LABORATORIO DI PROGRAMMAZIONE

Corso di laurea in matematica | Anno accademico 2004/05 | Numero 1

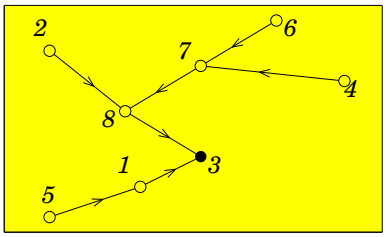
Sistemi dinamici finiti

X sia un insieme finito e

$$f : X \longrightarrow X$$

un'applicazione. Allora (X, f) si chiama un sistema dinamico finito. Questi sistemi sono attualmente molto studiati (spesso sotto il nome di automi cellulari) e vengono usati per rappresentare processi fondamentali in molte campi dell'informatica e della scienza, dalla medicina alla geofisica.

Un sistema dinamico finito può essere rappresentato graficamente come nel seguente esempio, in cui $X = \{1, \dots, 8\}$, mentre l'applicazione $f : X \longrightarrow X$ è data dalle frecce nel grafo. Il cerchietto pieno indica un punto fisso (in questo caso il numero 3) che viene mandato in se stesso da f:



Situazione 1.1. *(X, f) sia un sistema dinamico finito e, quando non indicato diversamente, $x, y, a, b, \dots \in X$ ed $A, B \subset X$.*

Orbite

Definizione 1.2. Un sottoinsieme $A \subset X$ si dice *invariante*, se $f(A) \subset A$, cioè se $f(a) \in A$ per ogni $a \in A$. In tal caso possiamo definire un'applicazione

$$\begin{aligned} f_A : A &\longrightarrow A \\ a &\longmapsto f(a) \end{aligned}$$

La coppia (A, f_A) è ancora un sistema dinamico finito.

È chiaro anche che $f^n(A) \subset A$ per ogni $n \in \mathbb{N}$.

Definizione 1.3. Invece di $f^n(x)$ scriveremo anche $x + n$. Nel grafico del sistema $x + n$ è il punto (univocamente determinato) che si ottiene seguendo le frecce per n passi a partire da x.

Useremo questa notazione (che ricorda un po' l'aritmetica dei puntatori in C) soprattutto nelle considerazioni astratte; è una notazione molto comoda e intuitiva che naturalmente crea ambiguità quando gli elementi di X, come spesso nei nostri esempi, sono dei numeri. Sono immediate le seguenti relazioni che esprimono il fatto che $\mathbb{N}$ opera su X mediante l'applicazione $(x, n) \longmapsto f^n(x)$:

(1) $x + 0 = x$.

(2) $(x + n) + m = x + (n + m)$ per ogni $n, m \in \mathbb{N}$. Possiamo quindi tralasciare le parentesi e scrivere semplicemente $x + n + m$.

Definizione 1.4. L'insieme (finito)

$$S(x) := \{x + n \mid n \in \mathbb{N}\}$$

si chiama l'*orbita* di x. Gli elementi di $S(x)$ sono detti *successori* di x. Notiamo che $x \in S(x)$, perché $x = x + 0$.

È chiaro che $S(x)$ è un sottoinsieme invariante non vuoto di X.

Un sottoinsieme di X si chiama un'*orbita*, se coincide con l'orbita di qualche punto di X.

Lemma 1.5. *Siano $a, b \in S(x)$. Allora $a \in S(b)$ oppure $b \in S(a)$.*

Dimostrazione. Per ipotesi $a = x + n$ e $b = x + m$ con $n, m \in \mathbb{N}$. Sia ad esempio $m \geq n$. Allora

$$b = x + n + (m - n) = a + (m - n) \in S(a)$$

Osservazione 1.6. *Se b è un successore di a e c un successore di b, allora c è un successore di a. In altre parole:*

$$b \in S(a) \Longrightarrow S(b) \subset S(a)$$

Dimostrazione. Se l'ipotesi è soddisfatta, abbiamo $b = a + n$ e $c = b + m$ per qualche $n, m \in \mathbb{N}$.

Ciò implica $c = a + m + n \in S(a)$.

Definizione 1.7. Introduciamo una relazione $\leq$ su X ponendo

$$a \leq b : \Longleftrightarrow \ b \in S(a)$$

Dall'osservazione 1.6 segue che la relazione è transitiva; siccome sempre $a \in S(a)$, vediamo che $\leq$ è un quasiordine su X. In genere non si tratta di un ordine parziale perché può accadere che $a \leq b \leq a$ con $a \neq b$. Vedremo che ciò avviene se e solo se a e b appartengono allo stesso insieme minimale.

Osservazione 1.8. $a \leq b \Leftrightarrow S(b) \subset S(a)$.

Dimostrazione. Sia $a \leq b$, cioè $b \in S(a)$. Per l'osservazione 1.6 allora $S(b) \subset S(a)$.

Sia invece $S(b) \subset S(a)$. Però $b \in S(b)$, quindi $b \in S(a)$.

Osservazione 1.9. *Sono equivalenti:*

(1) A è invariante.

(2) $a \in A \Longrightarrow S(a) \subset A$.

In questo numero

Catene

Definizione 1.10. Un sottoinsieme A di X si dice una *catena*, se $A \neq \emptyset$ e se per due suoi elementi a e b si ha sempre $a \leq b$ oppure $b \leq a$. Ricordiamo che il nostro quasiordine in genere non è antisimmetrico e quindi in genere non è un ordine parziale.

Proposizione 1.11. *$S(x)$ è una catena.*

Dimostrazione. Questo è esattamente l'enunciato del lemma 1.5.

Lemma 1.12. *A sia una catena. Allora esiste un $a \in A$ tale che $A \subset S(a)$.*

Dimostrazione. Dimostriamo l'enunciato per induzione su $|A|$.

Sia $|A| = 1$. Allora $A = \{a\}$ per un singolo elemento a e si ha $A = \{a\} \subset S(a)$.

Sia $A = B \cup x$ una catena con $x \notin B$. Allora anche B è una catena e per ipotesi di induzione esiste $b \in B$ tale che $B \subset S(b)$. Se $x \in S(b)$, allora $A \subset S(b)$. Altrimenti, siccome x e b appartengono alla catena A, deve valere $b \in S(x)$. Ma allora, per l'osservazione 1.6, $B \subset S(b) \subset S(x)$ e quindi anche $A \subset S(x)$.

Proposizione 1.13. *Per un sottoinsieme $A \neq \emptyset$ di X sono equivalenti:*

(1) A è una catena.

(2) A è contenuto in un'orbita.

Dimostrazione. (1) $\Longrightarrow$ (2): Lemma 1.12.

(2) $\Longrightarrow$ (1): Segue direttamente dall'osservazione 1.11, perché chiaramente ogni sottoinsieme non vuoto di una catena è una catena.

Proposizione 1.14. *Per un sottoinsieme $A \neq \emptyset$ di X sono equivalenti:*

(1) A è una catena invariante.

(2) A è un'orbita.

(3) A è invariante e contenuto in un'orbita.

Dimostrazione. (1) $\Longrightarrow$ (2): Per il lemma 1.12 esiste $a \in A$ tale che $A \subset S(a)$. Ma A è invariante, perciò $S(a) \subset A$, e ciò implica $A = S(a)$.

(2) $\Longrightarrow$ (3): Chiaro.

(3) $\Longrightarrow$ (1): Proposizione 1.13.

La proposizione 1.11 esprime una condizione di direzionalità delle ramificazioni tipica di strutture ad albero e implica, come vedremo nel prossimo numero, che ogni sistema dinamico finito è costituito dall'unione disgiunta di alberi che, come fiumi in un complesso di laghi, sfociano nei cicli del sistema.

Insiemi invarianti minimali

Definizione 2.1. Un sottoinsieme invariante $A \subset X$ si dice *invariante minimale*, se $A \neq \emptyset$ e A è minimale tra i sottoinsiemi invarianti non vuoti di X. Ciò significa che, se B è un sottoinsieme invariante di X con $B \subset A$, allora $B = \emptyset$ oppure $B = A$.

Proposizione 2.2. *Un sottoinsieme non vuoto di X è invariante minimale se e solo se coincide con l'orbita di ogni suo punto.*

Dimostrazione. (1) A sia un sottoinsieme invariante minimale di X ed $a \in A$. Allora $S(a)$ è un sottoinsieme invariante $\neq \emptyset$ di A e coincide quindi con A.

(2) A sia un sottoinsieme non vuoto di X che coincide con l'orbita di ogni suo punto. Allora in particolare A è un orbita e quindi invariante.

Sia B un sottoinsieme invariante $\neq \emptyset$ di A. Scegliamo un punto b qualsiasi di B. Per ipotesi $A = S(b)$. Ma $S(b) \subset B$ perché B è invariante e quindi $A \subset B$. Ma B era un sottoinsieme di A, e vediamo che $B = A$.

Corollario 2.3. *A sia un sottoinsieme non vuoto di X. Allora sono equivalenti:*

(1) A è invariante minimale.
(2) Per ogni $a, b \in A$ vale $a \in S(b)$.

Osservazione 2.4. *Ogni sottoinsieme invariante non vuoto di X contiene un insieme invariante minimale.*

Osservazione 2.5. *A e B siano due sottoinsiemi invarianti minimali di X. Se A e B hanno un punto in comune, allora $A = B$.*

Dimostrazione. Per la proposizione 2.2 l'orbita di un punto comune coincide sia con A che con B.

Lemma 2.6. *A e B siano sottoinsiemi invarianti della stessa orbita. Allora $A \subset B$ oppure $B \subset A$.*

Dimostrazione. Se uno dei due insiemi è vuoto, l'enunciato è chiaramente vero. Altrimenti per la proposizione 1.14 A e B sono orbite e quindi esistono $a \in A$ e $b \in B$ tali che $A = S(a)$ e $B = S(b)$. Per il lemma 1.5 ad esempio $a \in S(b)$ e quindi $S(a) \subset S(b)$, cioè $A \subset B$.

Teorema 2.7. *Ogni orbita di X contiene esattamente un insieme invariante minimale.*

Dimostrazione. Per l'osservazione 2.4 un'orbita contiene almeno un insieme invariante minimale A. Se B fosse un altro insieme invariante minimale contenuto nella stessa orbita, per il lemma 2.6 si avrebbe $A \subset B$ oppure $B \subset A$ e quindi, essendo A e B insiemi invarianti minimali, necessariamente $A = B$.

Definizione 2.8. Denotiamo con $\Omega(x)$ l'unico sottoinsieme invariante minimale di $S(x)$ e poniamo inoltre

$$d(x) := \min\{k \in \mathbb{N} \mid x + k \in \Omega(x)\}$$
$$\omega(x) := x + d(x)$$

$\omega(x)$ è quindi il primo punto di $\Omega(x)$ che si raggiunge partendo da x. $d(x)$ può essere considerata come la distanza di x da $\Omega(x)$.

Osservazione 2.9. *Sia $m \geq d(x)$. Allora $x + m \in \Omega(x)$.*

Dimostrazione. Chiaro, perché $\Omega(x)$ è invariante.

Teorema 2.10. *Sono equivalenti i seguenti enunciati:*

(1) $S(x)$ è un insieme invariante minimale.
(2) $a \in S(x) \Longrightarrow x \in S(a)$.
(3) Esiste $n > 0$ tale che $x + n = x$.
(4) $x \in \Omega(x)$.
(5) $x = \omega(x)$.
(6) $S(x) = \Omega(x)$.

