Capitolo B18:
Insiemi generati da procedure

Contenuti delle sezioni

B18:0.01 All'inizio di questo capitolo motiviamo, mediante alcuni esempi ed alcune considerazioni sulla portata delle procedure, l'opportunità di introdurre la nozione di infinito nella sua prima accezione di infinito potenziale. Contemporaneamente si introduce la nozione di insieme procedurale ambientandola tra le sequenze illimitate di stringhe e di numeri interi. Come nel precedente capitolo ancora si rinvia una precisa definizione delle macchine cui si fa riferimento e si tralasciano molti dettagli.

Nell'ultima parte si sviluppano considerazioni generali sopra insiemi e proprietà e si introduce l'idea dell'impostazione assiomativa della teoria degli insiemi, ancora senza pretesa di completezza, ma solo facendo ricorso ad esempi comprensibili intutivamente.

In effetti si è ritenuto opportuno che, prima di trattare in modo più completo le questioni generali sulle costruzioni formali, venissero introdotte varie nozioni ampiamente utilizzate, in quanto la loro conoscenza consente di esemplificare in modo più significativo e di apprezzare meglio la strumentazione che si va introducendo. In questo capitolo, comunque, si cominciano ad individuare 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. Requisiti del calcolo scientifico-tecnologico

B18:a.01 Qui riprendiamo ad affrontare alcune questioni generali concernenti il **calcolo scientifico e tecnologico**.

Innanzi tutto una mera convenzione: nel seguito abbrevieremo la specificazione "scientifico e tecnologico" e le sue varianti con la specificazione sincopata **-ST**.

Una finalità generale che caratterizza il calcolo -ST consiste nella definizione di **procedure computazionali di portata tendenzialmente ampia**, ovvero in grado di affrontare una gamma di problemi tendenzialmente estesa e varia. Con le procedure che non posseggono ampia portata, ma cercano risposte ad esigenze specifiche, talvolta episodiche, non si fa matematica, informatica, scienza o tecnologia e non si ottengono enunciazioni -ST.

Se ci si limitasse ad affrontare problemi specifici sarebbe sufficiente accumulare conoscenze empiriche sui singoli problemi attraverso osservazioni il più possibile accurate, formularle e registrarle con precisione

ed organizzare i dati registrati seguendo criteri che ne rendano agevole il recupero per particolari circostanze simili a quelle che caratterizzano le osservazioni.

La sottolineatura della distinzione fra finalità generale del calcolo -ST e obiettivi di calcolo specifici non intende sminuire le attività mosse da esigenze particolari, spesso impegnative, meritorie e capaci di aprire nuovi campi di indagine e di azione; essa vuole solo evidenziare che le attività -ST si pongono come fine primario quello di dare indicazioni conoscitive e operative che tendono ai più elevati potenziali di utilizzabilità.

B18:a.02 Dall'esame dell'obiettivo delle procedure e delle elaborazioni di ampia portata che caratterizzano le attività -ST emerge un complesso di esigenze concernenti la **riproducibilità**, la **condivisibilità**, la **generalità**, la **versatilità** e la **precisione**. Conviene osservare che riproducibilità e condivisibilità sono essenziali per conseguire la generalità, mentre versatilità e precisione sono importanti per i successi applicativi.

Lo sviluppo del calcolo -ST ha riguardato e riguarda una varietà di fasi, di temi e di interessi. In effetti questo sviluppo, soprattutto nei tempi recenti, si colloca al di sopra delle possibilità dei singoli, ma richiede il contributo di un gran numero di ricercatori, di operatori e di sistemi automatici; tutti questi a loro volta devono poter disporre di strumenti conoscitivi che si concretizzano in sistemi di documenti che devono essere utilizzabili in svariati contesti e in molteplici circostanze.

Si pongono quindi numerosi e delicati problemi concernenti le strategie di strutturazione degli strumenti conoscitivi, di definizione di termini dotati di significati condivisi (standards terminologici e notazionali, convenzioni su simboli e formule, ...), di precisazione di linguaggi, di sistemi software e di metodiche per l'organizzazione e l'uso degli archivi di documenti.

Inoltre, come abbiamo già osservato (B10:a.07, :a.10, B10:d.02, ...) e come dovremo sottolineare in vari punti, le attività di gestione delle conoscenze, accanto alle esigenze di precisione e rigore, si trovano davanti alla necessità di operare semplificazioni ed abbreviazioni.

Lo studio dei metodi e dei procedimenti di calcolo -ST e della loro implementazione mediante strumenti effettivi (essenzialmente mediante software) comporta un'ampia varietà di precisazioni e distinzioni e queste possono appesantire enormemente la descrizione degli strumenti. Queste difficoltà portano a pensare che per sviluppare procedure di ampia portata possa essere opportuno lasciare temporaneamente imprecisati molti dettagli concernenti la concreta realizzabilità, condizioni che devono invece essere attentamente eaminate e rispettate nelle realizzazioni specifiche impegnative.

Inoltre si constata che in molte fasi dello sviluppo del calcolo -ST alcune delle esigenze invocate all'inizio del paragrafo (riproducibilità, condivisibilità, generalità, precisione e versatilità,) entrano in conflitto tanto da spingere a procedere anche adottando dei compromessi.

In particolare si hanno spesso conflitti fra la tendenza alla generalità e la tendenza alla precisione: cercando di generalizzare dei risultati si devono introdurre distinzioni che possono appesantire parecchio le descrizioni delle procedure.

Molti di questi conflitti si contrastano operando delle semplificazioni; in particolare si impone l'opportunità di redigere descrizioni che rinunciano ad entrare in gran parte dei dettagli. Ad esempio può rendersi opportuno prescindere dalle precisazioni su dominio e codominio di funzioni numeriche ottenibili con algoritmi, come vedremo in :a.08 e :a.09.

B18:a.04 Un buon esame delle caratteristiche fondamentali e del ruolo delle procedure e delle elaborazioni in relazione con le loro applicazioni richiede anche di porsi in una prospettiva storica. Occorre tenere conto del progressivo accumulo delle esperienze del passato e della continua evoluzione del settore in termini di risultati, di strumenti disponibili e di aspettative.

Sia all'esigenza dell'ampia portata che all'esperienza storica conviene collegare tre richieste programmatiche per l'avanzare dello studio delle procedure -ST, ciascuna riguardante uno dei tre generi delle risorse che esse impiegano.

(1) Può rendersi necessario operare con stringhe progressivamente più lunghe: questo può tradursi nella richiesta di macchine che dispongano di nastri o di analoghi dispositivi estesi quanto serve.

(2) Può rendersi necessario far proseguire le elaborazioni per durate progressivamente più lunghe (per costruire stringhe via via più lunghe, ovviamente, sono richieste durate che crescono almeno proporzionalmente alla lunghezza): si chiede quindi che sia possibile far proseguire le elaborazioni a lungo quanto serve.

(3) Può rendersi necessario servirsi di manovre progressivamente più elaborate: quindi si chiede che sia possibile servirsi di dispositivi e/o di programmi articolati quanto serve, cioè che si avvalgano di combinazioni di scelte e di manovre elaborate quanto serve.

B18:a.05 Negli ultimi decenni a molte di queste richieste si è potuto andare incontro grazie alla crescita delle prestazioni delle apparecchiature informatiche, apparecchiature dotate di memorie sempre più estese per dati e programmi e dotate di sempre maggiori velocità di elaborazione.

Attualmente ogni giorno sono eseguiti trilioni di elaborazioni molte delle quali richiedono molti miliardi di passi. Queste prestazioni possono essere otenute anche grazie alla attuale possibilità di far cooperare in diversi modi molti milioni di computers interconnessi con canali di telecomunicazione di grande capacità e versatilità.

Le prospettive del perdurare dei progressi tecnologici consentono di prospettare per i prossimi anni ulteriori rilevanti crescite delle prestazioni computazionali e delle ambizioni dei progetti di sviluppo -ST.

Se i termini usati in :a.04, "stringhe lunghe quanto si vuole", "tempi lunghi quanto serve" e "manovre elaborate quanto serve", sono giudicati troppo vaghi, possiamo sostituirli con termini come "lunghezza sensibilmente maggiore delle lunghezze ottenute in precedenza per affrontare i problemi di quel settore", "tempi più lunghi delle durate delle precedenti elaborazioni con obiettivi simili", "manovre più articolate di quelle dispiegate in precedenza".

In effetti questa tendenza a spostare sempre più in alto le risorse da utilizzare nei calcoli (-ST o per fini particolari) costituisce una caratteristica dell'attuale stadio delle conoscenze umane e delle realizzazioni loro collegate. Questa tendenza ha consentito di ottenere, a partire dalla metà del XX secolo e in sempre più discipline, risultati progressivamente superiori ai precedenti; essa dovrebbe essere considerata come una forte spinta all'allargamento delle prospettive dei metodi e degli strumenti per i calcoli -ST e dell'intero complesso delle indagini -ST.

B18:a.06 Il precedente cenno sulla possibilità di connettere le apparecchiature induce ad esplicitare la considerazione che segue, anche se qualcuno la può giudicare un'ovvietà. Ad ogni passo delle elaborazioni riproducibili si utilizzano risorse finite e risulta cruciale che i risultati di maggiore importanza consentiti dalle procedure possano essere utilizzati agevolmente per altre elaborazioni. Ne conseguono due esigenze: si devono cercare soluzioni che portano a buoni risultati attraverso un numero finito ragionevolmente prevedibile di passi e le soluzioni ottenute devono essere documentate mediante modalità ampiamente condivisibili: servono quindi codifiche, linguaggi descrittivi, linguaggi per la definizione delle procedure e modalità di registrazione che seguono standards meditati e ampiamente adottati.

B18:b. Elaborazioni con risorse illimitate e infinito potenziale

B18:b.01 Diciamo **macchina sequenziale programmabile trasformatrice**, in sigla MSPT, una MSP che a partire da una stringa di ingresso ha il compito di produrre una seconda stringa; questa stringa in uscita può chiamarsi **stringa trasformata** di quella di ingresso.
Una tale macchina si dice **MSPT algoritmica**, in sigla MSTPA, sse per ogni stringa di ingresso attraverso una sequenza finita di passi è in grado di fornire la corrispondente trasformata.

Prendiamo in considerazione in particolare una MSPTA che a partire da un numero naturale rappresentato dalla sua notazione decimale ricava un secondo numero naturale anch'esso espresso nella sua scrittura decimale; per dare un tono pratico allo schema prospettato si può supporre che un utente consideri il numero in uscita dipendente da quello di ingresso secondo una regola determinata.

Ad esempio consideriamo la procedura 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 $SB(n)$: i valori $SB(n)$ 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 algoritmica finita.
Ad esempio 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 $\mathbf{[}0 : 2^k - 1\mathbf{]}$, si ha come codominio $\mathbf{[}0 : k\mathbf{]}$, mentre ad un dominio $\mathbf{[}0 : N\mathbf{)}$ con $N \in \mathbf{[}2^k : 2^{k+1}\mathbf{)}$ corrisponde il codominio $\mathbf{[}0 : k+1\mathbf{]}$.

B18:b.02 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 (v. B26:d.03) 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 operano su più interi 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, ovviamente, il codominio è l'intervallo $\mathbf{[}0 : 2^{31} - 1\mathbf{]}$, mentre il dominio è l'insieme sensibilmente meno banale delle coppie di naturali $\langle m, n\rangle$ che soddisfano la disuguaglianza $m + n < 2^{31}$.

Altre MSPA comportano limitazioni molto più impegnative da studiare in modo preciso: si pensi ad esempio alla valutazione di un'espressione un poco elaborata contenente più addizioni e moltiplicazioni sopra una dozzina di operandi da cercarsi tra gli interi naturali.

In effetti occorre segnalare che in molti campi applicativi vengono posti problemi, sia numerici che non numerici, che possiamo qualificare come **parzialmente indeterminati** per i quali è relativamente definito il genere di prestazione richiesto e quindi per i quali si impongono naturalmente l'insieme ambiente entro il quale vanno trovati i dati e l'insieme ambiente nel quale cercare i risultati; non sono invece precisati a priori il dominio ed il codominio delle funzioni che corrispondono ai possibili procedimenti risolutivi.

Per ogni algoritmo risolutivo di uno di questi problemi è naturale che si cerchi di determinare precisamente una tale coppia ⟨ dominio , codominio ⟩ in quanto in linea di principio solo essa permette di definire completamente l'algoritmo stesso e conseguentemente permette di chiarire la concreta realizzabilità della corrispondente MSPA.
Ora accade che per molti problemi parzialmente indeterminati la soluzione della questione della precisazione degli algoritmi risulta tutt'altro che banale e anche molto impegnativa.

La definizione precisa degli algoritmi risolutivi è sicuramente cruciale in alcuni casi specifici, ad esempio per le elaborazioni numeriche che devono essere effettuate da computers incorporati in apparecchiature costose e utilizzate per scopi delicati e in qualche modo determinanti.
In questi casi per la precisazione di dominio e codominio potrebbe essere opportuno procedere euristicamente ricorrendo ad un secondo algoritmo "supervisore" che proceda ad un esame sperimentale dell'intero ambiente del dominio o di una sua parte critica.
Un tale supervisore spesso risulterebbe dispendioso e per talune applicazioni pratiche potrebbe essere preferibile procedere probabilisticamente attraverso tests di opportuni campioni, cioè attraverso indagini non del tutto sicure, ma molto meno costose delle sperimentazioni esaurienti.
In effetti per la precisazione di dominio e codominio di sottoprogrammi di interesse pratico, strumenti che devono considerarsi come importanti prodotti industriali, ci si basa su prove di collaudo da definire accuratamente, sia sul piano tecnico che sul piano legale. Anche qui emerge la diversità tra le esigenze delle applicazioni specifiche e le esigenze degli studi con finalità generali.

Bisogna tuttavia riconoscere che delle funzioni associate ad algoritmi spesso non risulta determinante. Vi sono molte situazioni nelle quali si lascia correre.

