ambiente ricorsivoè una relazione ricorsiva.

Dim.: Si considera una relazione ricorsiva $\mathsf{R} \in \mathsf{U} \times \mathsf{U}$ e si procede ad individuare una MSPGIp $\mathbf{T}$ che genera R^{eqv}.

Similmente alla macchina individuata per la dimostrazione precedente $\mathbf{T}$ organizza una successione di fasi associate alle successive secondo un ordine sequenziale $\preceq$ coppie $\langle u, w\rangle$ e in ciascuna fase considera l'insieme

$$K_{\langle u,w\rangle} \;:=\; \{\langle x, y\rangle \in \mathsf{U} \times \mathsf{U} \;\big[\!\big]\; \langle x, y\rangle \preceq \langle u, w\rangle\}$$

e stabilisce la partizione di tale insieme corrispondente all'equivalenza dovuta all'intermediazione delle sole coppie dell'insieme stesso.

Ancora si tratta di fatti che si possono decidere in un numero finito di passi in quanto riducibili all'esame di un insieme finito di coppie. Anche con questo processo vengono individuate tutte le coppie di R^{eqv} ed esse sole ∎

B18:**f.05** Le più importanti di queste composizioni e relazioni (ovvero proprietà) ricorsive si possono esprimere in termini che prescindono dalle modalità secondo le quali le composizioni e relazioni specifiche sono state costruite o definite. Esse quindi, come vedremo, possono essere estese a relazioni più generali di quelle costruibili con MSPGIM oppure con procedure; si tratta di relazioni più generali che possono essere definite ed esaminate nell'ambito di una teoria assiomatica degli insiemi.

Come già segnalato per le relazioni finite, sono particolarmente importanti le equivalenze e le relazioni d'ordine. Per individuarle conviene definire alcune collezioni di relazioni associate a proprietà piuttosto semplici che estendono collezioni di relazioni finite.

Consideriamo per questo una relazione-G R avente come dominio D e codominio C

Essa si dice **relazione riflessiva** sse $\forall x \in D \cap C \;:\; x\mathsf{R}x$;

Si dice invece **relazione antiriflessiva** sse $\forall x \in D \cap C \;:\; x \not\mathsf{R}x$;

Si dice **relazione simmetrica** sse $\forall x, y \in D \cap C \;:\; x\mathsf{R}y \Longrightarrow y\mathsf{R}x$;

Si dice **relazione antisimmetrica** sse $\forall x, y \in D \cap C \;:\; x\mathsf{R}y \Longrightarrow y \not\mathsf{R}x$;

Si dice **relazione transitiva** sse $D = C$ e $\forall x, y, z \in D \;:\; x\mathsf{R}y \;\wedge\; y\mathsf{R}z \Longrightarrow x\mathsf{R}z$.

Si dice **relazione di equivalenza** o semplicemente '**equivalenza** sse $D = C$ ed essa è riflessiva, simmetrica e transitiva.

Si dice **relazione d'ordine**, o **poset**, sse $D = C$ ed è riflessiva, antisimmetrica e transitiva.

B18:**f.06** Limitiamoci ora a presentare le costruzioni e le proprietà di maggior rilievo che possono riguardare le relazioni tra interi o tra stringhe; per generalità consideriamo una arbitraria relazione procedurale $\mathsf{R} \subseteq \mathsf{U} \times \mathsf{V}$.

Si dice **dominio** della R l'insieme $\mathsf{dom}(\mathsf{R})$ dei primi membri delle coppie che costituiscono la relazione; si dice **codominio** della R l'insieme $\mathsf{cod}(\mathsf{R})$ dei secondi membri delle coppie che costituiscono la relazione.

Sia il dominio che il codominio di una relazione-G è un insieme-G: infatti si può arricchire la procedura che lo genera con una prestazione che comporta l'emissione su un apposito nastro dei soli primi membri, oppure dei secondi membri, di ogni coppia elemento della relazione.

Si distinguono le relazioni-G funzionali, che chiamiamo anche **funzioni-G** da U in V, le relazioni $\mathsf{R} \subseteq \mathsf{U} \times \mathsf{V}$ tali che se $\langle a, b\rangle, \langle a, b_1\rangle \in \mathsf{R}$, allora deve essere $b = b_1$.