Dimostrazione. (1) $\Longrightarrow$ (2): Ciò segue dal corollario 2.3.

(2) $\Longrightarrow$ (3): Per ipotesi abbiamo $x \in S(x+1)$. Ciò significa che esiste $m \geq 0$ tale che $x = x + 1 + m$ e quindi $n := m + 1$ soddisfa l'enunciato.

(3) $\Longrightarrow$ (4): Sia $n > 0$ tale che $x + n = x$. Allora $x + nk = x$ per ogni $k \in \mathbb{N}$. Siccome $n > 0$, possiamo però trovare un $k \in \mathbb{N}$ tale che $nk \geq d(x)$. Per l'osservazione 2.9 ciò implica $x = x + nk \in \Omega(x)$.

(4) $\Longleftrightarrow$ (5): Chiaro.

(4) $\Longrightarrow$ (6): Siccome $\Omega(x)$ è invariante, l'ipotesi $x \in \Omega(x)$ implica $S(x) \subset \Omega(x)$. D'altra parte $\Omega(x) \subset S(x)$, cosicché $S(x) = \Omega(x)$.

(6) $\Longrightarrow$ (1): Per definizione $\Omega(x)$ è invariante minimale e coincide, per ipotesi, con $S(x)$.

Corollario 2.11. $\Omega(x) = S(\omega(x))$.

Punti periodici

Definizione 2.12. Poniamo

$$\mathcal{R}(x, A) := \{n \in \mathbb{N} \mid x + n \in A\}$$
$$\mathcal{R}(x) := \mathcal{R}(x, x) = \{n \in \mathbb{N} \mid x + n = x\}$$
$$\mathcal{R}^+(x) := \mathcal{R}(x) \setminus 0 = \{n > 0 \mid x + n = x\}$$

La scelta della lettera $\mathcal{R}$ vorrebbe indicare che, soprattutto nel caso di $\mathcal{R}(x)$, gli elementi di questi insiemi possono essere considerati come *tempi di ritorno*.

È chiaro che si ha sempre $0 \in \mathcal{R}(x)$ e che

$$0 \in \mathcal{R}(x, A) \iff x \in A.$$

Può accadere che $\mathcal{R}(x, A) = \emptyset$, infatti

$$\mathcal{R}(x, A) \neq \emptyset \iff A \cap S(x) \neq \emptyset.$$

Osservazione 2.13. *Per la definizione stessa di $d(x)$ abbiamo*

$$d(x) = \min \mathcal{R}(x, \Omega(x))$$

Definizione 2.14. Il punto x si chiama *periodico* se $\mathcal{R}(x) \neq 0$, cioè se esiste un $n > 0$ tale che $x + n = x$ ed x soddisfa quindi la terza e perciò tutte le condizioni equivalenti del teorema 2.10.

Proposizione 2.15. *Esiste sempre un numero $h \in \mathbb{N}$ tale che $\mathcal{R}(x) = \mathbb{N}h$. Se x non è periodico, $h = 0$, altrimenti*

$$h = \min \mathcal{R}^+(x)$$

Dimostrazione. Se x non è periodico, $\mathcal{R}(x) = 0 = \mathbb{N}0$. Assumiamo quindi che x sia periodico. Allora l'insieme $\mathcal{R}^+(x)$ non è vuoto e possiede quindi un minimo $h > 0$.

Dobbiamo dimostrare che $\mathcal{R}(x) = \mathbb{N}h$.

Siccome h stesso appartiene ad $\mathcal{R}(x)$, è chiaro che $x + h = x$ e quindi $x + kh = x$ per ogni $k \in \mathbb{N}$. Ciò mostra $\mathbb{N}h \subset \mathcal{R}(x)$.

Sia viceversa $n \in \mathcal{R}(x)$. Con l'algoritmo euclideo troviamo una rappresentazione

$$n = kh + r \text{ con } k, r \in \mathbb{N} \text{ e } 0 \leq r < h.$$

Usando la relazione $x + kh = x$ abbiamo allora

$$x + r = x + kh + r = x + n = x$$

per cui $r \in \mathcal{R}(x)$. Dalla minimalità di h segue $r = 0$, per cui $n = kh \in \mathbb{N}h$.

Definizione 2.16. Il numero h della proposizione 2.15 si chiama il *periodo* di x.

Il periodo di x è > 0 se e solo se x è periodico.

Il punto $\omega(x)$ per il teorema 2.10 è sempre periodico; chiamiamo il suo periodo il *periodo asintotico* di x.

Proposizione 2.17. *x sia un punto periodico di X con periodo h. Allora:*

(1) $S(x) = \{x, x + 1, \ldots, x + h - 1\}$.
(2) I punti $x, x + 1, \ldots, x + h - 1$ sono tutti distinti tra di loro.
(3) $|S(x)| = h$.

Dimostrazione. La prima affermazione segue da $x + h = x$, mentre la terza è una conseguenza immediata delle prime due. Rimane da dimostrare la seconda.

Sia $x + n = x + m$ con $0 \leq n < m < h$. Allora

$$\begin{aligned} x + (m - n) &= x + h + (m - n) \\ &= x + m + (h - n) \\ &= x + n + (h - n) \\ &= x + h = x \end{aligned}$$

Ma $0 < m - n < h$, una contraddizione alla minimalità di h.

Corollario 2.18. *La cardinalità di $\Omega(x)$ coincide con il periodo asintotico di x.*

Definizione 2.19. x si chiama un *punto fisso*, se $f(x) = x$.

Osservazione 2.20. *Sono equivalenti:*

(1) x è un punto fisso.
(2) $x + 1 = x$.
(3) $|S(x)| = 1$.
(4) x possiede periodo 1.

Osservazione 2.21. *$\omega(x)$ è un punto fisso se e solo se $|\Omega(x)| = 1$.*

Dimostrazione. Corollario 2.18.

Osservazione 2.22. *Sia $0 \leq n < m$ con $x + n = x + m$. Allora $x + n$ è un punto periodico.*

Dimostrazione. Infatti

$$x + n = x + m = x + n + (m - n)$$

con $m - n > 0$.

Corollario 2.23. *$d(x) < |X|$. Il punto $x + (|X| - 1)$ è quindi sempre periodico.*

Dimostrazione. Siccome due dei punti $x, x + 1, \ldots, x + |X|$ devono coincidere, esistono $n, m \in \mathbb{N}$ con $0 \leq n < m \leq |X|$ tali che $x + n = x + m$. Per l'osservazione 2.22 $x + n$ è periodico e quindi $d(x) \leq n < |X|$.

Installazione di R

Dalla pagina del corso scegliere la pagina di *R* e su questa *CRAN*. Vengono offerti i pacchetti per Linux, per Windows e per Macintosh.

L'installazione per Linux è molto semplice. Si ritira il file *.rpm* adatto per la propria versione di Linux e poi lo si installa, diventando `root`, con

```
rpm -Uvh nome-del-file.rpm
```

A questo punto, tornati utenti normali, basta battere `R` dalla tastiera e il programma parte. Se il file *.rpm* non è aggiornato, si può anche installare la sorgente *.tar.gz*.

Sotto Windows bisogna andare in *base* e scaricare il file *rw2001.exe*.

Benché si tratti di un linguaggio ad alto livello, gli ideatori di R preferiscono presentare R come linguaggio con cui lavorare in linea e non mediante l'esecuzione di programmi scritti su files. Oggigiorno ciò non è perfettamente comprensibile ed è possibile scrivere programmi e farli eseguire nel modo familiare ai programmatori con la tecnica che adesso descriviamo.

Prima creiamo una nuova cartella *Programmi* in cui vogliamo svolgere un determinato lavoro. In essa creiamo le due sottocartelle *Esempi* e *Libreria*. Nella prima conserveremo i nostri esempi; essa non è necessaria per la programmazione. Il contenuto della seconda può essere copiata dal sito del corso.

Sotto Windows dobbiamo creare in *Programmi* un alias per R, in modo che i nomi dei files possano essere indicati in forma abbreviata.

Nella stessa cartella *Programmi* copiamo adesso il file *alfa* dal sito del corso. Il programma principale (che cambia ogni volta a seconda dell'esperimento che stiamo eseguendo) risiederà nel file *programma* che avrà quindi funzioni simili a quelle del file *alfa.c* dei nostri programmi in C. Quando un esperimento ci sembra interessante e ben riuscito, lo copiamo da *programma* in un file della cartella *Esempi*.

Poi lavoriamo in questo modo: Facciamo partire R e battiamo, solo in questa fase,

```
source("alfa")
```

risp. `source("alfa.txt")` sotto Windows. Possiamo fare a meno di questo comando, se utilizziamo il file *.Rprofile*. Successivamente, dopo ogni cambiamento nel file *programma*, è sufficiente battere `Alfa()` dal terminale oppure ripetere questo comando usando il tasto ↑. Si esce da R con `q()`, rispondendo *no* alla domanda se vogliamo salvare l'ambiente di lavoro.

Lo scopo del file *alfa* è quello di rendere disponibile la funzione di esecuzione generale `Alfa` che a sua volta carica le funzioni contenute nella cartella *Libreria*. Il comando `source("alfa")` è necessario solo alla partenza di R perché, come si vede dal listato, verrà automaticamente eseguito ad ogni chiamata di `Alfa`.

In questo modo ogni cambiamento in *programma* o in uno dei files della cartella *Libreria*, diventa effettivo.

La prima riga di *alfa* serve a cancellare tutti gli oggetti definiti precedentemente.

```
rm(list=ls())
#######################################
Caricalibreria = function ()
{for (x in dir("Libreria",
  full.names=TRUE,recursive=TRUE))
  source(x)}

# Sotto Windows "alfa.txt"
# e "programma.txt".
Alfa = function ()
{source("alfa")
Caricalibreria()
source("programma")}
```

A parte l'aggiunta dei suffissi, il file *alfa* non deve essere modificato.

Il file .Rprofile

Se la nostra cartella (la stessa in cui si trovano *alfa* e *programma*) contiene un file *.Rprofile*, i comandi contenuti in questo file vengono eseguiti all'inizio di ogni sessione in R. Inseriamo quindi in esso l'istruzione

`source("alfa")` risp. `source("alfa.txt")`

potendo così successivamente fare a meno di battere questo comando ogni volta che apriamo R. Siccome il file *.Rprofile* rimane nascosto nei normali cataloghi delle cartelle e talvolta anche nel browser, conviene creare prima un file *Rprofile* e di questo un alias con il nome *.Rprofile* riconosciuto da R.

Calcolo di $S(x)$, $\omega(x)$ e $\Omega(x)$

Rimandiamo qui e nel seguito al corso di Algoritmi 04/05 per i termini riguardanti R non spiegati nel testo, utilizzando il seguente formato in cui i numeri alla destra corrispondono alle pagine del corso di Algoritmi.

Nomi in R	17
Assegnamento	17
Commenti	18
`function`	17, 22
`c`	18
`repeat, break`	40
`is.element`	33
`union`	33
`%%`	32
`cat`	17
`for`	40
`a:b`	22
`seq`	18

Alcuni concetti della programmazione in R verranno comunque ripresi o introdotti anche in questo corso.