B18:b.03 In sintonia con la illimitatezza degli oggetti degli enunciati presentata in precedenza si può considerare lecito assumere la "illimitata estendibilità" degli ambienti nei quali collocare il dominio e il codominio di molte funzioni.
L'opportunità di una tale assunzione è sostenuta, innanzi tutto, dalla osservazione che le definizioni di moltissime operazioni vengono semplificate in misura notevole.
Come primo esempio concreto consideriamo il calcolo della somma di interi naturali. Ricordiamo che su tutti i computers sono disponibili dispositivi che effettuano la somma di due interi a ciascuno dei quali sono dedicat registri di memoria di estensione fissa, ad esempio di 16, o di 32 o di 64 bits. Alternativamente si hanno sistemi che permettono di eseguire la somma di du interi di estensione quasi illimitata: se vengono loro proposti addendi interi forniri da centinaia o migliaia di cifre riescono a dedicare a questi numeri interi grandi opportune sequenze di memorie standard e ad effettuare su tali memorie la somma richiesta. In realtà si incontrano dei limiti pratici che non consentono di affermare la possibilità di sommare due interi di estensioni del tutto arbitrarie. Questa situazione ideale si può tuttavia avvicinare e si può pensare che una richiesta relativa a interi superiori di quelli trattati fino ad un certo momento possa essere soddisfatta con nuovi potenziamenti delle apparecchiature.
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.
Risulta tuttavia conveniente postulare questa possibilità fino a prova contraria in quanto una tale convenzione apre la possibilità di esposizioni di sviluppi di strumenti numerici e di formulazioni molto più concise e semplici da comunicare e da condividere.
Secondo questo approccio diventa possibile affermare che la somma può effettuarsi su qualsiasi coppia di interi, o in parole povere dire che la somma di interi risulta "sempre possibile".

B18:b.04 Andando oltre alla semplice operazione del sommare due interi, per una grande varietà di operazioni numeriche si può accettare l'idea che sia possibile avvalersi di macchine più versatili e più adattive che consentano modalità di utilizzo umano più semplici e contemporaneamente discorsi di presentazione più compatti e scorrevoli; naturalmente la costruzione di queste macchine è più impegnativa e la loro descrizione più complessa.

Questo approccio della illimitatezza potenziale può estendersi anche ad operazioni e a proprietà diverse da quelle sopra i numeri interi. In particolare si può proporre per le elaborazioni sulle stringhe molto estese e quindi su tutti gli oggetti matematici esprimibili con precisione mediante stringhe. Ad esempio si possono considerare le operazioni sopra numeri razionali illimitatamente precisi (e quindi alle elaborazioni di numeri reali ovvero sopra le operazioni di calcolo approssimato). E ancora si possono approcciare con la presunzione della illimitatezza le elaborazioni sopra testi ed archivi amministrativi, finanziari, storici, ...

In generale possiamo affermare che si possono ottenere semplificazioni di utilizzo e di inquadramento conoscitivo con macchine che siano capaci di adattarsi alle estensioni delle risorse disponibili. Ribadiamo che sul piano espositivo la possibilità di ricorrere a termini come "risorsa illimitatamente estendibile" consente di semplificare molti discorsi.

Anche queste considerazioni dicono che le esigenze di generalità in linea di massima si trovano a confliggere con le esigenze delle precisazioni necessarie per garantire l'attendibilità degli strumenti concreti.

Qui vogliamo segnalare primariamente che tutti i contrasti segnalati vanno visti in relazione alle esposizioni ed alla documentazione delle procedure -ST e delle conoscenze che le giustificano.

B18:b.05 Nel procedere di questa esposizione ci proponiamo di mostrare che l'esperienza plurisecolare e recente circa gli sviluppi delle elaborazioni e delle procedure -ST conduce in modo ineluttabile ad adottare formalizzazioni e astrazioni; le entità, il linguaggio e le finalità della tradizione matematica sono riconducibili ad esigenze sostanziali ed i risultati che si sono raggiunti e che continuano ad essere conseguiti danno ragione ai metodi ed allo stile elaborato da questa disciplina.

Questa propensione alla formalizzazione ed alla astrazione richiederebbe molteplici approfondimenti che in parte saranno presentati in B60:, B61: eB65: . Questa propensione comunque emergerà progressivamente.

Un primo atteggiamento che si deve segnalare riguarda l'opportunità di ricorrere ampiamente alla nozione di **infinito** , a partire dalla sua accezione di **infinito potenziale discreto**.

Come si è detto, quando si parla di calcolo -ST preoccupandosi anche della realizzabilità si dovrebbero descrivere procedure che sono in grado di generare sequenze lunghe quanto si vuole, su nastri estesi quanto serve, procedendo attraverso un numero di passi elevato quanto è necessario. Si può tuttavia constatare facilmente che se si continuasse ad usare solo queste espressioni si otterrebbero esposizioni lunghe, pesanti alla lettura e poco incisive.

Un'alternativa di una certa convenienza 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 in misura rilevante.

B18:b.06 Si tratta, in buona sostanza, di indivuduare un linguaggio che possa avvalersi di un sistema di notazioni simile a quello con il quale sono stati trattati gli insiemi finiti introdotto a partire da B11: . Questo linguaggio viene chiamato **linguaggio degli insiemi**; esso e la conseguente nozione di insieme, possono essere introdotte su un piano intuitivo per essere utilizzate in una ampia varietà di

situazioni. Si tratta di un linguaggio diretto ampiamente adottato dai matematici e da tutti quelli che della matematica si servono strategicamente e che si è rivelato molto vantaggioso per molte attività conoscitive ed elaborative.

Nelle sezioni che seguono il linguaggio degli insiemi e le relative notazioni saranno giustificati in termini costruttivi per gli insiemi che chiamiamo generati da procedure (:b, :e).
Successivamente in B19: si allargherà il discorso mediante considerazioni che rimangono su di un piano intuitivo per poterci avvalere di entità che chiameremo **insiemi euristici** ed ai quali si propone di attribuire portata generale, fino a qualificarli "insiemi qualsiasi". Questi discorsi costituiscono la cosiddetta **teoria naïve degli insiemi**
A livello colloquiale per gli insiemi euristici useremo espressioni che si riconducono alla metafora degli insiemi come contenitori di propri elementi (scatole, sacchi, armadi, depositi, biblioteche, archivi, ...) .
Il linguaggio intuitivo (naïve, ingenuo) degli insiemi euristici per molte questioni risulta pratico ed efficace e non sembra porre problemi; tuttavia approfondendo talune questioni si trova che esso porta ad imprecisioni ed a conclusioni paradossali (v. [[en.Category:Paradoxes of naive set theory]]).
Una giustificazione più robusta e rigorosa degli insiemi è fornita da una loro specifica teoria formalizzata che si basa sopra un ben definito sistema di assiomi; essa si sviluppa nell'ambito della logica matematica e risulta non poco impegnativa.
La teoria formalizzata nell'attuale capitolo sarà solo accennata nella sezione :g e sarà ripresa in B65: .

B18:b.07 Nelle prossime sezioni tratteremo le cosiddette macchine MSP e le procedure associate servendoci di un linguaggio idealizzato al quale si permette di parlare di entità prodotte su "nastri infiniti" da procedure che hanno operato attraverso una "successione infinita" di passi e quindi servendosi di "risorse infinite". In tal modo si evitano la mole di precisazioni che sono richieste nelle situazioni concrete specifiche e si ottengono vantaggi assai rilevanti sia nell'impostazione dei problemi, sia nella descrizione dei metodi, sia nell'organizzazione dei risultati.

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

I primi insiemi non finiti che conviene introdurre con il linguaggio idealizzato delle MSP che possono rendere disponibili tante stringhe quante si possono richiedere (modello che assume evitabili le possibili carenze di risorse e gli eventi che possano ostacolare il loro procedere) sono i cosiddetti **insiemi numerabili basilari**.
Il più semplice di questi è l'insieme delle rappresentazioni unadiche dei naturali. Questa entità la introduciamo mediante una MSP che può procedere quanto serve nella generazione delle sequenze di un simbolo di base, ad esempio della tacca "|".
Questa è la prima delle varianti delle MSP che a partire da una sequenza finita di dati conduce a una sequenza di risultati che si può far crescere quando si vuole; queste macchine vengono le diciamo **macchine sequenziali programmabili che generano illimitatamente**, termine che abbreviamo con la sigla MSPGI.
È evidente che nessuna MSPA può essere una MSPGI.

Nel seguito useremo la sigla MSPG per abbreviare il termine "MSP finalizzate ad emettere stringhe sopra un nastro di uscita". Una esecuzione di una MSPG potrebbe concludersi dopo un numero finito di passi oppure procedere illimitatamente: nel primo caso la qualifichiamo MSPGF, nel secondo MSPGI. Va segnalato che precisare se una data MSPG sia una MSPGF oppure una MSPGI potrebbe costituire un problema di non facile soluzione e in taluni casi ci si può trovare incapaci di risolverlo e quindi indotti a collocarlo tra i problemi attualmente irrisolti.

B18:b.08 Descriviamo ora una procedura che permette di generare le rappresentazioni unadiche degli interi naturali. È semplice immaginare che una tale procedura riesca ad 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 una stringa come la seguente:

$$\dashv \, , \mid , \mid\mid , \mid\mid\mid , \mid\mid\mid\mid , \mid\mid\mid\mid\mid \vdash$$

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

$$\dashv \, , \mid , \mid\mid , \mid\mid\mid , \mid\mid\mid\mid , \mid\mid\mid\mid\mid , \mid\mid\mid\mid\mid\mid \vdash$$

cioè una rappresentazione di $n+1 = 7$ numeri naturali. Una fase operativa di accodamento al nastro di uscita di una ulteriore sequenza di "|" può essere eseguita quante volte si vuole grazie al postulato di illimitatezza che per l'occasione garantisce ladisponibilità di sufficiente tempo di calcolo e di sufficiente spazio sul nastro.

B18:b.09 Descriviamo ora una seconda MSPGI che consente di procedere nella generazione di "tutte" le stringhe sopra un dato alfabeto finito.
A questo proposito segnaliamo che l'insieme delle stringhe sopra l'alfabeto $\mathcal{A}$ viene chiamato anche **monoide libero** su $\mathcal{A}$ e viene denotato con $\mathcal{A}^*$.

Dato che questa macchina si chiede una portata maggiore di quella descritta in :b.08, ci si aspetta che si tratti di una apparecchiatura con prestazioni più diversificate ed 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 resa effettiva registrando i suoi caratteri sopra un apposito nastro predisposto per la consultazione; esprimiamo l'ordinamento scelto scrivendo $\{\mathtt{a} < \mathtt{b} < \mathtt{c}\}$.

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

Conviene considerare che le stringhe vengono generate per lotti ciascuno dei quali costituito dalle stringhe caratterizzate da una delle successive lunghezze; vogliamo inoltre che le stringhe di una data lunghezza 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):

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

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:

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

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

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

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

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

B18:b.10 Consideriamo la sequenza illimitata costituita dalle scritture binarie dei successivi interi naturali:

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

Si osserva che questa sequenza si può ottenere dalla procedura che genera nell'ordine len-lxg 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 la sequenza precedente segue l'ordine len-lxg.
Più in generale si osserva che ogni sequenza di stringhe ottenuta con una riduzione da una sequenza di stringhe che osserva un ordine len-lxg (o qualsiasi altro ordine totale verificabile algoritmicamente, cioè basato sopra una opportuna routine di confronto) mantiene questa proprietà.

(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.11 Si possono individuare molte altre sequenze illimitate arricchendo con opportuni meccanismi di selezione le MSPGI che procedono alla generazione di tutte le stringhe sopra un alfabeto. In particolare si hanno le notazioni in una qualche base degli interi naturali e applicando opportuni meccanismi selettivi a queste si possono ottenere varie sequenze di interi che sosddisfano proprietà specifiche che potrebbero servire a qualche applicazione: numeri pari, numeri divisibili per $d = 3, 4, ...$, palindromi, ...

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 ad un intero compreso tra 1 e 9. Se tale intero coincide con 3, 6 o 9 è garantita la visibilità, se vale 1, 2, 4, 5, 7 o 8 è garantita la non divisibilità 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 una 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'ordine len-lxg le stringhe su `a` e `b` che lette da sinistra a destra non presentino in alcuna posizione un numero di `a` lette superiore al numero di `b` lette.

Più esplicitamente si chiede di generare la successione delle stringhe che inizia con:

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

B18:b.12 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 altre MSPGI: 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 positivo e di emettere ogni nuovo intero trovato.
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 che permette di semplificare la presentazione di molte di tali sequenze. Accade però che per ciascuna macchina che segue il suddetto schema organizzativo si possono individuare diverse altre MSPG che generano la stessa sequenza di stringhe, cioè che le sono equivalenti. Inoltre può accadere che si trovino varianti equivalenti in grado di generare la sequenza di stringhe richiesta in modo sensibilmente più efficiente.
Come potremo constatare questo è un fatto generale: tutte le MSPG, come tutte le MSP, ovvero come tutte le procedure, possiedono numerose varianti equivalenti e allontanandosi dagli schemi organizzativi generali si possono definire macchine più efficienti; va osservato che queste tendenzialmente sono più complesse e nel corso degli studi vengono precisate in tempi successivi.

B18:c. Insiemi procedurali

B18:c.01 Consideriamo una MSPG $\mathbf{M}$ dotata di un unico nastro di uscita che denotiamo con $U_{\mathbf{M}}$; diciamo inoltre **alfabeto di uscita** l'alfabeto usato per scrivere le stringhe emesse su $U_{\mathbf{M}}$ e denotiamolo con $\mathsf{A}_{\mathbf{M}}$.
La sequenza potenziale di stringhe, finita o illimitata, che $\mathbf{M}$ è in grado di emettere sopra $U_{\mathbf{M}}$ si dice costituire una **lista [generata] da MSPG**.
Questa lista verrà denotata con $\mathbf{M}^{\mathcal{G}}$. Più in particolare se la sequenza potenziale risulta illimitata la chiamiamo **lista [generata] da MSPGI**, mentre in caso contrario, cioè se viene generata una lista finita, la possiamo chiamare **lista [generata] da MSPGF**.
Se nella posizione i di un nastro T, in particolare di $U_{\mathbf{M}}$, si trova registrata una stringa w si dice che $\langle w, i\rangle$ costituisce una **occorrenza** di w su T e si scrive $w \in \mathbf{M}^{\mathcal{L}}$. Se z è una stringa sull'alfabeto $\mathsf{A}_{\mathbf{M}}$ che non si trova su $U_{\mathbf{M}}$ scriviamo $z \notin \mathbf{M}^{\mathcal{L}}$.