Per ciascuna di tali relazioni-G risulta utile individuare una MSPT che a ogni elemento $a \in \mathsf{dom(R)}$ associa l'elemento $b \in \mathsf{cod(R)}$ tale che $\langle a, b\rangle \in \mathsf{R}$.

Tra le funzioni-G $f \in \lceil\, \mathsf{U} \longrightarrow \mathsf{V} \,\rceil$ si distinguono:

le **funzioni suriettive**, tali che $\mathsf{cod}(f) = \mathsf{V}$,

le **funzioni iniettive**, tali che ogni elemento del codominio può essere l'immagine di un solo elemento del dominio,

e le **funzioni biiettive** o **funzioni invertibili**, le funzioni tali che la relazione loro trasposta è anch'essa una relazoione-G funzionale chiamata funzione inversa della data.

Semplicissima funzione invertibile è l'identità su U, $\mathsf{Id_U}$, definita come l'insieme-GI $\{x \in \mathsf{U} :|\, \langle x, x\rangle\}$.

Le relazioni-G, in quanto insiemi-G di coppie, possono essere composte con operazioni insiemistiche binarie come unione, intersezione, eliminazion, differenza simmetrica e prodotto cartesiano.

Esse inoltre possono essere sottoposte alla complementazione, cioè alla operazione che trasforma ogni relazione-G nella sua negazione.
Si segnala che il complemento di una relazione-G può non essere una relazione-G; più precisamente una relazione-G il cui complemento è una relazione-G deve essere una relazione ricorsiva, cioè una relazione-GB.

Le relazioni-G possono essere composte anche con il cosiddetto prodotto di composizione o prodotto di Peirce.

Una biiezione composta con la propria inversa fornisce l'identità sul suo dominio.

B18:**f.07** Tra le funzioni-G rivestono particolare interesse quelle del genere $\lceil\, \mathbb{N} \longmapsto \mathbb{N} \,\rceil$, ovvero le entità note come **successioni di numeri naturali**.

In particolare le funzioni-GB si possono individuare con le sequenze illimitate dei valori generati da una MSPGIp; a ciascuna di esse possiamo attribuire la forma

$$\mathsf{a} = \langle a_0, a_1, a_2, ..., a_n, ...\rangle$$

Sono utili in particolare le successioni per le quali le componenti generiche a_n possono essere individuate mediante espressioni significative oppure mediante algoritmi.

Funzioni molto vicine alle funzioni del genere $\lceil\, \mathbb{N} \longmapsto \mathbb{N} \,\rceil$ sono quelle dei generi $\lceil\, \mathbb{P} \longmapsto \mathbb{N} \,\rceil$ e $\lceil\, \mathbb{P} \longmapsto \mathbb{P} \,\rceil$: in effetti a ciascuna $\mathsf{a} \in \{\mathbb{N} \longmapsto \mathbb{N} \,\rceil$ si può associare biunivocamente la successione

$$\mathsf{b} \;:=\; \langle b_1, b_2, b_3, ..., b_n, ...\rangle \quad \text{con} \quad \forall n = 1, 2, 3, ... \,:\, b_n := a_{n-1} \;.$$

Si tratta quindi di generi di funzioni sostanzialmente equivalenti e la scelta tra di esse può essere motivata solo da piccoli vantaggi formali o pratici.

Raccogliamo anticipandoli alcuni esempi di successioni di interi.

la successione dei fattoriali definita ponendo $0! := 1$ e chiedendo $n! := n \cdot (n-1)!$ per $n = 1, 2, 3...$;

la successione dei numeri di Fibonacci ottenuta ponendo $F_0 := 1$, $F_1 := 1$ e calcolando i successivi mediamte la $F_n := F_{n-1} + F_{n-2}$ per $n = 2, 3, ...$;

la successione dei numeri di Catalan definita dalle richieste [D20e]

$$C_0 := 1 \;\text{ e }\; C_n := \sum_{i=0}^{n-1} C_i \, C_{n-1-i} \;\text{ per } n = 1, 2, 3, ...$$

la sequenza dei numeri di Bernoulli definita da

$$\mathrm{Brn}_0 := 1 \qquad \text{e} \qquad \mathrm{Brn}_m \;:=\; -\frac{1}{m+1}\sum_{j=0}^{m-1}\binom{m+1}{j}\mathrm{Brn}_j \;.$$

B18:**f.08** Come si è accennato in f0?, si possono anche definire insiemi ambiente ricorsivi della forma $\mathsf{U}_1 \times \mathsf{U}_2 \times ... \times \mathsf{U}_d$ dove $d = 3, 4, ...$, ove gli U_h sono ambienti ricorsivi.