Non è difficile calcolare l'orbita di x. Possiamo usare il seguente algoritmo, espresso prima in pseudolinguaggio:

$A = \{x\}$
repeat: $x = f(x)$
  if $(x \in A)$ return A
  $A = A \cup x$
end repeat

È quasi immediata la traduzione in R:

```
Din.orbita = function (x,f)
{A=c(x)
repeat {x=f(x);
if (is.element(x,A)) return(A)
A=c(A,x)}}
```

Si osservi che per l'unione abbiamo usato l'istruzione `A=c(A,x)` e non `A=union(A,x)`, essendo la seconda sensibilmente più lenta della prima.

Il `return` di R richiede che l'argomento sia racchiuso tra parentesi.

Per calcolare $\omega(x)$ osserviamo che, per il teorema 2.10, $d(x)$ coincide con il più piccolo numero naturale k per il quale esiste un $n > 0$ tale che $x + k + n = x + k$.

Ciò significa che $\omega(x)$ è il primo punto che si ripete nell'algoritmo che abbiamo usato per l'orbita. È sufficiente quindi sostituire `return(A)` con `return(x)`:

```
Din.omega = function (x,f)
{A=c(x)
repeat {x=f(x);
if (is.element(x,A)) return(x)
A=c(A,x)}}
```

Dal corollario 2.11 sappiamo infine che $\Omega(x) = S(\omega(x))$:

```
Din.Omega = function (x,f)
{Din.orbita(Din.omega(x,f),f)}
```

Consideriamo il sistema dinamico (X, f) in cui $X = \{0, \ldots, 19\}$ ed $f : X \longrightarrow X$ è definita da

$$f(x) = (4x^2 + 11x + 14) \mod 20$$

La funzione corrisponde alla tabella

```
0 14
1 9
2 12
3 3
4 2
5 9
6 4
7 7
8 18
9 17
10 4
11 19
12 2
13 13
14 12
15 19
16 14
17 17
18 8
19 7
```

che otteniamo con

```
f = function (x) (4*x^2+11*x+14)%%20
for (i in 0:19) {cat(i,f(i),'\n')}
```

Disegnare il grafo del sistema. Si vede che il sistema possiede 6 componenti connesse che corrispondono agli insiemi invarianti minimali $\{3\}$, $\{13\}$, $\{8, 18\}$, $\{7\}$, $\{17\}$ e $\{2, 12\}$.

Nel prossimo numero creeremo una funzione in R per un primo disegno automatico e piuttosto imperfetto del grafo di un sistema dinamico finito.

LABORATORIO DI PROGRAMMAZIONE

Corso di laurea in matematica | Anno accademico 2004/05 | Numero 2

Frecce

Nel disegno di grafi, ma anche nei diagrammi commutativi della matematica pura, sono molto utili funzioni per le frecce. R offre a questo scopo la funzione `arrows`*, di cui usiamo il prototipo (perché le funzioni di R possono essere spesso utilizzate con numerose configurazioni dei parametri)*

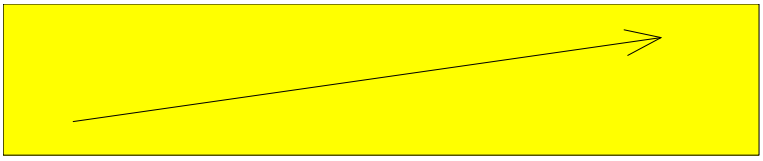
```
arrows(x1,y1,x2,y2,length=0.25,angle=30)
```

in cui `x1,y1` *e* `x2,y2` *sono le coordinate dei due punti che vengono congiunte dalla freccia,* `length` *è la lunghezza in pollici degli uncini alla punta,* `angle` *è l'angolo tra gli uncini e l'asta della freccia. Noi misureremo la lunghezza in centimetri e dobbiamo quindi dividere questi valori per 2.54; l'angolo sarà preimpostato a 20 gradi.*

Oltre a questo vogliamo, come faremo sempre nella grafica, rappresentare i punti del piano come numeri complessi le cui coordinate reali si ottengono mediante le funzioni `Re` *e* `Im` *per le parti reali e immaginarie.*

Una prima semplice funzione per le frecce sarebbe quindi

```
Freccia = function (z1,z2,punta=0.32)
{arrows(Re(z1),Im(z1),Re(z2),Im(z2),length=punta/2.54,angle=20)}
```



Otteniamo questa figura come file in formato PostScript con le istruzioni

```
Gs.postscript("04-frecciasemplice.ps",10,2)
latox=c(0,5); latoy=c(0,1); par(bg="yellow")
Gd.grafica(latox,latoy)
Freccia(0.3+0.2i,4.5+0.8i,punta=0.5)
dev.off()
```

Per vederla sullo schermo è sufficiente togliere (o commentare preponendo il simbolo `#`*) la prima riga e inserire* `locator(1)` *come penultimo comando. In entrambe le modalità non dimenticare* `dev.off()` *alla fine; questo comando chiude il dispositivo grafico.*

Frecce accorciate

Spesso i due punti z_1 e z_2 del piano, tra i quali si vuole disegnare una freccia, servono solo come riferimenti di direzione, mentre si vorrebbe che la freccia si fermi a una certa distanza da questi punti. Dobbiamo quindi congiungere con una freccia i punti p_1 e p_2 che si trovano sul segmento che unisce z_1 e z_2 alle distanze, ad esempio d_1 e d_2, dai due punti di riferimento. Se r è la distanza tra z_1 e z_2, abbiamo

$$p_1 = z_1 + t_1(z_2 - z_1)$$
$$p_2 = z_1 + t_2(z_2 - z_1)$$

con

$$t_1 = \frac{d_1}{r} \quad \text{e} \quad t_2 = \frac{r - d_2}{r}$$

Per calcolare la distanza r usiamo la funzione `abs`. Le distanze d_1 e d_2 sono preimpostate a `0`; se $d_1 + d_2 \geq r$, la freccia non viene disegnata.

```
Gdf = function (z1,z2,punta=0.32,
  d1=0,d2=0)
{r=abs(z2-z1); if (d1+d2>=r) return()
if ((d1==0)&(d2==0))
{arrows(Re(z1),Im(z1),Re(z2),Im(z2),
  length=punta/2.54,angle=20)}
else {p1=z1+(d1/r)*(z2-z1)
p2=z1+((r-d2)/r)*(z2-z1)
Gdf(p1,p2,punta=punta)}}
```

Alcuni richiami al corso di Algoritmi:

Numeri complessi	22-24
`Re`, `Im`	22
`abs`	42
`if ... else`	33
Grafica, `plot`	19,22
`par`	19
`postscript`	19, 36
`locator`	19
`dev.off`	19

In questo numero

Gd.grafica e Gs.postscript

Come nel corso di Algoritmi useremo la funzione `plot` soltanto per predisporre la finestra grafica, non per il disegno stesso. `plot` non apparirà quindi esplicitamente nei nostri programmi, ma solo come elemento della funzione `Gd.grafica` che è usata per la grafica sia sullo schermo che in un file PostScript:

```
Gd.grafica = function (x,y,cornice=T,
  rxy=1,assi=FALSE,marg=0)
{par(mai=rep(marg,4),omi=c(0,0,0,0),lwd=0.5);
plot(x,y,type="n",xlab="",ylab="",
  asp=rxy,axes=assi,frame.plot=cornice)}
```

A differenza dal corso di Algoritmi, la cornice è preimpostata con `cornice=T`.

Vero e falso in R sono dati dai valori `TRUE` e `FALSE` che possono essere abbreviati come `T` e `F`. Avviene comunque una conversione automatica di valori numerici diversi da zero a `TRUE` o viceversa a seconda del contesto e quindi come in C nelle condizioni ad esempio di `if` e `while` espressioni numeriche vengono considerate vere se e solo se sono differenti da zero.

```
print(as.logical(c(0.5,0,-0.3,T,0)))
# output: TRUE FALSE TRUE TRUE FALSE
```

Per conservare un'immagine in un file di formato PostScript si usa il commando `postscript` che abbiamo incorporato in una nostra funzione nel modo seguente:

```
Gs.postscript = function (file,larg,alt)
{postscript(file,width=larg/2.54,
  height=alt/2.54, horizontal=FALSE,
  onefile=FALSE, paper="special")}
```

Larghezza e altezza si riferiscono alle misure dell'immagine quando viene stampata e sono indicate in centimetri.

Se si esegue il programma per la freccia su questa pagina togliendo la prima riga, si vede per un istante lampeggiare la finestra grafica che però si chiude subito. Per poterla guardare bisogna inserire `locator(1)` come penultima riga; il parametro 1 indica che il programma aspetta che clicchiamo una volta sull'immagine prima di chiuderla.

Freccia con uncini nel mezzo

Una freccia con gli uncini nel mezzo tra due punti z_1 e z_2 può essere ottenuta calcolando il punto $p := z_1 + 0.55 * (z_2 - z_1)$ (leggermente spostato verso z_2 per lasciare spazio agli uncini) e unendo prima z_1 a p con una freccia e poi p a z_2 con un segmento di retta. Un segmento di retta può essere ottenuto nel modo più semplice usando la funzione per le freccie con il parametro `punta`, che definisce la lunghezza degli uncini, uguale a 0. Anche per questa funzione prevediamo i parametri `d1` e `d2` di `Gdf`.

```
Gdf.mezzo = function (z1,z2,punta=0.32,
  d1=0,d2=0)
{p=z1+0.55*(z2-z1)
Gdf(p,z2,punta=0,d1=0,d2=d2)
Gdf(z1,p,punta=punta,d1=d1,d2=0)}
```

lines e polygon

La funzione `lines` possiede (nel modo in cui la useremo noi) come argomenti principali un vettore `v` di numeri complessi oppure due vettori `x` e `y` di numeri reali. Nel primo caso unisce i punti rappresentati da `v`, nel secondo caso unisci i punti che si ottengono raccogliendo successivamente l'ascisse da `x` e l'ordinata dalla stessa componente di `y`. Le istruzioni

```
lines(c(2+1i,3-2i,6+4i))
```

e

```
lines(c(2,3,6),c(1,-2,4))
```

sono perciò equivalenti. Si può indicare il colore del tracciato mediante il parametro `col`; per gli altri parametri facoltativi vedere `?lines`.

`lines` viene spesso usata per disegnare grafici di funzioni e di curve parametriche piane. Il grafico del coseno si ottiene ad esempio con

```
x=seq(-2*pi,2*pi,by=0.01)
lines(x,cos(4*x))
```

la cuspide $y^2 = x^3$, che possiede la rappresentazione parametrica $x = t^2, y = t^3$, con

```
t=seq(-10,10,by=0.01)
lines(t^2,t^3)
```

La funzione `polygon` è molto simile a `lines`; anch'essa unisce una successione di punti, chiudendola però, mentre per ottenere un poligono chiuso con `lines` dobbiamo aggiungere il punto di partenza alla successione. `polygon` permette inoltre di colorare l'interno del poligono (oppure di tratteggiarlo in vari modi). Useremo soprattutto questa possibilità per disegnare poligoni colorati, e quindi anche rettangoli, cerchi ed ellissi (che otteniamo scegliendo successioni di punti molto ravvicinati). Come `lines` anche `polygon` permette l'uso di linee interrotte impostando il parametro `lty`; vedere `?par`.

I parametri per definire i colori sono `border` per il bordo e `col` per il colore di riempimento. L'interno del poligono può essere anche tratteggiato usando i parametri `density` e `angle`.