Le liste di base generate dalle MSPGI considerate in :a.10 e :a.11 per introdurre, risp., l'insieme dei numeri interi e l'insieme delle stringhe sopra l'alfabeto A le denotiamo, risp., con $|^*$ e con A^*. Altre liste generate da MSPGI sono quelle usate per introdurre gli insiemi di notazioni posizionali degli interi naturali $\mathbb{N}_{[B]}$ con $B = 2, 3, ...$ (e il più astratto insiemi degli interi naturali $\mathbb{N}$): si tratta di liste sull'alfabeto delle cifre 0, 1, ... e B-1, che aut si riducono alla stringa 0 aut iniziano con una cifra diversa da 0.
Altre liste da MSPGI si ottengono con macchine che sono in grado di rendere disponibili le stringhe di un insieme di base ma emettono sul nastro di uscita solo quelle che superano il vaglio di uno specifico algoritmo selettore.

B18:c.02 Date due liste da MSPG ${\mathbf{M}_1}^{\mathcal{G}}$ ed ${\mathbf{M}_2}^{\mathcal{G}}$ si dice che la prima è sottolista della seconda sse ogni stringa che occorre nella prima occorre anche nella seconda. Osserviamo esplicitamente che tra le sottoliste di una data lista da MSPG L si può considerare la stessa L; le restanti si dicono sottoliste proprie.

Occorre ra segnalare che per una data MSPG può accadere che in una data fase di studio, non si sia in grado di decidere se si tratta di una MSPGF o di una MSPGI. Va anche segnalato che si incontrano coppie $\langle \mathbf{M}_1, \mathbf{M}_2 \rangle$ tali che risulta difficile stabilire se ${\mathbf{M}_1}^{\mathcal{G}}$ è sottolista di ${\mathbf{M}_2}^{\mathcal{G}}$, oppure ${\mathbf{M}_2}^{\mathcal{G}}$ è sottolista di ${\mathbf{M}_1}^{\mathcal{G}}$, oppure se ogni stringa occorrenza di una lista lo sia anche per l'altra, oppure se vi siano occorrenze di una lista che non sono occorrenze dell'altra, oppure (in particolare) se esse abbiano qualche o nessuna componente in comune.

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

Va segnalato e0plicitamente che si incontrano coppie $\langle \mathsf{M}, \mathsf{S} \rangle$, per le quali in una data fase di studio non si è in grado di stabilire se la sottolista relativa al selettore sia finita o illimitata.

Incontreremo inoltre (v. :c.02) altre liste da MSPGI le cui componenti sono coppie di stringhe, sequenze di un dato numero di stringhe o altre composizioni articolate riconducibili a stringhe ciascuna delle quali si può scegliere arbitrariamente in una lista di base. Tali liste se di esse interessano varie sottoliste le chiameremo **liste da MSPG ambiente**.

B18:c.03 Tutte le liste da MSPG viste in precedenza, liste di base e liste ottenute per selezione delle precedenti, sono non ripetitive, cioè sono tali che su due diverse posizioni del nastro di emissione si trovano due stringhe diverse. A fortiori sono non ripetitive le sottoliste delle liste da MSPG non ripetitive.

Si trovano invece facilmente liste ripetitive ottenute applicando un trasduttore, cioè un algoritmo di trasformazione, a ciascuna delle stringhe rese disponibili da una MSPG che genera una lista non ripetitiva, in particolare una lista di base o una lista 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 n_2 e successivamente calcola ed emette la somma $SB(n)$ delle cifre di tale rappresentazione. Questa procedura, come visto in :a.01, genera una lista illimitata di occorrenze di interi naturali che presenta numerose ripetizioni: ad esempio si ha $SB(5) = SB(6) = SB(9) = 2$. Un esempio ancor più evidente è dato dalla MSPGI che fa corrispondere ad ogni intero naturale la lunghezza della sua notazione posizionale in una qualche base B.

Si può però ridurre ogni lista da MSPG ripetitiva L ad una lista (finita o illimitata) non ripetitiva: basta modificare la MSPG che genera la L facendo seguire alla individuazione di una nuova stringa una manovra che controlli se tale stringa è già presente sul nastro di emissione e qualora sia già stata generata una sua replica eviti di registrare in uscita la nuova copia.

Da molte liste ripetitive illimitate mediante l'eliminazione delle ripetizioni si ottengono liste non ripetitive 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 $\langle m, n \rangle \in \mathbb{P} \times \mathbb{P}$ otytenuta scorrendo questo quadrato cartesiano con procedimento diagonale.

Si trovano però anche liste da MSPGI illimitate ripetitive dalle quali si ottengono corrispondenti lista non ripetitive finite: un semplice 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 non ripetitiva contiene solo le 100 stringhe formate da due cifre decimali (abbiamo trascurato di distinguere 0 da 00, 1 da 01, ... , 9 da 09) .

B18:c.04 In genere data una MSPG $\mathbf{M}$ non è difficile ottenere una diversa MSPG che genera la stessa lista ${\mathbf{M}}^{\mathcal{G}}$. Ad esempio la lista degli interi pari si può generare con una macchina che inizia ponendo

0 sul nastro di uscita e quindi procede ad aggiungere dopo ogni pari $2k$ l'intero $2k+2$ facilmente ottenibile; tale lista 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 si dicono **equivalenti -list**.

Due **liste** da MSPG si dicono **equivalenti -set** sse ogni stringa che occorre nella prima occorre anche nella seconda e viceversa. Ad esempio sono equivalenti -set una lista ripetitiva e quella ottenuta eliminando le sue stringhe ripetute. Due MSPG si dicono **equivalenti -set** sse generano liste equivalenti -set.

Due liste da MSPGI equivalenti -set sono la lista $\mathsf{L}_{\mathtt{a,b,c}}$ delle stringhe sui caratteri `a`, `b` e `c` generata con la procedura descritta in :a.14 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 equivalenti -set degne di nota riguardano le coppie costituenti $\mathbb{N} \times \mathbb{N}$: la prima riguarda la generazione che procede per diagonali, la seconda la generazione che procede per ganci.

B18:c.05 Introduciamo ora una entità per la quale useremo il termine "insieme" in quanto presenta varie caratteristiche in comune con gli insiemi finiti. Come un insieme finito si ottiene per astrazione da una classe di elenchi equivalenti per occorrenza, uno degli insiemi che stiamo per introdurre si ottiene da una classe di liste equivalenti -set.
Consideriamo una MSPG $\mathbf{M}$, il suo nastro di uscita che denotiamo con $U_{\mathbf{M}}$, il suo alfabeto di uscita che scriviamo $\mathsf{A}_{\mathbf{M}}$ e la lista che essa genera $\mathbf{M}^{\mathcal{G}}$.
Si dice **insieme generato dalla MSPG $\mathbf{M}$** la classe di equivalenza -set delle liste della quale fa parte $\mathbf{M}^{\mathcal{G}}$; tale entità si denota con $\mathbf{M}^{\mathcal{L}}$. Se $\mathbf{M}'$ denota un'altra MSPG che genera una lista equivalente -set alle precedenti, cioè appartenente alla stessa classe di equivalenza di cui fa parte $\mathbf{M}^{\mathcal{G}}$, si ha l'uguaglianza $\mathbf{M}^{\mathcal{G}} = \mathbf{M}'^{\mathcal{G}}$.
Ogni insieme generato da una MSPG verrà detto **insieme procedurale**, oppure in forma abbreviata **insieme -Prcd**, oppure **insieme ricorsivamente enumerabile**, oppure **linguaggio ricorsivamente enumerabile** o anche **linguaggio REn**.

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

B18:c.06 Non è difficile descrivere alcune MSPG in grado di emettere coppie di numeri interi naturali o coppie di stringhe sopra un alfabeto A o coppie di entità appartenenti ad uno o due insiemi procedurali. Una tale lista di coppie si dice **relazione [binaria] procedurale**, o, in breve, **relazione [binaria] -Prcd**, oppure **relazione [binaria] da MSPG**, oppure **relazione [binaria] ricorsivamente enumerabile**.
Se $\mathbf{R}$ denota una MSPG in grado di generare coppie di un insieme procedurale, per tale insieme usiamo la scrittura

$$\mathsf{L} = \mathbf{R}^{\mathcal{G}} = \big\langle \langle a_0, b_0\rangle, \langle a_1, b_1\rangle, \langle a_2, b_2\rangle, ..., \langle a_n, b_n\rangle, ...\big\rangle \ ,$$

Useremo anche come notazione equivalente alla precedente la scrittura più concisa

$$\mathsf{L} = \mathbf{R}^{\mathcal{G}} = \big\langle n \in \mathbb{N} :| \ \langle a_n, b_n\rangle\big\rangle \ .$$

In questa compare l'entità NMb sul quale dobbiamo ritornare.

È semplice descrivere due varianti della $\mathbf{R}$ le quali procedono a generare, risp., la lista delle prime componenti $\mathsf{L}_1 := \langle a_0, a_1, a_2, ..., a_n, ...\rangle$ e la lista delle seconde componenti $\mathsf{L}_2 := \langle b_0, b_1, b_2, ..., b_n, ...\rangle$.

Va osservato che, anche se la lista L è non ripetitiva, ciascuna delle due liste L_1 ed L_2 può essere ripetitiva.
Come per le relazioni finite, gli insiemi individuati da L_1 ed L_2 sono detti, risp., **dominio** e **codominio** della relazione e si denotano con $\mathrm{dom}(\mathbf{R}^{\mathcal{G}})$ e $\mathrm{cod}(\mathbf{R}^{\mathcal{G}})$.
Come per le relazioni finite, può accadere che questi due insiemi -Prcd coincidano, non coincidano (eventualmente siano disgiunti) ma si collochino naturalmente in un loro sovrainsieme (ad esempio siano due insiemi di stringhe sopra lo stesso alfabeto), oppure si collochino in due insiemi ben distinti.
Un esempio di questa seconda situazione si ricava dalla lista 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 ed il codominio è un particolare insieme illimitato di stringhe sopra l'alfabeto della lingua naturale.

B18:c.07 Una relazione procedurale si dice **funzione procedurale**, o **funzione -Prcd**, o **funzione da MSPG** sse il suo dominio può essere fornito da una lista non ripetitiva.
Esempi di funzioni -prcd sono gli insiemi di coppie della forma

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

dove $f(n)$ denota un valore numerico associato all'intero naturale n da un definito algoritmo. Esempi di tale valore calcolabile sono la $SB(n)$ e la riduzione alle ultime due cifre incontrate in :c.03.

Si parla invece di **biiezione procedurale** o dn biiezione -Prcd sse sono non ripetitivi sia il suo dominio che il suo codominio.
Si osserva che da una coppia di liste non ripetitive equivalenti -set di elementi di un ambiente procedurale U si ricava una permutazione di tale U.
Anche queste costruzioni possono essere considerate generalizzazioni di quelle relative agli insiemi e alle liste finite. Ci limitiamo a segnalare che con richieste molto vicine a quelle utilizzate per i tipi particolari di relazioni e di funzioni finite si possono definire i corrispondenti tipi di relazioni e di funzioni procedurali.

B18:c.08 Tornando a considerare la MSPG $\mathbf{M}$ di :b.06, è semplice descrivere la variante della $\mathbf{M}$ la quale invece di generare la lista $\mathbf{M}^{\mathcal{G}} = \langle w_0, w_1, w_2, ..., w_n, ...\rangle$ genera la lista di coppie

$$\big\langle \langle 0, w_0\rangle, \langle 1, w_1\rangle, \langle 2, w_2\rangle, ..., \langle n, w_n\rangle, ...\big\rangle\ .$$

Questa lista individua una funzione da MSPG funzione dal dominio $\mathbb{N}$ nell'insieme di base $\mathsf{A}_{\mathbf{M}}{}^*$. Le due liste precedenti spesso possono essere identificate; di conseguenza l'insieme $\mathbf{M}^{\mathcal{G}}$ si può denotare anche con $\mathrm{cod}(\mathbf{M}^{\mathcal{G}})$.

Se $\mathbf{M}$ è una MSPGF, $\mathbf{M}^{\mathcal{L}}$ è l'insieme (finito) delle stringhe che sono registrate sul nastro di uscita quando la macchina si arresta oppure quando un algoritmo o una dimostrazione hanno garantito che la macchina non potrà emettere stringhe diverse da quelle già registrate e quindi che consentono di interrompere la sua evoluzione con la certezza di avere ottenute tutte le stringhe di $\mathbf{M}^{\mathcal{L}}$.
Nel caso in cui $\mathbf{M}$ sia la MSPGI che genera $|^*$, viene generato l'insieme delle rappresentazioni unadiche degli interi naturali e si può scrivere $w \in \mathbf{M}^{\mathcal{L}}$ per ogni w stringa costituita da un numero qualsiasi di segni "|".
Nel caso in cui A denota un insieme finito di simboli ed $\mathbf{M}$ è la MSPGI che genera A^*, viene generato l'insieme delle stringhe sull'alfabeto A e si può scrivere $w \in \mathbf{M}^{\mathcal{L}}$ per ogni w stringa costituita da un numero qualsiasi di caratteri di A.

B18:c.09 Nel caso in cui $\mathbf{M}$ sia la MSPGI che genera $\mathbb{N}_{[B]}$ in relazione ad un intero $B = 2, 3, ...$, viene

generato l'insieme delle notazioni in base B degli interi naturali e si può scrivere $w \in \mathbf{M}^{\mathcal{L}}$ per ogni w scrittura in base B di un numero intero naturale.