Tra i più semplici di tali ambienti vi sono gli insiemi $\mathbb{N}^d$ costituiti dalle d-uple di interi naturali.

Ai loro elementi si possono associare biunivocamente le classi della equivalenza per permutazioni delle stringhe sopra un alfabeto ordinato di d caratteri che scriviamo $\langle a_1, a_2, ..., a_d\rangle$: più precisamente alla d-upla di interi naturali $\langle h_1, h_2, ..., h_d\rangle$ si associa la stringa $a_1{}^{h_1}\, a_2{}^{h_2}\, , ... , a_d{}^{h_d}$ da considerare rappresentante di tutte le stringhe ad essa equivalentoi per permutazione.

Negli ambienti ricorsivi sopra accennati si collocano le espressioni che consentono di definire formalmente le strutture matematiche di una data specie.

In particolare si possono considerare le espressioni che definiscono le MSP di un data specie, ad esempio le macchine di Turing originali, quelle dotate di un solo nastro.
Possiamo quindi considerare gli insiemi di MSP, di MSPG, di MSPGI, di MSPGip, MSPGIM e di altre specie di macchine come insiemi ricorsivi.
Per questi insiemi usiamo notazioni come MSP, MSPG, MSPGI e MSPGF.

B18:g. considerazioni critiche sugli insiemi procedurali

B18:**g.00** Conviene ricordare che finora abbiamo spesso proceduto con atteggiamento fiduciario: non abbiamo definito precisamente MSP e quindi non si ossono affrontare rigorosamente problemi come quello di decidere se una MSPG sia una MSPGI o meno.

Ci limitiamo a sostenere sulla fiducia che queste precisazioni sono fattibili e rinviamo la loro formulazione a causa della sua pesantezza espositiva ritenendo opportuno non ritardare l'introduzione di nozioni che possono presentare maggiore interesse, in particolare per chi è interssato principalmente alle applicazioni.

B18:**g.01** Ci proponiamo ora un primo chiarimento di una problematica che al livello degli orientamenti generali riveste grande importanza, quella concernente le possibilità di tenere sotto controllo le evoluzioni delle MSPG.

Innanzi tutto occorre delineare i tipi di situazioni che si può trovare davanti un esecutore di procedure, umano o automatico, incaricato di fornire informazioni sulla evoluzione di una MSPG M.

Si cerca di classificare le situazioni che si possono riscontrare dopo che la macchina ha eseguito un certo numero n di passi, e quindi dopo che è stata impiegata di una certa quantità q_n di risorse (tempo, memorie ed eventuali attività di adattamento delle procedure).

(1) Si è dimostrato che M genera una lista limitata di stringhe, cioè che $\mathsf{M} \in \mathsf{MSPGF}$. In tal caso, dopo n passi possono essere individuate due situazioni.
   (1a) La macchina si arresta oppure si dimostra di aver ottenuta l'individuazione esplicita dell'intera lista $\mathsf{M}^{\mathcal{G}}$ e che è inutile far proseguire l'evoluzione: quindi si ha la conclusione dell'evoluzione.
   (1b) Con le risorse impiegate si è ottenuta solo una parte E dell'insieme $\mathsf{M}^{\mathcal{G}}$ e si ha la consapevolezza della incompletezza dei risultati, cioè si sa che l'impiego di altre risorse consentirà la generazione di altre stringhe, forse di tutte le altre.

Si pone quindi la scelta (1bα) aut proseguire nella speranza di ottenere altre stringhe di $\mathsf{M}^{\mathcal{G}} \setminus E$ a un costo contenuto (1bβ) aut interrompere l'evoluzione per evitare di impiegare altre costose risorse per ottenere poco o nulla di nuovo.