Rotazioni

Una rotazione per l'angolo α attorno all'origine nel piano corrisponde alla moltiplicazione con $e^{i\alpha}$. Usiamo quindi la funzione

```
G.rot = function (z,alfa,centro=0,t=1)
{centro+exp(1i*alfa*Cm.pid180)*(z-centro)*t}
```

La costante `Cm.pid180` è uguale a $\frac{\pi}{180}$ e appare nella funzione per poter indicare l'angolo in gradi; il parametro facoltativo `t` permette di aumentare o diminuire la distanza del punto ruotato dal centro.

Ellissi e cerchi

Le funzioni

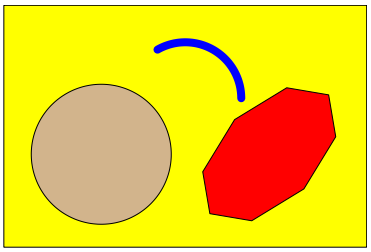
```
Gf.ellisse = function (t,a,b)
{a*cos(t)+1i*b*sin(t)}

Gf.cerchio = function (t,r)
{Gf.ellisse(t,r,r)}
```

definiscono successioni di punti su un'ellisse o su un cerchio; se questi punti sono molto vicini (quando nella definizione di `t` abbiamo usato una risoluzione sufficientemente fine, ad esempio `by=0.01`), otteniamo una curva che all'occhio appare continua, altrimenti un poligono.

Se `t` varia da 0 a 2π, l'ellisse sarà chiusa, scegliendo un intervallo minore si otterrà un arco. Così con

```
Gs.postscript("05-poligoni.ps",
  4.8,3.2)
latox=c(0,4.8); latoy=c(0,3.2);
par(bg="yellow")
Gd.grafica(latox,latoy)
t1=seq(0,2*pi,by=0.01)
centro1=1.2+1.2i
cerchio=Gf.cerchio(t1,1)+centro1
polygon(cerchio,col="tan")
t2=seq(0,2*pi,by=pi/4)
centro2=3.6+1.2i
poligono=Gf.ellisse(t2,1.2,0.7)+centro2
poligono=G.rot(poligono,45,centro2)
polygon(poligono,col="red")
t3=seq(0,2*pi/3,by=0.01)
centro3=2.4+2i
arco=Gf.cerchio(t3,0.8)+centro3
lines(arco,col="blue",lwd=4)
dev.off()
```

otteniamo la figura



Richiami al corso di Algoritmi:

`lines`	19
`polygon`	35
Grafici di funzioni	19, 26-29
Curve piane	28
Tratteggi	35
Rotazioni	21, 28

Linea con cerchietto colorato

Questa funzione è una semplice variazione della funzione `Gdf.mezzo` che disegna una freccia con gli uncini nel mezzo. Stavolta invece degli uncini appare un cerchietto colorato. L'avremmo potuto utilizzare per rappresentare frecce in entrambe le direzioni, nei sistemi dinamici quindi quando si ha simultaneamente $y = f(x)$ ed $x = f(y)$.

```
Gdf.cerchio = function (z1,z2,r=0.08,
  d1=0,d2=0,col=1)
{p=z1+0.5*(z2-z1)
Gdf(z1,z2,punta=0,d1=d1,d2=d2)
t=seq(0,2*pi,by=0.01)
cerchio=p+Gf.cerchio(t,r)
polygon(cerchio,col=col)}
```

Liste

Mentre i componenti di un vettore devono essere tutti dello stesso tipo e quando ad esempio si definisce

```
v=c(1,2,"a")
```

anche i componenti numerici vengono convertiti in stringhe, come si vede con

```
v=c(1,2,"a")
print(v)
# output: "1" "2" "a"
```

Liste vengono create con il comando `list` e possono contenere altre liste o vettori come componenti:

```
L=list(1,2,"a",c(2,4))
```

L'i-esimo elemento di una lista `L` lo si ottiene con `L[[i]]`.

P.quale

Creiamo adesso una funzione, simile allo `switch` e all'operatore ternario *punto interrogativo* del C, che permette di definire rapidamente funzioni mediante un elenco dei valori che la funzione assume. Esempi:

`P.quale(x,5,v1,8,v2)` è uguale a `v1` se `x` è uguale a 5 e uguale a `v2`, se `x` è uguale a 8, e corrisponderebbe quindi in C a

```
x==5 ? v1 : x==8 ? v2
```

mentre `P.quale(x,5,v1,8,v2,v3)` assume gli stessi valori dell'espressione precedente per `x` uguale a 5 o ad 8, e il valore `v3` in tutti gli altri casi, corrispondendo quindi in C a

```
x==5 ? v1 : x==8 ? v2 : v3
```

La funzione è così definita in R:

```
P.quale = function (x,...)
{a=list(...); i=1; n=length(a)
while (i<n)
{if (x==a[[i]]) return(a[[i+1]])
i=i+2}
if (i==n) return(a[[n]])}
```

`x` e i valori con cui viene confrontato devono essere numerici. Si noti l'uso di `list(...)` in funzioni con un numero variabile di argomenti.

Argomenti ignoti (...)

In una funzione si possono usare anche argomenti non noti in anticipo; essi vengono indicati con `...`. Nel corpo della funzione questi argomenti devono poi essere trasformati in una lista con un'istruzione della forma

```
a=list(...)
```

spesso unita a un'istruzione che ha lo scopo di determinare il numero degli argomenti, per esempio

```
n=length(a)
```

come nella definizione di `P.quale`.

match

`match(x,a)` restituisce l'indice della prima apparizione di `x` nel vettore o nella lista `a`. Se `x` non appare in `a`, il valore restituito è `NA`.

Altri richiami:

Funzioni d'aiuto	16
`round`	25
`length`	42
`NA`	32
`is.null`	35
`text`	26-28
`lwd`	19
Indici vettoriali	40

Din.grafo

Definiamo adesso una funzione che disegna il grafo di un sistema dinamico finito (X, f). X deve essere rappresentato come un vettore di numeri. Proviamo prima con una versione molto semplice a cui aggiungeremo successivamente altre funzionalità per ottenere uno strumento che, in modo semiautomatico, fornisce risultati piuttosto soddisfacenti.

```
Din.grafo = function (X,f,r=2,
  cex=0.5,lwd=0.8,punta=0.2)

{n=length(X);
t=seq(0,length=n,by=2*pi/n);
centri=round(Gf.cerchio(t,r),2)

t=seq(0,2*pi,by=0.01)
cerchio=Gf.cerchio(t,0.2)

for (k in 1:n)
{polygon(cerchio+centri[k],col="yellow");
text(Re(centri[k]),Im(centri[k]),
  X[k],cex=cex)}

par(lwd=lwd)
for (x in X) {y=f(x); if (y!=x)
Gdf.mezzo(centri[match(x,X)],
  centri[match(y,X)],
  punta=0.2,d1=0.2,d2=0.2)}}
```

Abbiamo suddiviso il codice in gruppi di istruzioni separati da righe vuote. Il primo gruppo consiste dell'intestazione, in cui oltre ai parametri `X` ed `f` che definiscono il sistema dinamico appare il raggio `r` (in centimetri) del cerchio su cui verranno disposti i dischetti che rappresentano i punti del sistema.

Nel secondo gruppo vengono calcolati i centri di questi dischetti.

Il terzo gruppo definisce semplicemente la forma dei dischetti rappresentata da un cerchio di raggio `0.2` nell'origine.

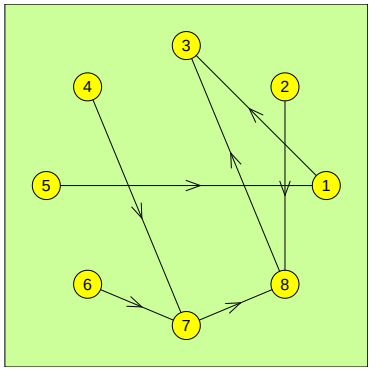
Nel quarto gruppo per ogni punto del sistema il cerchio viene trasferito nella posizione del dischetto corrispondente (data da `cerchio+centri[k]`); il dischetto viene disegnato mediante `polygon` e colorato di giallo; con `text` facciamo apparire nel dischetto il nome del punto che in questa prima versione è semplicemente il numero a cui il punto è uguale.

Nell'ultimo gruppo disegniamo le frecce (con uncini nel mezzo) da `x` a `f(x)` per tutti i punti che non sono punti fissi. Si noti qui che i centri sono numerati da `1` a `n`, dobbiamo quindi calcolare con `match` la posizione con cui `x` appare nel vettore `X`. Se ad esempio `X` è uguale a `c(5,1,7,9,8)`, al punto `7` corrisponderà il dischetto in `centri[3]`.

Applichiamo adesso questa prima versione al sistema considerato a pagina 1.

```
X=1:8
f = function (x)
  P.quale(x,1,3,2,8,3,3,4,7,5,1,6,7,
    7,8,8,3)

Gs.postscript("06-dingrafo-1.ps",
  4.8,4.8)
lato=c(-2.4,2.4); par(bg="#ccff99")
Gd.grafica(lato,lato,cornice=T)
Din.grafo(X,f)
dev.off()
```

otteniamo così

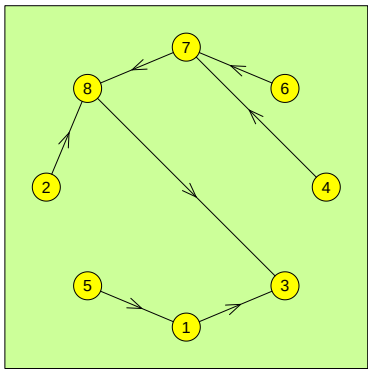


Quando guardiamo la figura sullo schermo useremo piuttosto le impostazioni `cex=1`, `lwd=1`, `punta=0.3`.

Possiamo facilmente migliorare la figura, senza modificare la funzione `Din.grafo` stessa, semplicemente raggruppando in modo diverso i punti del sistema, cercando di porre vicini tra di loro i punti uniti da frecce. Ad esempio con

```
X=c(4,6,7,8,2,5,1,3)
```

otteniamo in questo sistema molto semplice già un grafo in cui le frecce non si intersecano più:



Vogliamo adesso fare in modo che i punti periodici, cioè i punti $x \in X$ per cui x coincide con $\omega(x)$, siano contrassegnati da un bordo rosso nel dischetto che li rappresenta. A questo aggiungiamo le seguenti istruzioni alla fine della definizione di `Din.grafo`:

```
for (x in X) if (x==Din.omega(x,f))
lines(cerchio+centri[match(x,X)],
  col="red",lwd=2)
```

Nonostante la nostra funzione richieda che gli elementi di X siano rappresentati da numeri, vorremmo talvolta indicare i punti del sistema nel grafo anche con altri nomi.