Va osservato che le elaborazioni sui numeri interi naturali si possono effettuare servendosi di ciascuna delle sue rappresentazioni $\mathbb{N}_{[B]}$ senza differenze sostanziali, ma solo con differenze concernenti dettagli operativi. Le diverse notazioni posizionali sono equivalenti operativamente, nel senso che le elaborazioni condotte su notazioni in due diverse basi si possono porre in collegamento servendosi di opportune trasformazioni di trascodifica e due elaborazioni su diverse notazioni conducono a risultati trasformabili l'uno nell'altro e quindi da considerare equivalenti.

È allora lecito individuare come **insieme degli interi naturali** la soggettizzazione delle diverse rappresentazioni $\mathbb{N}_{[B]}$ per $B = 2, 3, ...$, nonché della notazione unadica (non avendo qui peso la sua inefficienza). Questo insieme è necessariamente numerabile, come lo sono tutte le notazioni sopra richiamate e le molte altre rappresentazioni che sono state studiate. Esso si denota con $\mathbb{N}$.

D'ora in avanti in linea di massima trascureremo di distinguere fra un insieme come $\mathbb{N}$ e le sue diverse rappresentazioni; queste scritture le chiameremo **codifiche degli interi naturali**.

I numeri naturali ottenuti da definizioni o costruzioni specifiche in genere, per la forza dell'abitudine, saranno presentati attraverso la loro notazione decimale.

B18:c.10 Consideriamo due MSPG $\mathbf{M}$ e $\mathbf{N}$ ed i due corrispondenti insiemi procedurali $\mathsf{L}_M := \mathbf{M}^{\mathcal{G}}$ e $\mathsf{L}_N := \mathbf{N}^{\mathcal{G}}$. Si dice che il primo è **sottoinsieme** del secondo sse ogni $w \in \mathsf{L}_M := Mbf^{\mathcal{G}}$ appartiene anche a $\mathsf{L}_N := \mathbf{N}^{\mathcal{G}}$; in tal caso si scrive $\mathsf{L}_M \subseteq \mathsf{L}_N$. Si dice che L_M è **sottoinsieme proprio** di L_N e si scrive $\mathsf{L}_M \subset \mathsf{L}_N$ sse $\mathsf{L}_M \subseteq \mathsf{L}_N$ e $\mathsf{L}_M \neq \mathsf{L}_N$.

Evidentemente un insieme procedurale fornito da una sottolista propria della $\mathbf{N}^{\mathcal{G}}$ è sottoinsieme dell'insieme corrispondente a $\mathbf{N}^{\mathcal{G}}$.

Consideriamo due insiemi procedurali L_1 e L_2; come per gli insiemi finiti scriviamo $\mathsf{L}_1 \subseteq \mathsf{L}_2$ e diciamo che L_1 è **sottoinsieme** di L_2 sse ogni $w \in \mathsf{L}_1$ appartiene anche a L_2; se inoltre si trova almeno una stringa che appartiene a L_2 e non a L_1 si scrive $\mathsf{L}_1 \subset \mathsf{L}_2$ e si dice che L_1 è **sottoinsieme proprio** di L_2.

Contrariamente a quanto accade agli insiemi finiti, può accadere che si incontrino coppie di insiemi procedurali per le quali in una data fase degli studi relativi si incontrino difficoltà sostanziali, non dovute solo a questioni di efficienza, per decidere se uno dei due è sottoinsieme o sottoinsieme proprio dell'altro.

Si possono avere MSPG che possono procedere senza mai fermarsi, ma che generano linguaggi finiti, vuoi perché da un certo passo in poi non generano altre stringhe, vuoi perchè 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 le è equivalente -set. La effettiva attuazione di una tale sostituzione potrebbe incontrare difficoltà in quanto, in linea di massima, dipende dalle conoscenze che in una data fase degli studi si posseggono sulle MSPG in esame in relazione alle stringhe che esse possono o non possono generare.

B18:c.11 Definiamo **insieme infinito** un insieme che si può porre in corrispondenza biunivoca con un suo sottoinsieme proprio.

Si osserva che attualmente questa è una definizione fiduciaria, in quanto non si è ancora data una definizione generale di insieme e tanto meno quella di corrispondenza biunivoca tra insiemi qualsiansi. Quindi ce ne serviremo solo nei casi nei quali si trattano insiemi e corrispondenze biunivoche particolari per le quali si conoscono definizioni utilizzabili esplicite o che si possono ragionevolmente presumere esplicitabili.

(1) Prop.: Gli insiemi associati a liste da MSPGI non ripetitive sono insiemi infiniti.
Dim.: Da una MSPGI $\mathbf{M}$ che genera la lista (illimitata) $\mathsf{E} = \mathbf{M}^{\mathcal{G}}$ non ripetitiva si ottengono facilmente MSPGI che generano sottoliste di $\mathbf{M}^{\mathcal{G}}$ illimitate (ed evidentemente non ripetitive). Si hanno ad esempio macchine che trascurano alcune stringhe iniziali della $\mathbf{M}^{\mathcal{G}}$, oppure macchine che trascurano infinite stringhe di tale lista servendosi di un qualche selettore algoritmico, ma che riescono ad accettarne infinite. In particolare è semplice descrivere la macchina che seleziona tra gli interi naturali solo i pari e quella che si serve della $\mathbf{M}$ ma emette della lista $\mathbf{M}^{\mathcal{G}}$ solo una stringa su 3, grazie ad un uso facilmente immaginabile di un contatore ciclico a 3 stati.

Scriviamo $\langle a_1, a_2, ..., a_n, ...\rangle := \mathbf{M}^{\mathcal{G}}$, denotiamo con $\mathbf{N}$ la macchina che genera una sottolista illimitata della precedente e scriviamo $\langle b_1, b_2, ..., b_n, ...\rangle := \mathbf{N}^{\mathcal{G}}$. È facile definire una macchina $\mathbf{R}$ che chiede alternatamente alla $\mathbf{M}$ e alla $\mathbf{N}$ di procedere alla emissione di una propria nuova stringa e che dopo aver ottenuta la b_n emette la coppia $\langle a_n, b_n\rangle$ sul proprio nastro.
Evidentemente $\mathbf{R}^{\mathcal{G}} = \big\langle\langle a_1, b_1\rangle, \langle a_2, b_2\rangle, ..., \langle a_n, b_n\rangle, ...\big\rangle$ e la nonripetitività di $\mathbf{M}$ e di $\mathbf{N}$ implica che questa successione di coppie generata individui una biiezione fra E e $\mathrm{cod}(\mathbf{N}^{\mathcal{G}})$ ∎

B18:c.12 **(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 si presentino). Una tale lista può essere registrata in un tempo finito (anche se non facilmente prevedibile) sopra un nastro T 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.
Procederemo ora costruttivamente ad 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 di insiemi finiti (unione, intersezione, prodotto cartesiano, ...).
In tal modo saremo in grado di definire un complesso costituito da operazioni binarie e unarie e di relazioni riguardanti insiemi procedurali che permetterà di tenere sotto controllo la collezione di tali entità **SetPrcd** mediante espressioni di genere algebrico e relazionale.

B18:c.13 Gli sviluppi sopra annunciati riguardano due MSPG $\mathbf{M}_1$ ed $\mathbf{M}_2$ ed i relativi insiemi procedurali generati $\mathsf{L}_1 := \mathbf{M}_1^{\mathcal{G}}$ ed $\mathsf{L}_1 := \mathbf{M}_1^{\mathcal{G}}$. Per talune argomentazioni risulta vantaggioso supporre che $\mathbf{M}_1$ ed $\mathbf{M}_2$ siano non ripetitive, supposizione che è sempre lecita, come abbiamo visto.

Per tenere sotto controllo un linguaggio procedurale potrebbe essere utile disporre di una sua MSPG che genera le sue stringhe secondo un ordine che possa essere concretamente tenuto sotto controllo.
Più circostanziatamente diciamo **ordinamento totale algoritmico** di un insieme ambiente un suo ordinamento totale per il quale si dispone di un **algoritmo di confronto** che per ogni duetto di stringhe (diverse) dell'ambiente consente di stabilire quale sia la precedente.
Sono ordinamenti totali algoritmici: l'ordinamento $\leq$ per i numeri interi naturali, l'ordinamento len-lxg per le stringhe sopra un alfabeto finito e gli ordinamenti diagonale e per ganci di $\mathbb{N} \times \mathbb{N}$.

Data una MSPG, la si può modificare in modo che in ogni stadio della sua evoluzione presenti le stringhe sul suo nastro di uscita U secondo un ordine algoritmico (e di conseguenza senza ripetizioni).

Basta infatti aggiungerle un dispositivo che, dopo che essa ha reso disponibile una nuova stringa, stabilisca mediante l'algoritmo di confronto se è già presente su U e solo nel caso si sia verificata la sua assenza la aggiunga collocandola in modo da mantenere la lista registrata sopra U nell'ordine richiesto.

Un ordinamento di questo genere si può chiamare **ordine sequenziale algoritmico**.

B18:c.14 Occorre osservare che questa possibilità di mantenere ordinato il nastro di uscita in ogni stadio evolutivo della macchina è cosa ben diversa della disponibilità di una MSPG che presenti le stringhe che sta generando sopra un nastro di uscita immodificabile secondo un ordinamento totale algoritmico.
Si individuano macchine $\mathbf{M}$ che rendono disponibili le stringhe in ordine diverso da quello associato ad un algoritmo di confronto al quale si vuole fare riferimento. Quindi se ad un certo stadio della evoluzione della $\mathbf{M}$ ci chiediamo se una data stringa w sull'alfabeto di uscita apparterrà alla lista che sarà generata, aut si osserva sul nastro di uscita U la stringa w, aut non la si osserva e si potrebbe avere il dubbio se in uno stadio successivo la w sarà emessa o meno.
In effetti si incontrano MSPG che presentano una difficoltà che si rivela grave: quando si cerca di individuare una tale macchina $\mathbf{M}$ una procedura in grado di decidere se una generica stringa sull'alfabeto di uscita appartiene o meno al linguaggio da essa generato, in una data fase degli studi relativi non si riesce ad individuare tale procedura decisionale. Di più incontreremo MSPG per le quali si dimostra impossibile individuare la auspicata procedura decisionale.
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 il corrispondente linguaggio secondo un determinato ordine algoritmico.

B18:c.15 Abbiamo vista l'opportunità di servirsi di termini come nastro di lunghezza infinita, risorse che crescono all'infinito ed insieme infinito. Vedremo più oltre (B19:f) la necessità di stabilire una gerarchia tra infiniti che risultano non confrontabili. Altri infiniti si incontreranno con la definizione degli interi negativi (B20:a) e dei primooggetti geometrici (B21:-B23:, B30:, ...) e con l'introduzione delle varie accezioni della nozione di limite (B35:, I12:, I13:, I16:, ...).
Si intravvede quindi che risulti opportuno introdurre delle notazioni per le varie entità infinite.
Un primo simbolo è $+\infty$ e ad esso diamo due primi significati: esso può esprimere il numero delle caselle di un nastro illimitato e può contribuire alla notazione $\mathbf{[}k : +\infty\mathbf{)}$ con la quale vogliamo individuare l'insieme (infinito) di tutti gli interi positivi maggiori di un dato intero positivo k.

B18:d. Composizioni di insiemi procedurali

B18:d.01 Ci proponiamo di definire, coerentemente a quanto fatto per gli insiemi finiti, le composizioni booleane sugli insiemi procedurali; giungeremo a dimostrare che anche gli insiemi così ottenuti sono procedurali, e che le suddette composizioni sono operazioni binarie ed unarie per le quali valgono proprietà che generalizzano quelle riguardanti gli insiemi finiti.
Consideriamo, per $i \in \{1, 2\}$, le due MSPG $\mathbf{M}_i$ ed i corrispondenti linguaggi procedurali generati $\mathsf{L}_i := \mathbf{M}_i^{\mathcal{G}}$; denotiamo inoltre con A_i l'alfabeto che la $\mathbf{M}_i$ usa per il suo nastro di emissione e l'alfabeto unione con $\mathsf{A} := \mathsf{A}_1 \cup \mathsf{A}_2$.

Diciamo **unione** di L_1 ed L_2 il linguaggio su A le cui strighe appartengono o ad L_1 o ad L_2 (o a entrambi).

Diciamo **intersezione** di L_1 ed L_2 il linguaggio su A le cui strighe appartengono sia ad L_1 che ad L_2.

Diciamo **eliminazione** di L_2 da L_1 il linguaggio su $\mathsf{A}_1 \cap \mathsf{A}_2$ le cui strighe appartengono ad L_1 ma non ad L_2.

Diciamo **complementazione** di L_i entro ${\mathsf{A}_i}^*$ il linguaggio su A_i le cui strighe non appartengono ad L_i.

Diciamo **differenza simmetrica** di L_1 ed L_2 il linguaggio su A costituito dalle strighe che appartengono ad L_1 ma non ad L_2 e dalle strighe che appartengono ad L_2 ma non ad L_1.

Diciamo **prodotto cartesiano** di L_1 ed L_2 il linguaggio delle coppie di stringhe

$$\langle w_1, w_2 \rangle \in \mathsf{L}_1 \times \mathsf{L}_2 \subseteq {\mathsf{A}_1}^* \times {\mathsf{A}_2}^* \ .$$

B18:d.02 Descriviamo una MSPG $\mathbf{M}_\cup$ che si serve delle due MSPG $\mathbf{M}_1$ e $\mathbf{M}_2$ e che chiamiamo **composizione per l'unione** di tali macchine.

$\mathbf{M}_\cup$ procede per fasi successive in ciascuna delle quali chiede alternatamente ad $\mathbf{M}_1$ e ad $\mathbf{M}_2$ di effettuare un passo evolutivo e quando una di esse rende disponibile una stringa la emette sul proprio nastro di uscita che denotiamo con $U_\cup$.
È evidente che ogni stringa che compare su $U_\cup$ deve appartenere ad L_1 o ad L_2 (o ad entrambi questi insiemi procedurali), come è evidente che, per $i = 1, 2$, ogni stringa emessa da $\mathbf{M}_i$ dopo n_i passi deve comparire su $U_\cup$ dopo $2\,n_i$ passi al più.
L'insieme procedurale generato dalla $\mathbf{M}_\cup$ è evidentemente l'unione di $\mathsf{L}_1 := {\mathbf{M}_1}^{\mathcal{G}}$ ed $\mathsf{L}_2 := {\mathbf{M}_2}^{\mathcal{G}}$.