   (1c) Con le risorse impiegate si è ottenuto sul nastro di uscita una lista L e non si sa se impiegando altre risorse possano essere generate altre stringhe, cioè non si sa se $L = \mathsf{M}^{\mathcal{G}}$ oppure $L \subset \mathsf{M}^{\mathcal{G}}$.

(2) Si è dimostrato che M genera un elenco illimitato di stringhe, cioè che $\mathsf{M} \in \mathsf{MSPGI}$, ovvero che M individua un insieme procedurale infinito. In questo caso si possono prospettare due sottocasi.
   (2a) Si è trovato che la M genera una lista progressiva (in particolare che procede a implementare una formula dipendente da un parametro estensivo n per valori crescenti con n) e quindi che è in grado di procedere illimitatamente e progressivamente alla generazione di $\mathsf{M}^{\mathcal{L}}$; conseguentemente si sa che per una qualsiasi stringa $w \in \mathsf{A}^*$ impiegando altre risorse si potrà decidere se essa appartiene o meno a $\mathsf{M}^{\mathcal{G}}$.
   (2b) Si è consapevoli che le risorse impiegate hanno consentito solo di individuare una lista parziale delle stringhe di $\mathsf{M}^{\mathcal{G}}$ ma non si ha la garanzia che la generazione sia progressiva e non si sa se per una qualsiasi stringa $w \in \mathsf{A}^*$ impiegando altre risorse si potrà decidere se essa appartiene o meno a $\mathsf{M}^{\mathcal{G}}$.
   (2c) Si è individuata una MSPG progressiva N in grado di procedere illimitatamente alla generazione della sequenza $\mathsf{N}^{\mathcal{G}}$ costituita da stringhe appartenenti a $\mathsf{M}^{\mathcal{G}}$, ma che non lo esauriscono e si è individuato un elenco esplicito parziale di stringhe di $\mathsf{N}^{\mathcal{G}}$ non facenti parte di $\mathbf{Q}^{\mathcal{L}}$.

(3) Non si è riusciti a decidere se la sequenza che la MSPG M va generando può crescere illimitatamente o meno; con le risorse impiegate fino a un passo m si dispone di una lista di stringhe generate L che potrebbe esaurire $\mathsf{M}^{\mathcal{G}}$ o all'opposto rivelarsi parziale, in quanto impiegando nuove risorse diventa possibile generare altre stringhe che appartengono ad $\mathsf{M}^{\mathcal{G}} \setminus L$.

Si tratta allora di prendere una decisione operativa piuttosto rischiosa. Se si decide di proseguire impiegando una data quantità di altre risorse potrebbe accadere sia di emettere nuove stringhe di $\mathsf{M}^{\mathcal{G}} \setminus L$, sia di sprecare risorse per ottenere poco o nulla. In alternativa si può scegliere di studiare più a fondo la M sperando di ottenere nuove informazioni sopra i suoi comportamenti, oppure di studiare il problema che ha condotto alla M, sperando di individuare una macchina migliore per la sua soluzione.

Come ripiego rispetto a quest'ultima speranza si può cercare di ridurre la portata del problema per individuare procedure risolutive più controllabili.

B18:**g.02** Come abbiamo già osservato, per trattare con efficienza espositiva dati da elaborare o prodotti di elaborazioni, si devono individuare tutti gli insiemi di tali oggetti che rivestono interesse e occorre contraddistinguere ciascuno di essi con notazioni chiaramente definite e bene identificabili.

Per questi identificatori si possono utilizzare convenientemente i segni di operazioni su insiemi come intersezione, unione, eliminazione, differenza simmetrica, complementazione e prodotto cartesiano, le relazioni di sottoinsieme in senso lato e in senso stretto, la costruzione dell'insieme delle parti, le particolarizzazioni riguardanti funzioni, relazioni d'ordine, relazioni di equivalenza, partizioni, strutture algebriche e tante altre.