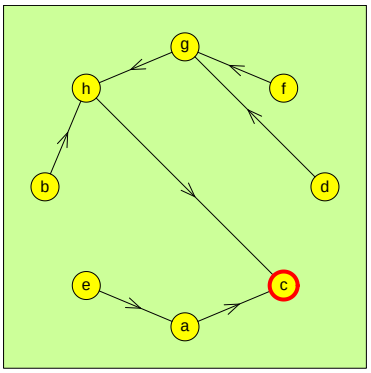
Aggiungiamo per questa ragione ai parametri di `Din.grafo` una funzione `nomi`, uguale nella preimpostazione all'identità, che fornisce i nomi che vengono usati nell'istruzione `text` del quarto gruppo, sostituendo in questa `X[k]` con `nomi(X[k])`. La definizione di `Din.grafo` a questo punto è la seguente:

```
Din.grafo = function (X,f,r=2,
  cex=0.5,lwd=0.3,punta=0.2,
  nomi=function (x) x)

{n=length(X);
t=seq(0,length=n,by=2*pi/n);
centri=round(Gf.cerchio(t,r),2)

t=seq(0,2*pi,by=0.01)
cerchio=Gf.cerchio(t,0.2)

for (k in 1:n)
{polygon(cerchio+centri[k],col="yellow");
text(Re(centri[k]),Im(centri[k]),
  nomi(X[k]),cex=cex)}

par(lwd=lwd)
for (x in X) {y=f(x); if (y!=x)
Gdf.mezzo(centri[match(x,X)],
  centri[match(y,X)],
  punta=punta,d1=0.2,d2=0.2)}

for (x in X) if (x==Din.omega(x,f))
lines(cerchio+centri[match(x,X)],
  col="red",lwd=2)}
```

Con

```
nomi = function (x)
  P.quale(x,1,"a",2,"b",3,"c",4,"d",
    5,"e",6,"f",7,"g",8,"h")

X=c(4,6,7,8,2,5,1,3)
f = ... # come prima

Gs.postscript("06-dingrafo-3.ps",4.8,4.8)
lato=c(-2.4,2.4); par(bg="#ccff99")
Gd.grafica(lato,lato,cornice=T)
Din.grafo(X,f,nomi=nomi)
dev.off()
```

otteniamo allora la figura



Rotazione e scambio dei punti

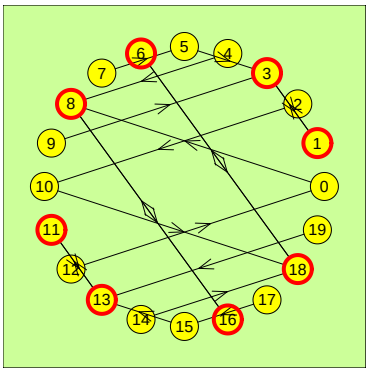
Consideriamo adesso il sistema (X, f) in cui $X = \{0, \ldots, 19\}$ e f è definita da

$$f(x) := (2x^3 + 13x + 8) \mod 20$$

Sostituendo

```
X=0:19
f = function (x) (2*x^3+13*x+8)%%20
```

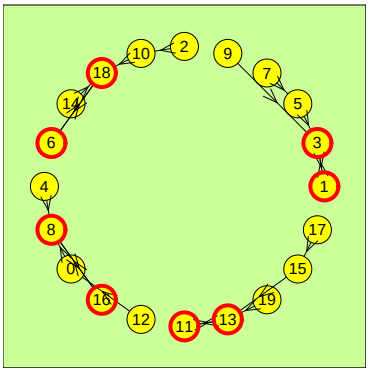
nell'ultimo programma a pagina 6 e rinunciando ai nomi otteniamo la figura



Seguendo le frecce, non è difficile intravvedere le componenti connesse del sistema, quindi proviamo con

```
X=c(1,3,5,7,9,2,10,18,14,6,4,8,
  0,16,12,11,13,19,15,17)
```

ottenendo



Per rendere il grafo più chiaro, vogliamo spostare alcuni dei punti girandoli attorno all'origine oppure scambiandoli tra di loro. A questo scopo aggiungiamo ai parametri di `Din.grafo` una funzione `rotazioni` che restituisce per ogni punto l'angolo di rotazione e il fattore `t` nella funzione `G.rot` a pagina 5 e un vettore che elenca le coppie di punti da scambiare. Le preimpostazioni sono:

```
scambi=c()
rotazioni=function (x) c(0,1)
```

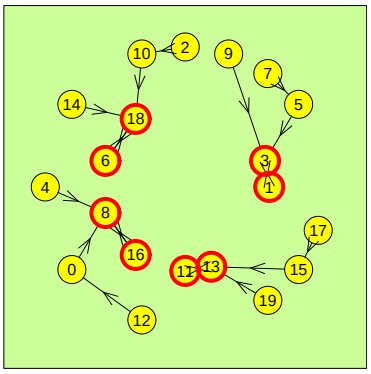
La versione finale di `Din.grafo` è quindi:

```
Din.grafo = function (X,f,r=2,
  cex=0.5,lwd=0.3,punta=0.2,
  nomi=function (x) x, scambi=c(),
  rotazioni=function (x) c(0,1))

{n=length(X);
t=seq(0,length=n,by=2*pi/n);
centri=round(Gf.cerchio(t,r),2)
for (k in 1:n) {u=rotazioni(X[k]);
  centri[k]=G.rot(centri[k,u[1],t=u[2])}

m=length(scambi)/2
if (m>0) for (k in 1:m)
{x=scambi[2*k-1]; y=scambi[2*k]
ix=match(x,X); iy=match(y,X)
cx=centri[ix];
centri[ix]=centri[iy]; centri[iy]=cx}

t=seq(0,2*pi,by=0.01)
cerchio=Gf.cerchio(t,0.2)

for (k in 1:n)
{polygon(cerchio+centri[k],col="yellow");
text(Re(centri[k]),Im(centri[k]),
  nomi(X[k]),cex=cex)}

par(lwd=lwd)
for (x in X) {y=f(x); if (y!=x)
Gdf.mezzo(centri[match(x,X)],
  centri[match(y,X)],
  punta=punta,d1=0.2,d2=0.2)}
for (x in X) if (x==Din.omega(x,f))
lines(cerchio+centri[match(x,X)],
  col="red",lwd=2)}
```

Portiamo adesso i punti periodici più vicini al centro utilizzando come parametro `rotazioni` la funzione

```
rota = function (x)
{if (x==Din.omega(x,f)) c(0,0.6)
else c(0,1)}
```

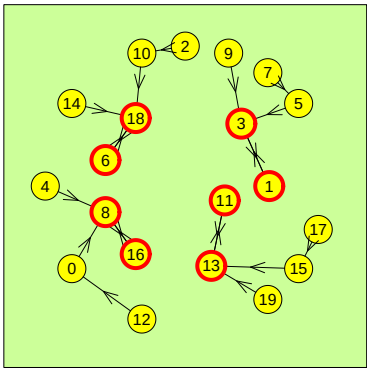
ottenendo



Spostiamo ancora i punti 3 e 11 con la modifica

```
rota = function (x)
{if (x==Din.omega(x,f)) u=c(0,0.6)
else u=c(0,1)
P.quale(x,3,c(30,0.6),11,c(70,0.3),u)}
```

e otteniamo



Punti iniziali

Situazione 7.1. (X, f) sia un sistema dinamico finito ed $x, y \in X$.

Definizione 7.2. $\Omega(X) := \bigcup_{x \in X} \Omega(x)$ sia l'insieme di tutti i punti periodici di X.

Definizione 7.3. $S^+(x) := S(x) \setminus x$.

Definizione 7.4. x si chiama un *punto iniziale* di X, se non è successore di un punto non periodico $\neq x$.

Osservazione 7.5. *L'insieme dei punti non iniziali coincide quindi con* $\bigcup_{x \notin \Omega(X)} S^+(x)$.

Otteniamo perciò l'insieme dei punti iniziali con il seguente algoritmo:

$N = \emptyset$
for $x \in X$
  if (x==$\omega(x)$) continue
  $N = N \cup S^+(x)$
end for
  return $X \setminus N$

che possiamo direttamente tradurre in R, utilizzando l'istruzione `next` che in R corrisponde al `continue`:

```
Din.iniziali = function (X,f)
{N=c(); for (x in X)
{if (x==Din.omega(x,f)) next
N=union(N,setdiff(Din.orbita(x,f),x))}
setdiff(X,N)}
```

Calcoliamo ad esempio i punti iniziali del sistema considerato su questa pagina:

```
X=0:19
f = function (x) (2*x^3+13*x+8)%%20

print(Din.iniziali(X,f))
# output: 2 4 7 9 12 14 17 19
```

Analizzando il grafico ci convinciamo facilmente che il risultato è corretto.

Osservazione 7.6. *x è periodico se e solo se esiste $y \neq x$ tale che $S(x) = S(y)$.*

Dimostrazione. Ciò segue direttamente dall'osservazione 2.22 e dal teorema 2.10.

Proposizione 7.7. *x è un punto iniziale se e solo se $S(x)$ è un orbita massimale.*

Dimostrazione. (1) x sia un punto iniziale ed $S(x) \subsetneq S(y)$. Ciò implica $x \in S(y)$ e quindi y deve essere un punto periodico, perché $y \neq x$ ed x è un punto iniziale. Ma allora $S(x) = S(y)$ per il teorema 2.10, una contraddizione.

(2) $S(x)$ sia un'orbita massimale ed $x \in S^+(y)$. Allora $S(x) \subset S(y)$ e dalla massimalità segue $S(x) = S(y)$. Siccome $y \neq x$, l'osservazione 7.6 implica che y è periodico.

Esercizio 7.8. Per un sottoinsieme $A \neq \emptyset$ di X sono equivalenti:

(1) A è una catena massimale.
(2) A è una catena invariante massimale.
(3) A è un'orbita massimale.
(4) A è orbita di un punto iniziale.

LABORATORIO DI PROGRAMMAZIONE

Corso di laurea in matematica | Anno accademico 2004/05 | Numero 3

Orbite minimali e massimali

Situazione 8.1. (X, f) sia un sistema dinamico finito ed $x, y \in X$.

Osservazione 8.2. *A sia un sottoinsieme di X. Allora sono equivalenti:*

(1) A è orbita di un punto periodico.

(2) A è un insieme invariante minimale.

(3) A è un'orbita minimale.

Dimostrazione. (1) $\Leftrightarrow$ (2): Teorema 2.10.

(2) $\implies$ (3): A sia invariante minimale. Per la proposizione 2.2 A è un'orbita. Per la minimalità A non può contenere un'orbita diversa da A, perché un'orbita è un insieme invariante non vuoto.

(3) $\implies$ (2): A sia un'orbita minimale e B un sottoinsieme invariante non vuoto contenuto in A. Sia $b \in B$. Allora $S(b) \subset B \subset A$. Per ipotesi però A è un orbita minimale e quindi $S(b) = A$. Ciò implica che anche $B = A$.

Osservazione 8.3. *Ogni orbita è contenuta in un'orbita massimale. Quindi anche ogni punto è contenuto in un'orbita massimale.*

Dimostrazione. Siccome X è finito, ogni catena ascendente di sottoinsiemi, e quindi anche ogni catena ascendente di orbite, deve terminare.

Corollario 8.4. *Ogni punto è contenuto nell'orbita di un punto iniziale.*

Dimostrazione. Per l'osservazione 8.3 ogni punto è contenuto in un'orbita massimale. Questa, per la proposizione 7.7, è l'orbita di un punto iniziale.

Definizione 8.5. Denotiamo con $I := I(X)$ l'insieme dei punti iniziali di X e poniamo

$$I(x) := \{u \in I \mid x \in S(u)\}$$

$I(x)$ è quindi l'insieme dei punti iniziali nella cui orbita x è contenuto.

Definizione 8.6. Diciamo che x *raggiunge* un sottoinsieme A di X, se $S(x) \cap A \neq \emptyset$.