Osserviamo che se in particolare una delle $\mathbf{M}_i$ (o entrambe) emette un numero finito di stringhe la costruzione rimane valida: basta considerare che i passi della macchina che ha concluso l'emissione si riducono ad inazioni. Quindi la suddescritta costruzione che si serve di due MSPG è un'estensione dell'unione di due insiemi finiti.

Va notato che anche se $\mathbf{M}_1$ ed $\mathbf{M}_2$ sono non ripetitive, la macchina $\mathbf{M}_\cup$ potrebbe esserlo; non è però difficile dotare $\mathbf{M}_\cup$ di un dispositivo che evita la ripetitività non consentendo di emettere sul suo nastro $U_\cup$ una stringa già presente.
Inoltre anche se $\mathbf{M}_1$ ed $\mathbf{M}_2$ emettono stringhe seguendo un ordine sequenziale algoritmico, la $\mathbf{M}$ potrebbe non avere questa proprietà. Tuttavia per un qualsiasi ordinamento sequenziale algoritmico Ω_K basato sopra un algoritmo di confronto K delle stringhe su A, è possibile dotare la $\mathbf{M}_\cup$ di un dispositivo che, servendosi di K, si incarica di inserire ogni nuova stringa emessa nella posizione di $U_\cup$ che rende la sequenza ivi registrata ordinata compatibilmente con Ω_K.

B18:d.03 Definiamo una MSPG $\mathbf{M}_\cap$ che si serve di due date MSPG $\mathbf{M}_1$ e $\mathbf{M}_2$ e che chiamiamo **composizione per l'intersezione** delle macchine date.
Anche la MSPG $\mathbf{M}_\cap$ procede per fasi successive in ciascuna delle quali chiede alternatamente ad $\mathbf{M}_1$ e ad $\mathbf{M}_2$ di effettuare un passo evolutivo.
Entrambe queste due macchine devono mantenersi il proprio nastro di emissione che può essere più chiaro pensare in grado di evitare le ripetizioni; in tal modo la $\mathbf{M}_\cap$ risulta in grado di registrare sul proprio nastro di uscita, che denotiamo con $U_\cap$, le sole stringhe che vengono generate da entrambe le macchine, stringhe "approvate" dopo ogni emissione di una nuova stringa da parte della macchina $\mathbf{M}_i$ (i=1,2) mediante l'esame del nastro di uscita della macchina alternativa $\mathbf{M}_{3-i}$.
L'insieme procedurale generato dalla $\mathbf{M}_\cap$ è evidentemente l'intersezione di $\mathsf{L}_1 := {\mathbf{M}_1}^{\mathcal{G}}$ e $\mathsf{L}_2 := {\mathbf{M}_2}^{\mathcal{G}}$.

Se una delle due macchine emette una lista finita e si è in grado di decidere ad un certo punto che questa macchina si può arrestare, è lecito arrestare anche l'altra macchina ed affermare che la $\mathbf{M}_\cap$ è una MSPGF.

Quindi, come l'unione di due insiemi procedurali, anche la loro intersezione può considerarsi un'estensione dell'intersezione di insiemi finiti.

Consideriamo il caso in cui sia $\mathbf{M}_1$ che $\mathbf{M}_2$ 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 $\mathsf{L}_\cap = \mathsf{L}_1 \cap \mathsf{L}_2$, ordine che denotiamo con Ω_K, dove con K si denota un opportuno algoritmo di confronto.
Si osserva che fissato un qualsiasi algoritmo di confronto K si può ottenere un ordine sequenziale algoritmico Ω_K basato sopra un algoritmo di confronto K scelto ad arbitrio munendo $\mathbf{M}_\cap$ di un opportuno dispositivo di mantenimento di tale ordine sulle 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 costruisce $\mathsf{L}_\cap$. Evidentemente conviene che Ω_K sia coerente con Ω_1 ed Ω_2, in quanto i controlli sui nastri per le singole macchine possono essere evitati.

Consideriamo vi sia interesse per una particolare stringa w e per la sua appartenenza ad $\mathsf{L}_1 \cap \mathsf{L}_2 = \mathbf{M}_1^G cl \cap \mathbf{M}_2^G cl$. Se ad un certo stadio dell'evoluzione della $\mathbf{M}_\cap$ osserviamo che l'ha emessa solo $\mathbf{M}_1$, rimane il dubbio se $\mathbf{M}_2$ la emetterà successivamente (ed in tal caso $w \in \mathsf{L}_1 \cap \mathsf{L}_2$, oppure non la emetterà, ed in tal caso $w \notin \mathsf{L}_1 \cap \mathsf{L}_2$.

B18:d.04 Introduciamo una MSPG $\mathbf{M}_\setminus$ che si serve di due date MSPG $\mathbf{M}_1$ e $\mathbf{M}_2$ e che chiamiamo **composizione per l'eliminazione** fra le tali macchine. Il nastro di emissione di $\mathbf{M}_\setminus$ lo denotiamo con $U_\setminus$.
Anche la MSPG $\mathbf{M}_\setminus$ procede per fasi successive in ciascuna delle quali chiede alternatamente ad $\mathbf{M}_1$ e ad $\mathbf{M}_2$ di effettuare un passo evolutivo.
Per semplicità supponiamo che le $\mathbf{M}_i$ emettano sul proprio nastro liste non ripetitive. La $\mathbf{M}_\setminus$ adotta come nastro di uscita $U_\setminus$ il nastro sul quale la $\mathbf{M}_1$ emette le stringhe che genera, ma riservandosi della possibilità di cancellare ciascuna di esse. In ogni fase nella quale la $\mathbf{M}_2$ emette una (nuova) stringa w_2 la $\mathbf{M}_\setminus$ controlla se essa non compare su $U_\setminus$ e se la trova la "cancella" sostituendola con un segno neutro. In tal modo la $\mathbf{M}_\setminus$ viene ad avere su $U_\setminus$, prima o poi, ciascuna delle stringhe che sono generate da $\mathbf{M}_1$ ma non da $\mathbf{M}_2$.
L'insieme procedurale generato dalla $\mathbf{M}_\setminus$, evidentemente, è l'eliminazione da $\mathsf{L}_1 := {\mathbf{M}_1}^{\mathcal{G}}$ di $\mathsf{L}_2 := {\mathbf{M}_2}^{\mathcal{G}}$.
Se una delle due macchine $\mathbf{M}_i$ emette una lista finita e si sa riconoscere quando può essere arrestata, risulta sufficiente far proseguire solo la macchina alternativa.
Come le due precedenti costruzioni, quindi, anche l'eliminazione fra due insiemi procedurali può essere considerata un'estensione della eliminazione fra due insiemi finiti.

Anche per questa composizione di MSPG si può avere interesse per una particolare stringa w e per la sua appartenenza al linguaggio risultato. Evidentemente questa stringa deve appartenere a L_1, mentre per la sua appartenenza ad L_2 si può avere una situazione di attesa nell'incertezza simile a quella accennata per l'intersezione.
Se ad un certo stadio dell'evoluzione della $\mathbf{M}_s etminus$ osserviamo che l'ha emessa solo $\mathbf{M}_1$, rimane il dubbio se $\mathbf{M}_2$ la emetterà successivamente (ed in tal caso $w \notin \mathsf{L}_1 \setminus \mathsf{L}_2$, oppure non la emetterà, ed in tal caso $w \in \mathsf{L}_1 \setminus \mathsf{L}_2$. Se invece ad un certo stadio dell'evoluzione la w non è stata emessa dalla $\mathbf{M}_1$ e dalla $\mathbf{M}_2$ rimane il dubbio se essa verrà emessa da una o entrambe queste macchine generatrici.

Anche per questa composizione può accadere che sia $\mathbf{M}_1$ che $\mathbf{M}_2$ emettano stringhe seguendo un ordine sequenziale algoritmico ed interessi mantenere questa proprietà per la $\mathbf{M}_\setminus$. Anche in questo caso valgono considerazioni simili a quelle esposte per la macchina intersezione.

B18:d.05 Introduciamo una MSPG $\mathbf{M}_\Delta$ che si serve di due date MSPG $\mathbf{M}_1$ e $\mathbf{M}_2$ e che chiamiamo **composizione per la differenza simmetrica** delle macchine date. Il suo nastro di uscita lo denotiamo con $U_D lt$.

Anche la MSPG $\mathbf{M}_\Delta$ procede per fasi successive in ciascuna delle quali chiede alternatamente ad $\mathbf{M}_1$ e ad $\mathbf{M}_2$ di effettuare un passo evolutivo.
Queste due macchine mantengono i rispettivi nastri di uscita U_1 e U_2 e per semplicità supponiamo ancora che ciascuna di esse emetta sul proprio nastro una lista non ripetitiva. La $\mathbf{M}_\Delta$ ad ogni nuova emissione delle $\mathbf{M}_i$ tenta un aggiornamento di $U_D lt$ che può consistere in una aggiunta o in una cancellazione.
Quando $\mathbf{M}_i$ emette una nuova stringa w_i, se essa compare su U_{3-i} non viene emessa su $U_D lt$ ma viene cancellata da tale nastro; se viceversa non compare su U_{3-i} viene aggiunta ad $U_D lt$. In tal modo la $\mathbf{M}_\Delta$ viene ad avere sul suo nastro di emissione, prima o poi, ciascuna delle stringhe che sono generate da $\mathbf{M}_1$ ma non da $\mathbf{M}_2$ e ciascuna delle stringhe generate da $\mathbf{M}_2$ ma non da $\mathbf{M}_1$.
L'insieme procedurale generato dalla $\mathbf{M}_\Delta$ evidentemente è la differenza simmetrica di $\mathsf{L}_1 := {\mathbf{M}_1}^{\mathcal{G}}$ e $\mathsf{L}_2 := {\mathbf{M}_2}^{\mathcal{G}}$

Se una delle due macchine $\mathbf{M}_i$ emette una lista finita e si sa conoscere la sua conclusione, risulta sufficiente far proseguire le operazioni della sola macchina alternativa.
Come per le precedenti costruzioni si trova che la differenza simmetrica fra due insiemi procedurali va considerata una estensione della differenza simmetrica di due insiemi finiti.

Valgono poi considerazioni analoghe sulla possibilità di avere sul nastro sul quale si costruisce $\mathsf{L}_1 \Delta \mathsf{L}_2$ 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 ${\mathbf{M}_\Delta}^{\mathcal{G}}$.

B18:d.06 Introduciamo una MSPG $\mathbf{M}_C$ che si serve di una MSPG $\mathbf{M}_2$ e che chiamiamo **trasformazione per complementazione** della macchina data. Il nastro di uscita di tale macchina lo denotiamo con U_C.
Può accadere che si voglia il complementare in un dato monoide libero A^*, o in mancaza di tale richiesta si assume che A sia l'alfabeto che contiene tutte le stringhe di $\mathsf{L}_2 := {\mathbf{M}_2}^{\mathcal{G}}$.

Anche la MSPG $\mathbf{M}_C$ procede per fasi successive in ciascuna delle quali o chiede alla $\mathbf{M}_2$ di effettuare un passo evolutivo oppure procede alla generazione di un gruppo di stringhe di A^*.
Inizialmente il nastro U_C è vuoto e rimane tale fino a che viene generata una prima stringa w' di L_2. Con questa stringa vengono inserite su U_C tutte le stringhe di A^* aventi lunghezza minore o uguale di $|w'|$, ad esclusione della w' stessa.
In seguito quando viene generata una nuova stringa $w_2 \in \mathsf{L}_2$, se questa presenta una lunghezza non superiore a quella massima, che denotiamo con l_U, delle stringhe precedentemente collocate su U_C (che successivamente avrebbero anche potuto essere cancellate), allora viene cancellata da U_C; se invece w_2 presenta una lunghezza superiore alla attuale l_U, si procede aggiungere a questo nastro tutte le stringhe aventi lunghezza maggiore di l_U e minore o uguale ad $|w_2|$, ad esclusione della w_2 stessa.
Si constata allora che la $\mathbf{M}_C$ viene ad avere su U_C, prima o poi, tutte e sole le stringhe di A^* che non sono generate da $\mathbf{M}_2$.
L'insieme procedurale generato da $\mathbf{M}_C$, 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 :b.17: il ruolo di una macchina che genera L_1 è stato implicitamente assunto da una macchina che genera l'intero monoide liberoU.

Si osserva che la macchina sopra prospettata è in grado di generare liste non ripetitive le quali inoltre seguono un ordine algoritmico, un ordine len-lxg scelto per A^*. Va però osservato che ciascuna delle liste via via registrate su U potrebbero contenere elementi di L_2 non ancora generati, situazione che potrebbe rendere problematico decidere se una data stringa di A^* appartiene o meno ad ${\mathsf{L}_1}^{\complement}$.

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

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

B18:d.07 Introduciamo una MSPG $\mathbf{M}_\times$ che si serve di due date MSPG $\mathbf{M}_1$ e $\mathbf{M}_2$ e che chiamiamo **composizione per prodotto cartesiano** delle macchine date. Il nastro di emissione di questa macchina lo denotiamo con $U_\times$.