Vedremo più avanti altri procedimenti che consentono di costruire insiemi che risulta naturale o utile presentare in due dimensioni: innanzi tutto prodotti cartesiani di insiemi numerabili che risultano essi stessi numerabili; poi costruzioni innovative come quelle che hanno condotto ai numeri negativi, ai razionali e ai coefficienti binomiali [B13e13].

Sarà interessante anche servirsi di arborescenze illimitatamente costruibili e di digrafi graduati illimitatamente costruibili (che spesso sbrigativamente vengono chiamati anche arborescenze infinite e digrafi graduati infiniti) mediante i quali si possono tenere sotto controllo insiemi di strutture di grande utilità, ad esempio la totalità delle arborescenze distese [D30] e con queste intere collezioni di formule ben formate.

A proposito di queste strutture grafiche segnaliamo che etichettando secondo opportuni criteri i loro nodi terminali con operatori e operandi [C14], si giunge alla possibilità di generare algoritmicamente le totalità delle definizioni di una data forma di molte specie di strutture (algebriche, combinatoriche, ...) che introdurremo più avanti [e.g. C14e, C14f e C14g].

B18:**g.03** Di ogni linguaggio N emesso da una MSPGI M che genera un linguaggio ricorsivo sul suo alfabeto di uscita che denotiamo con A, si può dire che, in linea di principio può essere tenuto sotto controllo.

Questo controllo deve essere in grado di fare ricorso a algoritmi di appartenenza, di confronto o di definizione di stringa successiva che in generale risultano solo parzialmente definiti e che possono essere utilizzati con efficacia o addirittura con efficienza solo limitandosi a situazioni piuttosto specifiche.

Più precisamente, data una stringa qualsiasi $w \in \mathsf{A}^*$ si può dire se compare o meno sul nastro di uscita ed in caso affermativo si può attendere con piena fiducia che la corrispondente macchina che decide l'appartenenza al linguaggio $\mathsf{M}^{\mathcal{G}}$ emetta la w.

Questa attesa in certe circostanze può essere resa abbastanza consapevole, in quanto si può valutare la prossimità della sua comparsa mediante qualche tipo di confronto con la w di ogni nuova stringa emessa.

Dalla posizione sul nastro di emissione delle stringhe emesse da una MSPG che genera un linguaggio ricorsivo si può ricavare un cosiddetto ordinamento sequenziale delle stringhe emesse, cioè di un ordinamento tale che: (1) è individuabile la prima stringa generata, (2) data una stringa qualsiasi sull'alfabeto in uso, è possibile individuare in un numero finito di passi la sua (immediata) successiva. Questo tipo di relazione tra stringhe è evidentemente una relazione d'ordine totale.

Conviene osservare esplicitamente che si individuano duetti di MSPG diverse che generano le stesse stringhe ma che definiscono ordinamenti sequenziali diversi: questo ad esempio accade quando si generano nell'ordinamento len-lxg le stringhe sulle tre lettere `a`, `b` e `c` riferendosi a due diversi loro ordinamenti, ad esempio ad (`a` $\prec$ `b` $\prec$ `c`) ed a (`b` $\prec$ `c` $\prec$ `a`).

(1) Prop.: Per un insieme di stringhe ordinate sequenzialmente da un ordinamento che denotiamo con $\preceq$ esiste un algoritmo che, date due diverse stringhe generate v e w, consente di decidere (in un numero finito di passi) se $v \prec w$ o viceversa se $w \prec v$.

Algoritmo: Si costruiscono due liste, la prima che inizia con v, la seconda con w; si procede quindi a successive fasi nelle quali si accoda una stringa successiva sia alla prima che alla seconda lista. Questo processo deve aver fine, aut perché nella prima lista si ottiene w e in tal caso $v \prec w$, aut perché nella seconda lista si ottiene v e in questo caso $w \prec v$ ∎

Nel seguito denoteremo con MSPGIp l'insieme delle MSP che generano stringhe illimitatamente e progressivamente, cioè in modo tale che si conosca un algoritmo di confronto in grado di decidere quale di due stringhe generate preceda l'altra.

`Testi dell'`*esposizione* `in http://www.mi.imati.cnr.it/alberto/ e in http://arm.mi.imati.cnr.it/Matexp/`