Osservazione 8.7. *Un'orbita è allo stesso tempo minimale e massimale se e solo se è orbita di un punto periodico e non è raggiunta da punti esterni ad essa.*

Componenti connesse di un sistema dinamico finito

Lemma 8.8. *Per ogni $n \in \mathbb{N}$ esiste $r \geq n$ tale che $x + r = \omega(x)$.*

Dimostrazione. h sia il periodo asintotico di X. Allora $h \geq 1$, perciò esiste $j \in \mathbb{N}$ tale che $d(x) + jh \geq n$. Inoltre

$$x + d(x) + jh = \omega(x) + jh = \omega(x)$$

Teorema 8.9. *Sono equivalenti:*

(1) $S(x) \cap S(y) \neq \emptyset$.

(2a) $\omega(x) \in S(y)$.

(2b) $\omega(y) \in S(x)$.

(3a) $\Omega(x) \subset S(y)$.

(3b) $\Omega(y) \subset S(x)$.

(4) $\Omega(x) = \Omega(y)$.

Dimostrazione. (1) $\implies$ (2a): Per ipotesi esistono $n, m \in \mathbb{N}$ tali che $x + n = y + m$. Per il lemma 8.8 esiste $k \geq 0$ tale che

$$\omega(x) = x + n + k = y + m + k \in S(y).$$

(1) $\implies$ (2b): Nello stesso modo.

(2a) $\implies$ (3a): Per il corollario 2.11 e l'invarianza di $S(y)$ abbiamo

$$\Omega(x) = S(\omega(x)) \subset S(y).$$

(2b) $\implies$ (3b): Nello stesso modo.

(3a) $\implies$ (4): L'unico sottoinsieme invariante minimale contenuto in $S(y)$ è $\Omega(y)$. Ma anche $\Omega(x)$ è invariante minimale, quindi $\Omega(x)$ e $\Omega(y)$ devono coincidere.

(3b) $\implies$ (4): Nello stesso modo.

(4) $\implies$ (1): Chiaro.

Definizione 8.10. Per il teorema 8.9 possiamo introdurre una relazione di equivalenza $\sim_c$ su X ponendo

$$x \sim_c y \iff \Omega(x) = \Omega(y)$$

In questo caso diciamo anche che i punti x ed y sono *connessi* (tra di loro). Per la condizione (1) del teorema 8.9 ciò accade se e solo se si può arrivare da x ad y andando avanti sull'orbita di x fino a quando si incontra un punto dell'orbita di y e percorrendo poi quest'ultima all'indietro,

Le classi di equivalenza che così otteniamo si chiamano *componenti connesse* (dette spesso semplicemente componenti) del sistema. Esse sono evidentemente le componenti connesse del grafo non diretto che si ottiene dal grafo del sistema ignorando le direzioni delle frecce. La classe di equivalenza di x è

$$\mathcal{C}(x) := \{y \in X \mid \Omega(y) = \Omega(x)\}$$

Per la definizione stessa le componenti di X corrispondono biunivocamente alle orbite minimali; ogni componente contiene un'unica orbita minimale.

Osservazione 8.11. *Ogni componente connessa è invariante.*

Dimostrazione. Ciò intuitivamente è ovvio e segue ad esempio dalla condizione (2a) del teorema 8.9.

Osservazione 8.12. *Sia $x \in S(y)$. Allora $\mathcal{C}(y) = \mathcal{C}(x)$.*

Dimostrazione. L'ipotesi implica $x \in S(x) \cap S(y)$ e l'enunciato segue dal teorema 8.9.

Corollario 8.13. *Sia $u \in I(x)$. Allora $\mathcal{C}(u) = \mathcal{C}(x)$.*

Ogni componente connessa è quindi la componente connessa di un punto iniziale.

Corollario 8.14. $\mathcal{C}(x) = \bigcup_{u \in I \cap \mathcal{C}(x)} S(u)$.

Dimostrazione. Sia $y \in \mathcal{C}(x)$. Per il corollario 8.4 esiste $u \in I(y)$ tale che $y \in S(u)$. Allora $\mathcal{C}(u) = \mathcal{C}(y) = \mathcal{C}(x)$.

In questo numero

Calcolo delle componenti

Nota 8.15. L'idea dell'algoritmo è la seguente.

Per il corollario 8.13 dobbiamo calcolare solo le componenti connesse dei punti iniziali. Raccogliamo le componenti in un insieme L che nel programma è rappresentato da una lista.

(1) Calcoliamo l'insieme I dei punti iniziali di X e poniamo $L = \emptyset$.

(2) Percorriamo con u tutti gli elementi di I.
All'inizio di ogni passaggio poniamo la variabile `nuova`, che indica che u non appartiene a una delle componenti che abbiamo già iniziato a costruire, uguale a `T`, calcoliamo $x := \omega(u)$, l'orbita $O := S(u)$ e la lunghezza attuale m della lista. Quindi $L = \{C_1, \ldots, C_m\}$.

(3) All'interno del ciclo (2), se $m > 0$ (cfr. il primo articolo a pagina 9), percorriamo con k l'insieme $\{1, \ldots, m\}$ e controlliamo se $x \in C_k$. In tal caso sostituiamo C_k con $C_k \cup O$ e usciamo dal ciclo (3) dopo aver posto `nuova` uguale ad `F`.

(4) Sempre all'interno di (2), se `nuova` è ancora uguale a `T` (cioè se in (3) non è stata trovata un C_k esistente a cui si arriva da u), aggiungiamo O come nuovo elemento ad L. A questo punto il ciclo (2) ricomincia con il prossimo $u \in I$.

(5) Dopo aver eseguito questi passaggi per tutti gli elementi di I, restituiamo L come risultato della funzione.

Si noti che solo alle fine dell'algoritmo i C_k sono componenti connesse complete; nei passi intermedi devono essere ancora riempite unendo agli insiemi già esistenti le nuove orbite. L'algoritmo è di natura matematica e non particolarmente efficiente.

Traduciamo l'algoritmo in R:

```
Din.componenti = function (X,f)
{I=Din.iniziali(X,f); L=list();
for (u in I)
{nuova=T; x=Din.omega(u,f);
O=Din.orbita(u,f); m=length(L);
if (m>0) for (k in 1:m)
{C=L[[k]]; if (is.element(x,C))
{L[[k]]=union(C,O); nuova=F; break;}}
if (nuova) L=c(L,list(O))}; L}
```

Si noti, nell'ultima riga, il modo un po' complicato con cui si aggiunge un vettore come elemento a una lista.

Una brutta sorpresa nel for

Nella funzione `Din.componenti` vista a pagina 8 appare la sequenza

```
if (m>0) for (k in 1:m)
```

che in C corrisponderebbe a

```
if (m>0) for (k=0;k<m;k++)
```

Il programmatore in C sa però che non è necessario in questo caso anteporre al `for` la condizione `if (m>0)`, perché, se `m` non fosse maggiore di 0, il ciclo non verrebbe eseguito.

Perché in R dobbiamo allora inserire `if (m>0)`? La ragione è che il `for` percorre l'insieme `1:m` che, quando m è, come nel nostro programma poteva accadere, uguale a 0, corrisponde al vettore `c(1,0)`, e quindi l'istruzione che segue il `for` verrebbe eseguita due volte, prima per `k` uguale a 1 e poi per `k` uguale a 0.

R infatti definisce `a:b` per `b` minore di `a` come il vettore `c(a,a-1,a-2,...,b)`, con gli elementi elencati in ordine decrescente. Questa piccola comodità può confondere parecchio il programmatore.

Un esempio con 31 punti

Studiamo il sistema (X, f) in cui
$X = \{0, 1, \ldots, 30\}$ ed $f : X \longrightarrow X$ è data da

$$f(x) := x^3 + 23x^2 + 7x + 24 \mod 31$$

La prima figura su questa pagina la otteniamo con

```
L=Din.componenti(X,f)
X=c(L,recursive=T)

rota = function (x)
{if (x==Din.omega(x,f)) u=c(0,0.6)
else u=c(0,1); u}

Din.grafo(X,f,r=7,rotazioni=rota)
```

L'opzione `recursive=T` nella seconda riga fa in modo che i vettori che compongono la lista vengano uniti in un unico vettore.

Provando interattivamente sullo schermo troviamo i seguenti spostamenti con cui otteniamo la seconda figura:

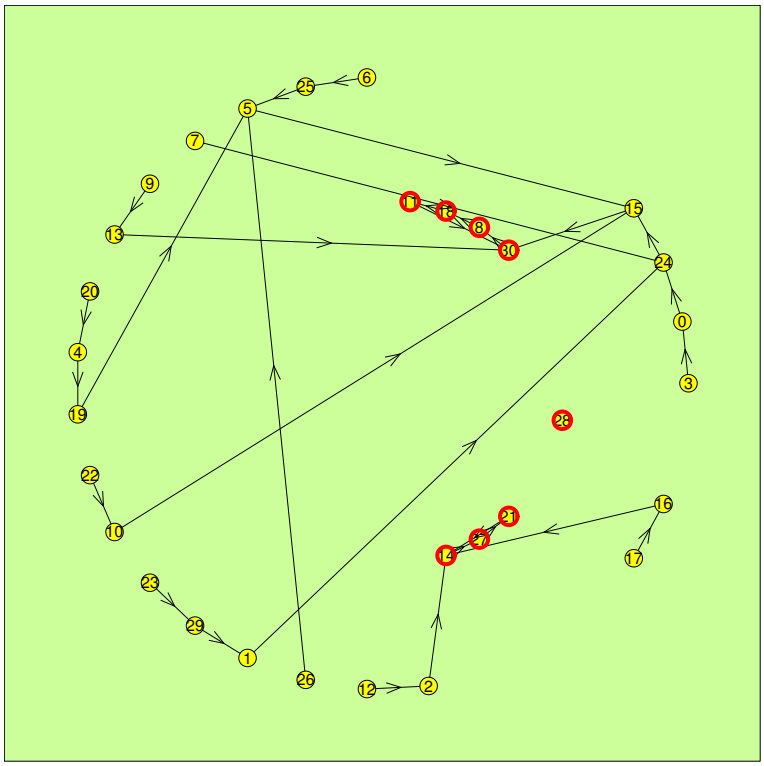
```
rota = function (x)
{if (x==Din.omega(x,f)) u=c(0,0.6)
else u=c(0,1)
P.quale(x,27,c(0,0.4),18,c(10,0.4),
  8,c(0,0.4),10,c(210,1),22,c(210,1),
  7,c(-90,0.8),1,c(130,0.7),29,c(140,0.4),
  23,c(130,0.2),26,c(-140,1.2),
  9,c(-30,0.7),13,c(0,0.5),28,c(220,0.6),u)}

scambi=c(16,17,10,22,30,11,8,18,19,20,9,13)
```