Anche la MSPG $\mathbf{M}_\times$ procede per fasi successive in ciascuna delle quali chiede alternatamente ad $\mathbf{M}_1$ e ad $\mathbf{M}_2$ di effettuare un passo evolutivo.
Supponiamo per semplicità che ciascuna delle $\mathbf{M}_i$ sul proprio nastro di uscita U_i generi una lista non ripetitiva e che ad ogni passo presenti una lista che segue un ordine sequenziale algoritmico. In tal caso si può fare in modo che la $\mathbf{M}_\times$ proceda a costruire una lista non ripetitiva e che segue un ordinamento sequenziale algoritmico.
In ogni fase nella quale viene generata una nuova stringa di $w_1 \in \mathsf{L}_1 = {\mathbf{M}_1}^{\mathcal{G}}$, viene aggiunta su $U_\times$ la sequenza delle coppie della forma $\langle w_1, w'' \rangle$ dove w'' corre sulla lista delle stringhe di L_2 generate fino a quel momento e registrate su U_2. Simmetricamente in ogni fase nella quale viene generata una nuova $w_2 \in \mathsf{L}_2 = {\mathbf{M}_2}^{\mathcal{G}}$ viene aggiunta ad $U_\times$ la sequenza delle coppie della forma $\langle w', w_2 \rangle$ dove w' corre sulla lista leggibile da U_1 delle stringhe di L_1 generate fino a quel momento.
Si osserva che dopo ogni fase sul nastro $U_\times$ si trova presentato un rettangolo di coppie che nella fase stessa è stato ampliato con coppie che costituiscono una sequenza aut orizzontale aut verticale di coppie. Con l'aggiunta di un opportuno dispositivo il rettangolo delle coppie si può mantenere in un ordine sequenziale algoritmico di struttura lessicografica.

B18:d.08 Le considerazioni dei precedenti paragrafi consentono di enunciare il seguente fatto.

(1) Teorema La collezione degli insiemi procedurali 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 ottenuti con MSPG note o con altre opportune notazioni e mediante coppie di parentesi coniugate si possono comporre espressioni insiemistiche ben formate che estendono quelle viste in B11:d.14 in grado di individuare insiemi finiti.

Osserviamo esplicitamente che ogni espressione ben formata $\mathcal{E}$ ottenuta con insiemi procedurali e con gli operatori insiemistici visti in precedenza (o con loro ragionevoli varianti) individua un insieme procedurale.
Questo fatto discende dalla considerazione che una espressione come la $\mathcal{E}$ individua una sequenza Σ di operazioni binarie o unarie che si servono di insiemi procedurali individuati nell'espressione iniziale e quindi riconducibili a ben definite MSPG, oppure di insiemi introdotti per individuare risultati di composizioni precedenti e quindi determinati da MSPG composite ottenute con le costruzioni indicate nei paragrafi precedenti. Alla fine della esecuzione delle operazioni della Σ, quindi, si riesce sempre ad ottenere una MSPG che individua un insieme procedurale risultato.

B18:d.09 A questo punto quindi possiamo disporre di un genere di espressioni in grado di individuare una ampia gamma di insiemi procedurali.
Vogliamo rilevare 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 determinati dgli operatori booleani che le caratterizzano, in quanto devono essere dotate di dispositivi che richiedono scorrimenti, confronti e riordinamenti che possono

essere piuttosto complessi. Tuttavia tutte queste composizioni si possono ricondurre ad operazioni elementari riguardanti stringhe, nastri, contatori e informazioni su oggetti discreti sostanzialmente semplici.

Osserviamo anche che abbiamo incontrato altre collezione di espressioni ciascuna delle quali, come quelle viste nelle pagine precedenti, si basa su un numero finito di oggetti che possono considerarsi elementari e su un numero finito di composizioni degli oggetti individuati dalle espressioni individuate in precedenza, composizioni utilizzabili per ampliare la collezione stessa. Tra queste espressioni segnaliamo le espressioni per stringhe e linguaggi, le espressioni su numeri interi e le espressioni per insiemi finiti.
Molte altre collezioni di espressioni con caratteristiche simili le incontreremo nel seguito: espressioni per polinomi, epressioni per 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 di calcoli della logica, ...
Queste collezioni di espressioni si riescono a trattare con strumenti della teoria dei linguaggi formali (v. in particolare C14:).
Con tali strumenti riusciremo ad esercitare vari controlli sulle espressioni dei vari generi ed a tenere sotto controllo le relative collezioni. Per molte di esse si riescono ad organizzare elaborazioni automatiche in modo da poterle trattare con stumenti di programmazione.

B18:e. Insiemi ricorsivi

B18:e.01 Una proprietà di grande rilievo che può essere goduta o meno dalle specifiche liste da MSPGI che abbiamo viste sta nel fatto che per ogni stringa w sull'alfabeto dei caratteri trattati si possa decidere in un numero finito di passi, ossia algoritmicamente, se compare o meno nella lista stessa. Tale proprietà la diremo **decidibilità del problema di occorrenza**.

Questa proprietà è goduta da ogni linguaggio L sopra un alfabeto finito A di uscita, cioè contenuto in un monoide A^*, linguaggio le cui stringhe vengono generate da una MSPGI $\mathbf{M}$ seguendo un ordine algoritmico Ω_K basato sopra un algoritmo di confronto K. In particolare il problema di occorrenza risulta decidibile da un linguaggio esprimente numeri interi generato seguendo l'ordine $<$ e da un linguaggio di stringhe sopra un dato alfabeto (finito) seguendo l'ordine len-lxg.
Infatti, data una qualsiasi stringa $w \in \mathsf{A}^*$, sul previsto alfabeto di uscita, procedendo alla generazione della lista, dopo un certo numero di passi, sicuramente finito, aut la w risulta generata, aut viene generata una stringa che segue la w secondo l'ordine Ω_K; a questo punto risulta accertato che la w non compare nella lista generata.

Passando a parlare in modo un poco più astratto di insiemi generati da procedure invece che di liste di stringhe generate da MSPG, siamo di fronte ad un problema che si può porre per tutte le procedure generative e che viene chiamato, **problema di appartenenza** o *membership problem*:

Consideriamo una procedura generativa $\mathbf{P}$ *il cui alfabeto di emissione denotiamo con* $\mathsf{A}_{\mathbf{P}}$. *Si chiede di stabilire per una qualsiasi stringa* w *su tale alfabeto se essa si può trovare come componente della lista* $\mathbf{P}^{\mathcal{G}}$ *o meno. Nel primo caso si può scrivere* $w \in \mathbf{P}^{\mathcal{L}}$, *nel secondo caso si può scrivere* $w \notin \mathbf{P}^{\mathcal{L}}$.

B18:e.02 La risoluzione di una istanza di problema di appartenenza si può vedere come la individuazione di una MSPA che denotiamo con $\mathbf{D}$ che appartenga al genere delle cosiddette **MSP dicotomiche** . Una

tale macchina ha il compito di eseguire le cosiddette procedure dicotomiche: essa è caratterizzata da avere un solo nastro di uscita U ridotto ad un unico registro binario su 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 un valore opposto a quello del precedente; Si presuppone che questi segni siano utilizzabili in scelte successive all'esecuzione della procedura dicotomica, le quali riguardino la scelta di una fra due attività successive. anche queste scelte sono qualificate come dicotomiche.

Una MSP dicotomica $\mathbf{D}$ finalizzata a risolvere un problema di appartenenza deve disporre di due nastri di ingresso, il primo dei quali in grado di accogliere una descrizione di una generica MSPGI $\mathbf{P}$, il secondo una generica $w \in \mathsf{A}_{\mathbf{P}}{}^{*}$. La procedura che decide l'appartenenza deve essere in grado di emettere sul registro di uscita U, dopo una elaborazione finita, il segno con significato positivo sse se riscontra l'appartenenza e il segno con significato opposto in caso contrario.

La richiesta di poter utilizzare con una MSP la descrizione di una generica MSP richiede qualche giustificazione. Delle MSP non abbiamo dato una definizione ben circostanziata, ma abbiamo richiesto, fiduciariamente, che fosse possibile dare per ciascuna di esse una descrizione completa e non ambigua dalla quale un opportuno EP fosse in grado di ottenere ciascuna delle sue evoluzioni.

A questo punto possiamo chiedere un po' più precisamente, che le MSP dei vari modelli possano essere definite esaurientemente con una stringa sopra un opportuno alfabeto A_M che soddisfi regole precise. Come esempi di queste stringhe definenti MSP incontreremo, in particolare, le descrizioni dei trasduttori a stati finiti (C13:), le descrizioni mediante grammatiche dei meccanismi generativi di linguaggi formali (C14:) e le definizioni di macchine di Turing (C21:). Ricordiamo inoltre le descrizioni di odierni programmi per computers scritte in un linguaggio di programmazione (C, Java, PHP, linguaggi per elaborazioni simboliche, ...), descrizioni che per un caso particolare abbiamo trattate in B17: .

Dobbiamo inoltre supporre che vengano definiti solo modelli $\mathcal{M}$ di MSP a ciascuno dei quali si possa associare una MSPA in grado di stabilire se una qualsiasi stringa sull'alfabeto $\mathsf{A}_{\mathcal{M}}$ usato per definire le specifiche MSP di $\mathcal{M}$ descrive esaurientemente e correttamente una macchina del modello stesso.
Si viene quindi a prospettare una MSPGI in grado di procedere alla generazione di tutte le descrizioni delle MSP di un certo modello $\mathcal{M}$ ben trattabile e quindi la disponibilità dell'insieme $\mathcal{M}$ attraverso le stringhe che costituiscono le descrizioni delle sue macchine specifiche.

B18:e.03 Prima di procedere è opportuno chiarire le nozioni di problema e delle sue istanze; si tratta di nozioni molto generali ma per ora ci limitiamo ai problemi di appartenenza. Una **istanza del problema** di appartenenza consiste nella decisione per una specifica MSPG $\mathbf{M}$ e una particolare stringa $w \in \mathsf{A}_{\mathbf{M}}{}^{*}$ se w si trova su $\mathbf{M}^{\mathcal{L}}$ 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 classe di procedure $\mathsf{MspG}\mathfrak{X}$ e per qualsiasi stringa sul corrispondente alfabeto di uscita, $\mathsf{A}_{\mathbf{P}}$.
Dunque un problema di appartenenza risulta formalmente ben definito quando sono dati un modello MspGX, o più operativamente una MSPA $\mathsf{M}_{\mathfrak{X}}$ in grado di decidere se una descrizione di MSP riguarda un membro di MspGX, ed un alfabeto A adatto ad esprimere le stringhe generate da membri di MspGX; tale problema si può individuare con la scrittura $\mathsf{PRBmember}(\mathsf{MspGX}, \mathsf{A})$.
Spesso l'alfabeto A non risulta essenziale, in quanto si possono avere formulazioni di problemi essenzialmente equivalenti con alfabeti diversi (ad esempio la gran parte dei problemi sui numeri interi non dipende dalla base delle notazioni posizionali usate per i numeri stessi). Inoltre l'alfabeto di uscita di una particolare MSPG si potrebbe ricavare dalla stringa che la definisce.

In genere alcune istanze di un problema di appartenenza sono facili da risolvere, talora banali. In

particolare sono tendenzialmente facili i problemi di appartenenza concernenti MPFG che generano insiemi finiti.
Al contrario le considerazioni sulle situazioni difficili da tenere sotto controllo per le composizioni di procedure MSPGI esposte in :b inducono ad aspettarsi che i problemi di appartenenza riguardante alcune classi di MSPGI possano presentare delle difficoltà.
A questo poposito conviene segnalare che, a causa della grande varietà delle prestazioni delle procedure, ovvero delle MSPGI, che risulta interessante prendere in considerazione, viene analizzata 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.

Se consideriamo una sottoclasse propria di MSPG $\mathbf{MspGi}\mathfrak{Y} \subset \mathbf{MspGi}\mathfrak{X}$ (queste classi di macchine o procedure si possono confondere con i linguaggi delle loro descrizioni e quindi si possono trattare come insiemi procedurali), con il relativo problema $\mathsf{PRBmember}(\mathbf{MspG}\mathfrak{Y})$ può essere chiamato **sottoproblema** del precedente $\mathsf{PRBmember}(\mathbf{MspG}\mathfrak{X})$.

Anche nello studio sistematico dei problemi di appartenenza è opportuno e naturale cercare di risolvere problemi di ampia portata. Si trova tuttavia che questa aspettativa incontra varie difficoltà ed anche limiti invalicabili. In effetti incontreremo problemi di appartenenza della forma $\mathsf{PRBmember}(\mathbf{MspG}\mathfrak{Z}\mathsf{A})$ per i quali si dimostra non esistere alcuna procedura dicotomica in grado di decidere l'appartenenza $w \in \mathbf{M}^{\mathcal{L}}$ per tutte le $w \in \mathsf{A}^*$.

B18:e.04 In generale chiamiamo **insieme numerabile** ogni insieme procedurale i cui elementi si possono porre in biiezione con una lista di stringhe sopra un determinato alfabeto generata da una MSPGI tale che il problema della occorrenza per tale lista risulti decidibile.

Primi esempi di insiemi numerabili sono evidentemente gli insiemi di base, entro i quali si collocano gli insiemi oggetto di uno studio; più specificamente citiamo l'insieme delle notazioni unadiche degli interi naturali e più in generale gli insiemi di tutte le stringhe sopra i vari alfabeti finiti A che risulta utile trattare, ovvero sopra i vari monoidi liberi A^*.
Per essi l'algoritmo che decide la occorrenza non fa che accettare ogni stringa generata.
Qui consideriamo solo insiemi numerabili che fanno parte di un monoide libero su un determinato alfabeto finito. In molte considerazioni si fa riferimento ai caratteri di uno solo di questi alfabeti e tali segni nel seguito saranno detti genericamente **caratteri in uso**.
Altri insiemi numerabili con finalità vicine agli insiemi di base sono quelli che abbiamo chiamato insiemi ambiente.
Tra questi segnaliamo gli insiemi delle notazioni posizionali degli interi naturali $\mathbb{N}_{[B]}$, insiemi ricavabile con una semplice riduzione di un monoide libero sopra l'insieme delle cifre $\{0, 1, ..., B-2\}$.
Altri insiemi ambiente si ottengono con composizioni insiemistiche di insiemi di base e di insiemi ambiente più semplici, con l'eventuale intervento di procedimenti selettivi algoritmici tendenzialmente semplici.