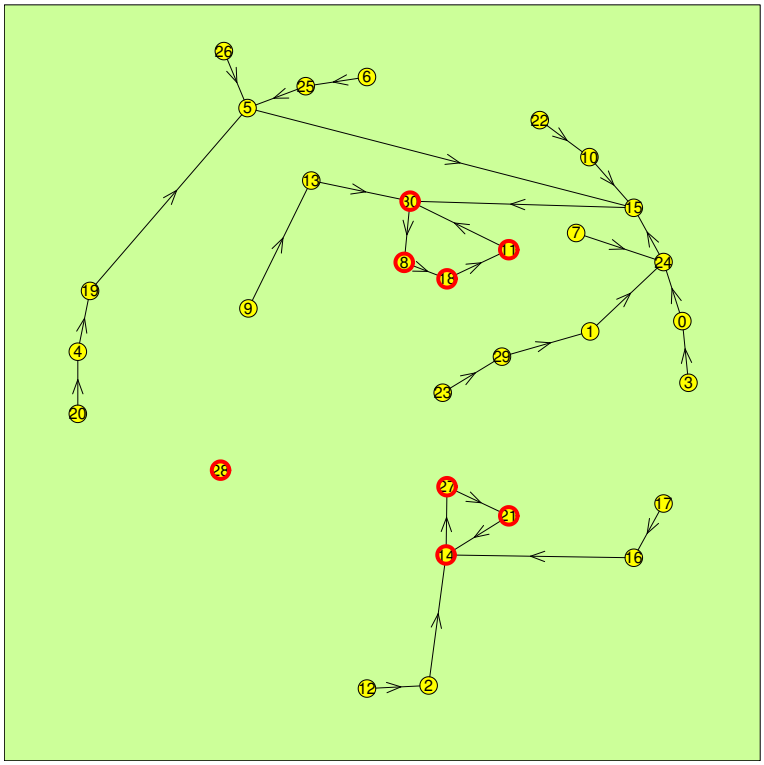
Si vede già dalla figura che le componenti connesse sono state calcolate correttamente. Possiamo anche verificarlo numericamente:

```
n=31
X=0:(n-1)
f = function (x)
{(x^3+23*x^2+7*x+24)%%n}

L=Din.componenti(X,f)
for (C in L) print(C)
```

L'output è corretto:

```
3 0 24 15 30 8 18 11 6 25 5 7 9 13 20
    4 19 22 10 23 29 1 26
12 2 14 27 21 17 16
28
```





Calcoliamo ancora i punti iniziali con

```
I=Din.iniziali(X,f)
print(I)
```

trovando, in accordo con quanto si veda nella figura,

```
3 6 7 9 12 17 20 22 23 26 28
```

Decomposizione in alberi

Definizione 10.1. Introduciamo su X una relazione di equivalenza $\sim_\omega$ definita da

$$x \sim_\omega y :\iff \omega(x) = \omega(y)$$

La partizione che si ottiene decompone X in alberi; ciascuno di questi alberi contiene esattamente un punto periodico che è allo stesso tempo l'elemento massimale nell'albero a cui appartiene (rispetto al quasiordine $\leq$ introdotto nella definizione 1.7). Naturalmente le classi di equivalenza non sono, in genere, invarianti, tranne nel caso che il punto periodico in cui terminano sia un punto fisso.

È chiaro inoltre che abbiamo una biiezione canonica tra le classi di equivalenza di $\sim_\omega$ e l'insieme $\Omega(X)$ dei punti periodici di X. Infatti se consideriamo l'applicazione suriettiva

$$\begin{aligned} \omega : X &\longrightarrow \Omega(X) \\ x &\longmapsto \omega(x) \end{aligned}$$

allora abbiamo la biiezione

$$\begin{aligned} \Omega(X) &\longrightarrow X/\omega \\ z &\longmapsto \omega^{-1}(z) \end{aligned}$$

che corrisponde al diagramma commutativo nel teorema 12.12 del corso di algoritmi dell'anno scorso.

Sottoinsiemi biiettivi

Definizione 10.2. Un sottoinsieme invariante A di X si dice *biiettivo*, se l'applicazione f_A è biiettiva.

Nota 10.3. A sia un sottoinsieme invariante minimale di X. Per definizione allora A non è vuoto e contiene un punto a. Combinando la proposizione 2.2 e il teorema 2.10 vediamo che $A = S(a) = \Omega(a)$. Viceversa ogni insieme della forma $\Omega(x)$ per $x \in X$ è un sottoinsieme invariante minimale. Dalla proposizione 2.17 segue che f opera su $\Omega(x)$ come una permutazione ciclica.

Corollario 10.4. *Ogni sottoinsieme invariante minimale di X è biiettivo.*

Proposizione 10.5. *f sia biiettiva. Allora ogni punto di X è periodico.*

Dimostrazione. Siano $x \in X$ ed h il periodo asintotico di x. Allora

$$h > 0 \text{ e } x + d(x) + h = x + d(x)$$

Questa equazione può essere scritta nella forma

$$(x + h) + d(x) = x + d(x)$$

Ma con f anche l'applicazione $f^{d(x)}$ è biiettiva e quindi $x + h = x$. Siccome $h > 0$, vediamo che x è periodico.

Teorema 10.6. *Per un sottoinsieme A di X sono equivalenti:*

(1) $f(A) = A$.
(2) *A è invariante e biiettivo.*
(3) *A è invariante e ogni punto di A è periodico.*
(4) *A è unione (necessariamente disgiunta per l'osservazione 2.5) di orbite minimali.*

Dimostrazione. (1) $\implies$ (2): È chiaro che A è invariante e che quindi f_A è ben definita e suriettiva. Ma un'applicazione suriettiva da un insieme finito in se stesso è biiettiva.

(2) $\implies$ (3): Applichiamo la proposizione 10.5 al sistema dinamico (A, f_A).

(3) $\implies$ (4): Siccome A è invariante, si ha $A = \bigcup_{a \in A} S(a)$. Per ipotesi ogni punto di A è periodico, quindi per ogni $a \in A$ si ha $S(a) = \Omega(a)$, cosicché $A = \bigcup_{a \in A} \Omega(a)$.

(4) $\implies$ (3): Chiaro.

(3) $\implies$ (1): Sia $a \in A$. Per ipotesi a è periodico, quindi esiste $n > 0$ tale che $a + n = a$. Ciò significa $a = f(a + n - 1)$. Siccome A è invariante, si ha $a + n - 1 \in A$ e vediamo che $a \in f(A)$.

Nota 10.7. Il teorema 10.6 è una riformulazione e leggera generalizzazione della decomposizione di una permutazione in cicli, spesso dimostrata nei corsi di algebra.

Sincronia

Definizione 10.8. I punti x ed y si chiamano *sincroni*, se esiste un $n \in \mathbb{N}$ tale che $x + n = y + n$. In tal caso scriviamo $x \cong y$.

La relazione $\cong$ si chiama la relazione di sincronia del sistema dinamico dato.

Osservazione 10.9. *La relazione di sincronia è una relazione di equivalenza.*

Dimostrazione. Riflessività e simmetria sono evidenti. Dimostriamo la transitività.

Siano $x + n = y + n$ ed $y + m = z + m$. Allora

$$\begin{aligned} x + (n + m) &= y + n + m = y + m + n \\ &= z + m + n = z + (n + m) \end{aligned}$$

Osservazione 10.10. *Se x ed y sono sincroni, allora $\Omega(x) = \Omega(y)$.*

Dimostrazione. Infatti, se $x + n = y + n$, allora $S(x) \cap S(y) \neq \emptyset$ e l'enunciato segue dal teorema 8.9.

Lemma 10.11. *Siano $x \cong y$ ed $r := \max(d(x), d(y))$. Allora*

$$y + r = x + r = \omega(x)$$

Dimostrazione. Siano $x + n = y + n$ con $n \in \mathbb{N}$ e ad esempio $d(x) \geq d(y)$, quindi $r = d(x)$. Allora

$$\begin{aligned} x + d(x) + n &= x + n + d(x) \\ &= y + n + d(x) \\ &= y + d(x) + n \end{aligned}$$

Siccome $d(x) \geq d(y)$, i punti $x + d(x)$ e $y + d(x)$ appartengono entrambi ad $\Omega(x)$ (cfr. osservazione 10.10). Ma su $\Omega(x)$ l'applicazione f è biiettiva, perciò $x + d(x) = y + d(x)$.

Teorema 10.12. *Sia $r := \max(d(x), d(y))$. Sono equivalenti:*

(1) $x \cong y$.
(2) $x + r = y + r$.
(3) $\omega(x) + d(y) = \omega(y) + d(x)$.

Dimostrazione. (1) $\implies$ (2): Lemma 10.11.

(2) $\implies$ (1): Chiaro.

(2) $\implies$ (3): Sia ad esempio $d(x) \geq d(y)$, quindi $r = d(x)$. L'ipotesi $x + r = y + r$ implica allora

$$\begin{aligned} \omega(x) &= x + d(x) \\ &= y + d(x) = y + d(y) + (d(x) - d(y)) \\ &= \omega(y) + (d(x) - d(y)) \end{aligned}$$

per cui $\omega(x) + d(y) = \omega(y) + d(x)$.

(3) $\implies$ (1): L'ipotesi (3) può essere riscritta nella forma

$$x + (d(x) + d(y)) = y + (d(y) + d(x))$$

e ciò implica $x \cong y$.

Osservazione 10.13. *Sia $d(x) \geq d(y)$. Allora sono equivalenti:*

(1) $x \cong y$.
(2) *$\Omega(x) = \Omega(y)$ e per ogni $k \in \mathbb{N}$ con $\omega(y) = \omega(x) + k$ vale, con $h := |\Omega(x)|$,*
$$k + d(x) - d(y) \in \mathbb{N}h.$$
(3) *Esiste un $k \in \mathbb{N}$ con $\omega(y) = \omega(x) + k$ tale che, con $h := |\Omega(x)|$,*
$$k + d(x) - d(y) \in \mathbb{N}h.$$

Dimostrazione. (1) $\implies$ (2): Sappiamo che $\Omega(x) = \Omega(y)$. Sia $k \in \mathbb{N}$ con $\omega(y) = \omega(x) + k$. Per il teorema 10.12 abbiamo

$$\begin{aligned} \omega(x) + d(y) &= \omega(y) + d(x) \\ &= \omega(x) + k + d(x) \\ &= \omega(x) + k + (d(x) - d(y)) + d(y) \end{aligned}$$

Per la iniettività di f su $\Omega(x)$ ciò implica

$$\omega(x) = \omega(x) + k + (d(x) - d(y))$$

e la proposizione 2.15 implica che

$$k + d(x) - d(y) \in \mathbb{N}h.$$

(2) $\implies$ (3): Chiaro, perché l'ipotesi $\Omega(x) = \Omega(y)$ implica che un k con $\omega(y) = \omega(x) + k$ esiste.

(3) $\implies$ (1): Sia $\omega(y) = \omega(x) + k$ con $k + d(x) - d(y) = jh$ per un $j \in \mathbb{N}$. Allora

$$\begin{aligned} \omega(y) + d(x) &= \omega(x) + k + d(x) \\ &= \omega(x) + jh + d(y) \\ &= \omega(x) + d(y) \end{aligned}$$

Per il teorema 10.12 $x \cong y$.

Osservazione 10.14. *h sia il periodo asintotico di x. Allora $x + h \cong x$.*

Dimostrazione. $x + h + d(x) = x + d(x)$.

Corollario 10.15. *Per ogni $x \in X$ esiste un unico $z \in \Omega(x)$ tale che $x \cong z$. Ogni classe di sincronia è quindi la classe di sincronia di un punto periodico. Se h è il periodo asintotico di x, per l'osservazione 10.14 è sufficiente scegliere $j \in \mathbb{N}$ tale che $jh \geq d(x)$. Allora $z = x + jh$. L'unicità segue dalla biiettività di f sulle orbite minimali.*

La classe di sincronia di z la troviamo percorrendo all'indietro a partire da z e a passi di h la componente $C(z)$ in tutti suoi rami.