Inoltre sono insiemi numerabili tutti gli insiemi procedurali generati secondo un ordinamento sequenziale algoritmico. Questi li chiameremo **insiemi generabili illimitatamente e progressivamente** o, in breve, **insiemi generabili -ip**.
Si dice **insieme progressivamente generato** un insieme per il quale si può precisare una MSPG che la genera e una MSPA in grado di decide il problema del confronto delle sue stringhe.
Viceversa un insieme procedurale generato secondo un ordinamento sequenziale algoritmico è generabile progressivamente.

Ricordiamo infine che viene spesso usato il termine **insieme contabile** per caratterizzare un insieme finito o numerabile.

B18:e.05 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 sappia valutare in modo di ricavare un intero naturale che sia in corrispondenza con lo stesso n. In particolare si possono avere espressioni nelle quali possono comparire come operandi, oltre ad n, altri naturali specifici operandi forniti da scritture decimali o riducibili ad esse, e operatori come somma, prodotto, differenza, quoziente, resto, massimo, minimo, MCD, mcm, ed altre funzioni definite in modo che sia garantito che possano effettivamente fornire interi naturali.
Alcune di queste funzioni vengono definite in queste pagine: tra di esse le funzioni $SB(n)$ (:b.01) e $\mathbb{N}_{[B]}$ per ogni $B = 2, 3, 4...$, la funzione totient di Eulero definita in B26:d.03 e le varie successioni introdotte in D20: (coefficienti binomiali, successione di Fibonacci $\mathsf{Fib}(n)$, di Catalan $Ctln_n$, di Lah $nlah(n,k)$, di Lucas $Luc(n)$, ...) .
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 da appositi algoritmi chiamati algoritmi di [[parsing]] i quali riescono a garantire che essa sia effettivamente calcolabile.

Esempi di presentazioni di queste successioni sono

$$\langle n \in \mathbb{N} :| \; 5 + 3\,n^2 + n^3 \rangle \quad , \quad \langle n \in \mathbb{N} :| \; n! + \min(8\,n, nlah(2\,n, n)\,7\,SB(3\,n)) + n\%3 \rangle \; .$$
$$\langle n \in \mathbb{N} :| \; \max(Fib_{2\,n}, 7\,\mathbb{N}_{[2]} + 2\,n\%3 \rangle \; .$$

Più in generale si possono presentare espressioni della forma $\langle n \in \mathbb{N} :| \; \mathsf{T}(n) \rangle$, nella quale T denota una una MSPA con compiti di trasduttore da $\mathbb{N}$ in $\mathbb{N}$, cioè una MSPA che costituisca una rappresentazione algoritmica di una funzione calcolabile di $\{\mathbb{N} \longmapsto \mathbb{N}\}$. Come si è accennato ciascuna delle espressioni nelle presentazioni precedenti conduce ad una MSPA, ma la formulazione attuale è più comprensiva, in quanto le caratteristiche delle MSPA sono aperte; in altre parole di tali macchine si è data una definizione fiduciaria che può comprendere una vasta gamma di strumenti.

Altre importanti successioni di interi naturali si possono definire mediante sistemi di uguaglianze di ricorrenza come quelli con i quali sono stati introdotti i numeri di Fibonacci (B17:a.11) e i numeri di Catalan (:c.04). Molte di queste presentazioni non si sanno ricondurre alle presentazioni della prima forma, mentre si chiede che siano tali da essere riconducibili a quelle della seconda forma.

Va inoltre segnalato fin d'ora che potremo considerare espressioni riguardanti altri insiemi numerabili di numeri, come i numeri interi relativi (B20:a), 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, si possono introdurre varie operazioni e numerose funzioni speciali. Per le successioni è quindi disponibile un'ampia varietà di espressioni e di procedure che consentono molteplici calcoli e verifiche di proprietà di vario genere. occorre anche avvertire che molti calcoli sopra le successioni numeriche non può essere che calcoli approssimati (B38:).

B18:e.06 Tra gli insiemi numerabili vanno considerati anche i linguaggi numerabili utilizzati per definire le MSP dei vari modelli.
Più in generale ogni genere di struttura matematica introdotta con atteggiamenti à costruttivi deve essere definita in modo preciso e quindi anche per le strutture di un dato genere dovrebbe potersi individuare una MSPA in grado di decidere se una stringa sull'alfabeto adottato descrive una MSP o meno

e quindi una MSPGI in grado di individuare l'insieme numerabile delle strutture del corrispondente genere.

Ad esempio non è difficile individuare un linguaggio per la presentazione dei singoli gruppi finiti (B41:b, 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 ed anche per i gruppi finiti semplici (un linguaggio con tale possibilità deve esprimere una grande varietà di oggetti).
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, ...

B18:e.07 Ci proponiamo ora di dimostrare costruttivamente che alcune composizioni di due insiemi numerabili N_1 e N_2 forniscono altrettanti insiemi numerabili. Per ogni composizione (unione, intersezione, eliminazione, ...) individueremo una MSPGI in grado di generare una lista degli elementi dell'insieme individuato servendoci della MSPGI $\mathbf{M}_1$ che genera N_1 e della MSPGI $\mathbf{M}_2$ che genera N_2. Denotiamo inoltre con $\mathbf{D}_1$ e con $\mathbf{D}_2$ le macchine in grado di risolvere, risp., il problema della appartenenza per $\mathbf{M}_1$ e $\mathbf{M}_2$. Può anche essere interessante supporre che N_1 ed N_2 vengano generati seguendo due ordini sequenziali algoritmici che contrassegnamo con i quattro segni da usare come infissi $\preceq_i$, $\prec_i$, $succeq_i$ e $\succ_i$ per $i = 1, 2$.

(1) Prop.: L'unione $\mathsf{N}_1 \cup \mathsf{N}_2$ 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 $w \in \mathsf{A}^{\mathcal{G}\mathrm{en}}$ appartiene o meno ad $NSs_1 \cup \mathsf{N}_2$.
Per questa $\mathbf{D}$, come per le MSPA decisionali che vedremo per le altre composizioni di insiemi ricorsivi, è lecito chiedere che si tratti di una macchina in grado di simulare sia la $\mathbf{M}_1$ che la $\mathbf{M}_2$.
La decisione sull'appartenenza della generica w ad N si ottiene in una o due fasi. Nella prima si simula Dbf_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 Dbf_2 per stabilire se $w \in \mathsf{N}_2$ e dopo un numero finito di passi si decide come verrebbe deciso dalla $\mathbf{M}_2$ ∎

B18:e.08 Osserviamo che anche $\mathsf{N}_1 \cup \mathsf{N}_2$ viene generato secondo l'ordine $\preceq$.
Nella pratica potrebbe porsi il problema di disporre di tutti gli elementi di $\mathsf{N}_1 \cup \mathsf{N}_2$ inferiori o uguali ad un dato intero positivo $\overline{n}$; chiaramente è garantito che per ogni $\overline{n}$ si riesce ad ottenere questo elenco finito mediante una sequenza finita di passi in quanto è possibile controllare quando sia $\mathbf{M}_1$ che $\mathbf{M}_2$ hanno esaurito la generazione delle stringhe che precedono o uguagliano $\overline{n}$. Questa possibilità sostanzialmente equivale alla possibilità di decidere per ogni stringa sull'alfabeto in uso se appartiene o meno a $\mathsf{N}_1 \cup \mathsf{N}_2$.

Può essere utile considerare come esempio le seguenti fotografie del nastro di uscita per la costruzione dell'unione degli insiemi (numerabili) dei numeri primi e dei multupli positivi di 6:
2 3 5 7 , 2 3 5 6 7 , 2 3 5 6 7 12 , 2 3 5 6 7 11 13 , 2 3 5 6 7 11 13 18 ,
2 3 5 6 7 11 13 17 18 19 23 , ... 18 19 23 29 , ... 18 19 23 24 29 30 36 , ...

Accanto alla proposizione :e.07(1) conviene considerare le seguenti come sue generalizzazioni.

(1) Prop.: L'unione di una collezione finita di insiemi numerabili è un insieme numerabile.

Dim.: Per decidere l'appartenenza si fa uso di una macchina in grado di simulare tutte le macchine della collezione finita, il cui numero denotiamo con m e che denotiamo con D_1, D_2, ... ,D_m. Questa possibilità non presenta difficoltà: si potrebbe ottenere ripetendo $m - 1$ volte le considerazioni sopra

la possibilità di simulare una macchina e ricorrendo alla transitività della simulazione. Con questa macchina decisionale e la generica stringa w si simulano una dopo l'altra le macchine decisionali parziali fino a che si trova una di queste che accetta la w in modo da concludere che $w \in \mathsf{N}$. Nel caso che tutte le macchine rifiutano w e solo in questo caso si decide che $w \notin \mathsf{N}$ ∎

(2) Coroll.: L'unione disgiunta di due insiemi numerabili è un insieme numerabile.

B18:e.09 Prop. L'intersezione $\mathsf{N} := \mathsf{N}_1 \cap \mathsf{N}_2$ è un insieme 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 $w \in \mathsf{A}^{\mathcal{G}\mathrm{en}}$ appartiene sia ad NSs_1 che a N_2.
Anche per questa $\mathbf{D}$s si chiede che si tratti di una macchina in grado di simulare sia la $\mathbf{M}_1$ che la $\mathbf{M}_2$.
La decisione sull'appartenenza della generica w ad N si ottiene in una o due fasi. Nella prima si simula Dbf_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 simulare Dbf_2 per stabilire se $w \in \mathsf{N}_2$ e dopo un numero finito di passi si decide come verrebbe deciso dalla $\mathbf{M}_2$ ∎

Se uno dei due insiemi, chiamiamolo N_1, è finito, evidentemente anche l'intersezione deve essere un insieme finito.
In termini operativi occorre dire che solo se ad 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 $\mathbf{M}$. Come sia possibile stabilire l'esaurimento delle emissioni utili della N_1 dipende dalle particolarità della N_1 e dall'andamento dell'esecuzioni.

B18:e.10 (1) Prop.: L'eliminazione $\mathsf{N} := \mathsf{N}_1 \setminus \mathsf{N}_2$ è un insieme 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 $w \in \mathsf{A}^{\mathcal{G}\mathrm{en}}$ appartiene ad NSs_1 e non fa parte di N_2.
Anche per questa $\mathbf{D}$s si chiede che si tratti di una macchina in grado di simulare sia la $\mathbf{M}_1$ che la $\mathbf{M}_2$.
La decisione sull'appartenenza della generica w ad N si ottiene in una o due fasi. Nella prima si simula Dbf_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 simulare Dbf_2 per stabilire se $w \in \mathsf{N}_2$ e dopo un numero finito di passi si assume la decisione opposta a quella decisa dalla $\mathbf{M}_2$; se w appartiene anche a N_2 viene rifiutata per N, se w non fa parte di N_2 si decide di accettarla in N ∎

(2) Prop.: Il complementare ${\mathsf{N}_2}^{\complement}$ è un insieme 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 $w \in \mathsf{A}^{\mathcal{G}\mathrm{en}}$ non fa parte di N_2.
Per questa $\mathbf{D}$s si chiede che sia in grado di simulare la $\mathbf{M}_2$.
La decisione sull'appartenenza della generica w ad N si ottiene simulando Dbf_2 per stabilire se $w \in \mathsf{N}_2$.
In caso di esito negativo si decide di accettare w in N; se all'opposto si trova che w appartiene a N_2, la stringa viene rifiutata per N ∎

(3) Prop.: La differenza simmetrica $\mathsf{N}_1 \Delta \mathsf{N}_2$ è un insieme 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 $w \in \mathsf{A}^{\mathcal{G}\mathrm{en}}$ appartiene ad uno solo dei linguaggi NSs_1 ed N_2.
Anche per questa $\mathbf{D}$s si chiede che si tratti di una macchina in grado di simulare sia la $\mathbf{M}_1$ che la $\mathbf{M}_2$.

La decisione sull'appartenenza della generica w ad N si ottiene in due fasi. Nella prima si simula Dbf_1 per stabilire se $w \in \mathsf{N}_1$ e pe entrambi gli esiti si procede a simula Dbf_2 per stabilire se $w \in \mathsf{N}_2$.
Si decide poi che $w \in \mathsf{N}$ sse si è trovato che w appartiene ad 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 apartiene a nessuno dei due linguaggi ∎

B18:e.11 **(1) Prop.**: Il prodotto cartesiano di due insiemi numerabili è un insieme numerabile.

Si tratta di descrivere una MSPA $\mathbf{D}$ in grado di decidere in un numero finito di passi se una qualsiasi coppia $\langle w_1, w_2\rangle$ appartenente 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 simulare sia la $\mathbf{M}_1$ che la $\mathbf{M}_2$.
La decisione sull'appartenenza della generica w ad N si ottiene in due fasi. Nella prima si simula Dbf_1 per stabilire se $w \in \mathsf{N}_1$ e per entrambi gli esiti si procede a simulare Dbf_2 per stabilire se $w \in \mathsf{N}_2$.
Si decide poi che $\langle w_1, w_2\rangle \in \mathsf{N}$ sse si è trovato che $w_1 \in \mathsf{N}_1$ e che $w_2 \in \mathsf{N}_2$ ∎

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

(2) Prop.: **(1) Prop.**: Il prodotto cartesiano di una sequenza finita di insiemi numerabili è un insieme numerabile ∎

B18:e.12 Come abbiamo già osservato, per trattare con efficienza espositiva dati da elaborare o prodotti di elaborazioni, conviene individuare degli insiemi di tali oggetti e contraddistinguerli con notazioni opportune.
Per questi identificatori si possono utilizzare convenientemente i segni di operazioni su insiemi come intersezione, unione, eliminazione, differenza simmetrica, complementazione, 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.

Altre sequenze interessanti si ottengono con procedimenti che procedono in fasi nelle quali si costruisce una nuova componente a partire da una delle precedenti, oppure da sue delle precedenti, oppure ... , oppure da tutte le precedenti.

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 come quelle che conducono ai numeri negativi, ai razionali e ai coefficienti binomiali (v. B13:e.13). Incontreremo 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:).
A proposito di queste strutture grafiche segnaliamo che etichettando secondo opportuni criteri i loro nodi terminali con operatori ed operandi (v. 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 (v. ad esempio C14:e-g).

B18:e.13 Del linguaggio N emesso da una MSPG $\mathbf{M}$ che genera un linguaggio ricorsivo sull'alfabeto che denotiamo con A, si può dire che, almeno in linea di principio, può essere tenuto sotto controllo efficacemente; più precisamente per una stringa qualsiasi $w \in \mathsf{A}^*$ si può dire se compare o meno sul nastro di uscita ed in caso affermativo la corrispondente macchina che decide l'appartenenza associata alla $\mathbf{M}$ rende lecito attendere con piena fiducia la emissione della w e l'attesa 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 (la prima 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 fra stringhe è evidentemente una relazione d'ordine: infatti essa si può considerare una relazione riflessiva ed è sicuramente antisimmetrica e transitiva; inoltre si tratta di un 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'ordine len-lxg le stringhe sulle tre lettere `a`, `b` e `c` riferendosi a due diversi loro ordinamenti, ad esempio ad ($\mathtt{a} \prec \mathtt{b} \prec \mathtt{c}$) e ad ($\mathtt{b} \prec \mathtt{c} \prec \mathtt{a}$).

(1) Prop.: Per un insieme di stringhe ordinate sequenzialmente 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 $w \prec v$.

Algor.: Si costruiscono due liste, la prima che inizia con v, la seconda con w; si procede quindi a successive fasi nele 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 $w \prec v$ ∎

Si possono individuare molte interessanti generazioni progressive.
Denoteremo con **MspGip** l'insieme delle MSP che generano ilimitatamente e progressivamente.

B18:f. Relazioni e funzioni ricorsive

B18:f.01 Si incontrano molte MSPGip che generano liste di coppie di grande interesse. Questi insiemi di coppie sono sia relazioni procedurali (:b.05) che insiemi ricorsivi e quindi sono chiamati **relazioni [binarie] ricorsive**. Segnaliamo alcune macchine piuttosto semplici che generano relazioni ricorsive.

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

(2) Macchina che emette le coppie $\langle 0, 1\rangle$, $\langle 1, 2\rangle$, $\langle 2, 3\rangle$, $\langle 3, 5\rangle$, ..., $\langle n, p_n\rangle$, ... cioè le coppie costituite dal numero naturale $n \in \mathbb{N}$ e dall'n-esimo numero primo.

(3) Macchina che genera le 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 un intero positivo e da un suo divisore proprio.

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

le prime tre precedenti sono relazioni fra interi, la quarta è una relazione fre stringhe di $\{\mathtt{a}, \mathtt{b}\}^*$.

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

B18:f.02 Riprendendo le considerazioni in :b.05, si può affermare che il dominio e il codominio di una relazione ricorsiva sono insiemi ricorsivi
È inoltre evidente che prodotti cartesiani di insiemi ricorsivi come $\mathbb{N} \times \mathbb{N}$, $\mathbb{N} \times \mathsf{A}^*$ e $\mathsf{A}^* \times \mathsf{A}^*$ si possono considerare relazioni ricorsive.

Le particolari le relazioni introdotte in :d.01 sono evidentemente sottoinsiemi infiniti degli ambienti ricorsivi $\mathbb{N} \times \mathbb{N}$ e $\{\mathtt{a}, \mathtt{b}\}^* \times \{\mathtt{a}, \mathtt{b}\}^*$.

Iu generale si possono introdurre relazioni binarie ricorsive che sono sottoinsiemi di prodotti cartesiani della forma $\mathsf{U} \times \mathsf{U}$ o della forma $\mathsf{U} \times \mathsf{V}$, dove con U e V denotiamo ambienti ricorsivi.
Varie relazioni ricorsive si possono ottenere con MSPGip che scorrono secondo un ordine algoritmico un ambiente della forma $\mathsf{U} \times \mathsf{V}$ ed a ciascuna delle sue coppie applicano una MSPA $\mathbf{S}$ con il compito di selettore, ossia di accettatore, ed emettono le coppie che esso accetta.
Esempi di relazioni ricorsive di questo genere sono, per ogni $k \in \mathbb{P}$, gli insiemi

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

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, simmetriche, antisimmetriche, transitive, di equivalenza, di ordine, di ordine totale, di natura reticolare, ...
Inoltre tra queste relazioni si possono stabilire delle relazioni: ad esempio può interessare che una relazione R_1 sia meno estesa di una seconda R_2 in tale caso si può anche dire che la R_1 implica la R_2.
Alle relazioni ricorsive, evidentemente, si possono applicare composizioni di natura insiemistica come unione, intersezione e complementazione che, in conseguenza di quanto dimostrato in :c, portano ad altre relazioni ricorsive.
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 $\mathsf{R}, \mathsf{S} \in \mathsf{U} \times \mathsf{U}$ e due MSPGip $\mathbf{M}$ ed $\mathbf{N}$ tali che $\mathsf{R} = \mathbf{M}^{\mathcal{G}}$ e $\mathsf{S} = \mathbf{N}^{\mathcal{G}}$. Si tratta di individuare una MSPGip $\mathbf{P}$ che genera $\mathsf{R} \circ \mathsf{S}$. Essa primariamente organizza una corsa sulle coppie $\langle u, w\rangle$ di $\mathsf{U} \times \mathsf{U}$ secondo un ordine algoritmico $\preceq$ per stabilire per ciascuna delle coppie $\langle x, y\rangle \preceq \langle u, w\rangle$ 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 $\langle u, w\rangle$. Questi fatti si possono decidere in un numero finito di passi in quanto riguardano un insieme finito di coppie di $\mathsf{U} \times \mathsf{U}$ non successive alla $\langle u, w\rangle$.
Dopo la fase relativa ad una coppia $\langle u, w\rangle$ sul nastro di uscita U della $\mathbf{M}$ sono elencate le coppie per le quali le coppie precedenti o la uguale garantiscono l'appartenenza ad $\mathsf{R} \circ \mathsf{S}$. Alla fine della fase relativa alla coppia successiva la lista su 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 ogni coppia della $\mathsf{R} \circ \mathsf{S}$ viene individuata ∎

(2) Prop.: La chiusura transitiva e di una relazione ricorsiva 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 successive secondo un ordine $\preceq$ coppie $\langle u, w\rangle$ ed 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 ad un insieme finito di coppie. Anche con questo processo ogni coppia di $\mathsf{R}^{\oplus}$ viene individuata ∎

(3) Prop.: La chiusura di equivalenza di una relazione ricorsiva 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 R^{eqv}. Similmente alla macchina individuata per la dimostrazione precedente $\mathbf{T}$ organizza una

successione di fasi associate alle successive secondo un ordine $\preceq$ coppie $\langle u, w\rangle$ ed in ciascuna fase considera l'insieme $K_{\langle u,w\rangle}$ delle $\langle x, y\rangle \preceq \langle u, w\rangle$ 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 ad un insieme finito di coppie. Anche con questo processo ogni coppia di R^{eqv} viene individuata ∎

B18:f.05 Le più importanti di queste proprietà, composizioni e relazioni si possono esprimere in termini che prescindono dalle modalità secondo le quali le relazioni specifiche sono state costruite o definite. Esse quindi, come vedremo, possono essere estese a relazioni più generali di quelle costruibili con macchine sequenziali o procedure.

Come già segnalato per le relazioni finite, sono particolarmente importanti le equivalenze e le relazioni d'ordine.
Per individuarle conviene definire varie collezioni associate a proprietà. Diciamo che una relazione R avente come dominio D e codominio $C = D$ è:
riflessiva sse $\forall x \in D : x\mathsf{R}x$;
antiriflessiva sse $\forall x \in D : x \not\mathsf{R}x$;
simmetrica sse $x\mathsf{R}y \Longrightarrow y\mathsf{R}x$;
antisimmetrica sse $D = C$ e $x\mathsf{R}y \Longrightarrow y \not\mathsf{R}x$;
transitiva sse $x\mathsf{R}y \ \wedge \ y\mathsf{R}z \Longrightarrow x\mathsf{R}z$.

Una relazione procedurale si dice **equivalenza procedurale** sse è riflessiva, simmetrica e transitiva.
Una relazione procedurale si dice **relazione d'ordine procedurale** o **ordinamento procedurale** sse 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 una certa generalità consideriamo una relazione procedurale $\mathsf{R} \subseteq \mathsf{U} \times \mathsf{V}$.
Si dice **dominio** della R l'insieme dom(R) dei primi membri delle coppie che costituiscono la relazione; si dice **codominio** della R l'insieme cod(R) dei secondi membri delle coppie che costituiscono la relazione.
Si distinguono le relazioni funzionali ossia le funzioni procedurali da U in U relazioni R tali che se $\langle a, b\rangle, \langle a, b'\rangle \in \mathsf{R}$, allora deve essere $b = b'$. Per una tale relazione in genere risulta utile individuare un meccanismo che ad ogni elemento $a \in$ dom(R) associa l'elemento $b \in$ cod(R) tale che $\langle a, b\rangle \in \mathsf{R}$.
Tra le funzioni procedurali $f \in \{\mathsf{U} \longrightarrow \mathsf{U}\}$ si distinguono le suriettive, tali che cod$(f) = \mathsf{U}$, le iniettive, tali che ogni elemento del codominio può essere l'immagine di un solo elemento del dominio, e le biiettive o **funzioni invertibili**. Semplicissima funzione invertibile è l'identità su U, IdU, definita come l'insieme $\{x \in \mathsf{U} :| \langle x, x\rangle\}$.
Le relazioni, in quanto insiemi (di coppie), possono essere composte con operazioni insiemistiche come unione ed intersezione.
La complementazione porta da una relazione alla sua negazione.
Le relazioni possono essere composte con il cosiddetto prodotto di composizione. Una biiezione composta con la propria inversa fornisce l'identità sul suo dominio.

B18:f.07 Tra le funzioni ricorsive rivestono particolare interesse quelle di $\mathbb{N}$ in $\mathbb{N}$; queste sono dette **successioni di numeri naturali**.
Esse si possono individuare con le sequenze illimitate dei valori generati da una MSPGip che le individua.

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

Sono utili in particolare quelle per le quali le componenti generiche a_n possono essere individuate mediante espressioni significative. Inoltre sono molto vicine alle funzioni di $\mathbb{P}$ in $\mathbb{N}$ oppure in $\mathbb{P}$: in effetti a ciascuna $\mathsf{a} \in \{\mathbb{N} \longmapsto \mathbb{N}\}$ si fa corrispondere biunivocamente la funzione

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

La scelta fra questi generi di funzioni dipende da vantaggi pratici che si possono avere con una delle due formalizzazioni.

Anticipiamo 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** ponendo $F_0 := 1$, $F_1 := 1$ e calcolando i successivi $F_n := F_{n-1} + F_{n-2}$ per $n = 2, 3, ...$;

la successione dei **numeri di Catalan** definita dalle richieste

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

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 Si possono anche definire insiemi ambiente della forma $\mathsf{U}_1 \times \mathsf{U}_2 \times ... \times \mathsf{U}_d$ dove $d = 3, 4, ...$ e 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. In essi in particolare si collocano le d-uple delle potenze dei caratteri associate alle stringhe sopra un alfabeto di d caratteri.

Negli ambienti sopra accennati si collocano le espressioni che definiscono formalmente le strutture 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.

Possiamo quindi definire gli insiemi di MSP, di MSPG, di MSPGI e di altre specie di macchine come insiemi ricorsivi.

Per questi insiemi usiamo, risp., le notazioni **Msp**, **MspG**, **MspGi** e **MspGf**.

Conviene rilevare che molte entità descritte, ed in particolare molti insiemi di macchine sequenziali programmabili, non sono state date definizioni precise e complete. In effetti per ora abbiamo proceduto con atteggiamento fiduciario: non definiamo precisamente **Msp** e non affrontiamo il problema di decidere se una MSPG sia in **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 elevato interesse (in particolare applicativo). per .

B18:g. Primi problemi critici sugli insiemi procedurali

B18:g.01 Ci proponiamo ora un primo chiarimento di una problematica importante, 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 EP, umano o automatico, incaricato di fornire informazioni sulla evoluzione di una MSPG $\mathbf{M}$.
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 eventualmente adattamenti dele procedure).

(1) Si è dimostrato che $\mathbf{M}$ genera una lista limitata di stringhe, cioè che $\mathbf{M} \in$ **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 $\mathbf{M}^{\mathcal{G}}$ 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 $\mathbf{M}^{\mathcal{L}}$ e si ha la consapevolezza della incompletezza dei risultati, cioè si sa che l'impiego di altre risorse consentirà la generazione delle altre stringhe. Si pone quindi la scelta (1bα) aut proseguire nella speranza di ottenere altre stringhe di $\mathbf{M}^{\mathcal{L}} \setminus E$ a un costo contenuto (1bβ) aut interrompere l'evoluzione per evitare di impiegare altre risorse per ottenere poco o nulla di nuovo.
   (1c) Con le risorse impiegate si è ottenuto sul nastro di uscita un elenco E e non si sa se impiegando altre risorse possano essere generate altre stringhe, cioè non si sa se $L = \mathbf{P}^{\mathcal{L}}$ oppure $L \subset \mathbf{P}^{\mathcal{L}}$.

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

(3) Non si è riusciti a decidere se l'elenco che la MSPG va generando può estendersi illimitatamente o meno; con le risorse impiegate fino al passo n si dispone di un elenco di stringhe generate E che potrebbe esaurire $\mathbf{M}^{\mathcal{L}}$ o all'opposto rivelarsi parziale, in quanto impiegando nuove risorse diventa possibile generare altre stringhe che appartengono ad $\mathbf{M}^{\mathcal{L}} \setminus E$. 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 $\mathbf{M}^{\mathcal{L}} \setminus E$ sia di sprecare risorse per ottenere poco o nulla. In alternativa si può scegliere di studiare più a fondo la $\mathbf{M}$ sperando di ottenere nuove informazioni sopra i suoi comportamenti , oppure di studiare il problema che ha condotto alla $\mathbf{M}$, sperando di individuare una macchina migliore per la sua soluzione.