Esercizio 10.16. Disegnare il grafo di un sistema dinamico finito e colorare i punti in modo tale che due punti hanno lo stesso colore se e solo se sono sincroni.

LABORATORIO DI PROGRAMMAZIONE

Corso di laurea in matematica | Anno accademico 2004/05 | Numero 4

L'insieme delle parti

Situazione 11.1. $X, Y, \ldots$ siano insiemi.
A partire dalla situazione 11.9 X ed Y saranno insiemi finiti.

Osservazione 11.2. *Si possono definire i numeri naturali induttivamente con:*

$$\begin{aligned} 0 &:= \emptyset \\ 1 &:= \{0\} = \{\emptyset\} \\ 2 &:= \{0,1\} = \{\emptyset, \{\emptyset\}\} \\ 3 &:= \{0,1,2\} \\ &\ldots \\ n+1 &:= \{0,1,\ldots,n\} \\ &\ldots \end{aligned}$$

Useremo spesso l'uguaglianza tra 2 *e* $\{0,1\}$. *Non è un'abbreviazione!*

Definizione 11.3. Denotiamo con Y^X l'insieme di tutte le funzioni $X \longrightarrow Y$.

Osservazione 11.4. *Se* X *ed* Y *sono finiti, allora* $|Y^X| = |Y|^{|X|}$.
In particolare $|2^X| = 2^{|X|}$.

Definizione 11.5. Per $f : X \longrightarrow Y$ ed $y \in Y$ sia

$$(f = y) := \{x \in X \mid f(x) = y\} = f^{-1}(y)$$

e similmente per $B \subset Y$

$$(f \in B) := f^{-1}(B)$$

Definizione 11.6. $\mathcal{P}(X) := \{A \mid A \subset X\}$ si chiama l'*insieme delle parti* di X.

Nota 11.7. Per ogni sottoinsieme $A \subset X$ possiamo definire una funzione $1_A \in 2^X$ ponendo

$$1_A(x) := \begin{cases} 1 & \text{se } x \in A \\ 0 & \text{se } x \notin A \end{cases}$$

1_A si chiama la *funzione caratteristica* di A; più precisamente si dovrebbe scrivere $1_{A,X}$. Dalla definizione segue

$$A = (1_A = 1)$$

Sia viceversa data una funzione $f \in 2^X$; allora f è univocamente determinata dall'insieme $(f = 1)$, poiché

$$f(x) = \begin{cases} 1 & \text{se } x \in (f = 1) \\ 0 & \text{se } x \notin (f = 1) \end{cases}$$

essendo 0 e 1 gli unici possibili valori di f. Quindi $f = 1_{(f=1)}$

Da ciò si deduce facilmente una biiezione naturale tra $\mathcal{P}(X)$ e 2^X, in cui un sottoinsieme A corrisponde alla funzione 1_A e una funzione f all'insieme $(f = 1)$.

$\mathcal{P}(X)$ e 2^X verranno spesso identificati.

Nota 11.8. Questa unificazione della notazione, precisata nell'articolo a lato sul prodotto cartesiano, porta a un'inconsistenza non più risolubile. Infatti, se n è un numero naturale, 2^n è da un lato un prodotto cartesiano n-plo, dall'altro lato vorremmo in questo modo denotare anche l'n-esima potenza aritmetica di 2. Ma queste due interpretazioni non coincidono; ad esempio aritmeticamente

$$2^2 = 4 = \{0,1,2,3\}$$

mentre come prodotto cartesiano

$$2^2 = \{(0,0),(0,1),(1,0),(1,1)\}$$

e, come si vede nella nota 11.12, le coppie $(0,0), \ldots$ sono oggetti molto più complicati dei numeri $0, 1, 2, 3$.

Dobbiamo quindi derivare dal contesto la corretta interpretazione di espressioni della forma 2^n. Un esempio è già la nota 11.11.

Funzioni booleane

Situazione 11.9. X sia un insieme finito con n elementi, $Y, Z, \ldots$ altri insiemi finiti.

Definizione 11.10. Una *funzione booleana* (semplice e espressa sopra X) è una funzione $\alpha : 2^X \longrightarrow 2$.

α è quindi un elemento di 2^{2^X}. Identificando 2^{2^X} con $\mathcal{P}(\mathcal{P}(X))$, vediamo che α non è altro che un insieme di sottoinsiemi di X, cioè un insieme $\alpha \subset \mathcal{P}(X)$. Useremo spesso questa interpretazione.

È chiaro che per $|X| = |Z|$ la struttura di 2^{2^X} è isomorfa a quella di 2^{2^Z}, perciò è la cardinalità n che conta.

In particolare potremmo anche assumere che $X = n$; siccome gli elementi di 2^n sono n-ple di elementi di 2, si dice anche che α è una funzione booleana di n variabili.

Più tardi considereremo anche funzioni $2^X \longrightarrow 2^Y$; queste funzioni sono dette *funzioni booleane multiple*.

Nota 11.11. Quante funzioni booleane ci sono? Se l'insieme X possiede n elementi, allora $|2^{2^X}| = 2^{2^n}$, dove l'espressione a destra è da interpretare numericamente (cfr. nota 11.8), quindi il numero delle funzioni booleane cresce nel modo seguente con il crescere di n:

n	$\lvert 2^{2^X}\rvert$
0	$2^1 = 2$
1	$2^2 = 4$
2	$2^4 = 16$
3	$2^8 = 256$
4	$2^{16} = 65536$
5	$2^{32} = 4294676296$
6	2^{64}
7	2^{128}

Si vede che ogni volta che si aumenta la cardinalità di X di uno, il numero delle funzioni booleane espresse sopra X diventa il quadrato di quello precedente. Esistono quindi moltissime funzioni booleane, già per 5 variabili sono più di 4 miliardi, e per 6 variabili diventano 18 miliardi di miliardi.

In questo numero

Il prodotto cartesiano

Nota 11.12. Una funzione è spesso definita come tripla, i cui componenti sono il dominio, il codominio e il grafico della funzione. Ciò richiede che abbiamo definito in precedenza il concetto di tripla.

D'altra parte vorremmo poter considerare coppie e triple come funzioni, vorremmo cioè che $X \times Y$ sia l'insieme di tutte le funzioni $f : 2 \longrightarrow X \cup Y$ tali che $f(0) \in X$ e $f(1) \in Y$ e che $X \times Y \times Z$ sia l'insieme di tutte le funzioni $f : 3 \longrightarrow X \cup Y \cup Z$ tali che in più $f(2) \in Z$. In questo caso poi avremmo automaticamente uguaglianze $X \times X = X^2$ e $X \times X \times X = X^3$ che sono in accordo con la definizione 11.3.

Se utilizziamo però le triple per definire le funzioni, ci troviamo in un circolo vizioso.

Ne possiamo venir fuori utilizzando nella definizione di funzione un altro concetto di tripla che solo, come si dimostra con un po' di pazienza, garantisce che una tripla sia univocamente determinata dai suoi componenti. Definiamo quindi per tre oggetti matematici a, b, c la *coppia ausiliaria* o *logica* $\langle a, b\rangle := \{a, \{a, b\}\}$ e la *tripla ausiliaria* o *logica* $\langle a, b, c\rangle := \langle\langle a, b\rangle, c\rangle$.

A questo punto possiamo riformulare la definizione di funzione, sostituendo il termine *tripla* con *tripla logica* e il termine *coppia* che appare nella definizione del grafico con *coppia logica* e definire i prodotti cartesiani come sopra. Elementi di $X \times Y$ saranno detti *coppie* e scritti nella forma (x, y) - ma in verità dovremmo addirittura scrivere $(x, y)_{X \times Y}$ - e similmente per le triple. La coppia (x, y) è quindi quella funzione $f : 2 \longrightarrow X \cup Y$ per cui $f(0) = x, f(1) = y$.

Formalmente questa coppia è un oggetto piuttosto complicato:

$$(x, y) = \langle 2, X \cup Y, \{\langle 0, x\rangle, \langle 1, y\rangle\}\rangle$$

Se $X = Y$, la coppia $(x, y) \in X^2$ è data da

$$(x, y) = \langle 2, X, \{\langle 0, x\rangle, \langle 1, y\rangle\}\rangle$$

cosicché ad esempio

$$2^2 = \{(0,0),(0,1),(1,0),(1,1)\}$$

con

$$(0, 0) = \langle 2, 2, \{\langle 0, 0\rangle, \langle 1, 0\rangle\}\rangle$$

ecc. Vediamo che il concetto di coppia è molto profondo è difficile. E i problemi non sono finiti, come si vede dalla nota 11.8.

Vita è codifica

Abbiamo visto che esistono più di 18 miliardi di miliardi di funzioni booleane di 6 variabili; ma gli ingegneri studiano funzioni booleane con 30 o anche 100 variabili. Di queste ultime ne esistono $2^{2^{100}}$, un numero che la matematica ci permette di scrivere, mentre non siamo in nessun modo in grado di elencare o anche solo lontanamente immaginare tutte queste funzioni, perché, se pensiamo che $N := 2^{100}$ è probabilmente superiore al numero di oggetti nell'universo, di quelle funzioni ne esistono addirittura 2^N. Il mondo delle funzioni booleane è quindi di una incredibile ricchezza, è un mondo di strumenti intellettuali a cui possiamo attingere per descrivere praticamente tutti gli oggetti, fenomeni e leggi del mondo reale. Per questo la teoria delle funzioni booleane è un mondo di ricerca intensissimo, non solo da parte degli ingegneri che cercano di studiare e semplificare i circuiti digitali, ma anche ad esempio, in modo sempre crescente, nella medicina, dove si cerca di descrivere i meccanismi biochimici e patologici con modelli booleani.

D'altra parte la vita stessa ne è un esempio. Abbiamo l'impressione che un organismo è tanto più vitale quanto più è ramificato. Troviamo strutture ramificate negli alberi, nell'organizzazione del sistema vasocircolatorio, dei polmoni e del sistema nervoso. Persino quando in un oggetto inanimato o su un pianeta lontano scopriamo sistemi ramificati questo ci dà una sensazione di vita anche quando forse è solo una somiglianza, un ricordo o una struttura potenziale.

Quindi la ramificazione organizzata è una delle basi della vita. Anche un programma in un linguaggio di programmazione comincia ad acquistare vita solo quando ci sono ramificazioni, espresse da istruzioni `if ... else`. La vita sul nostro pianeta è organizzata da un codice, il codice genetico, e, in maniera più nascosta e molto più complessa, da sistemi booleani che descrivono le regole della vita o del particolare organismo che in un certo modo è definito da quelle regole. Senza questa codifica la vita non avrebbe quella forza interna, quella pertinacia e quella capacità di rigenerarsi e di ottimizzarsi che la fa apparire superiore al mondo puramente fisico, e nemmeno quella capacità interattiva che sta alla base ad esempio dei fenomeni sociali.

Quindi la vita è codifica.

Sistemi di insiemi

Abbiamo osservato nella definizione 11.10 che una funzione booleana è la stessa cosa come un insieme di sottoinsiemi di un insieme finito, un *sistema di insiemi*. Come tali appaiono nella combinatoria (dove spesso i sistemi di insiemi vengono chiamati *ipergrafi*) e in molti altri campi della matematica e spesso la sola traduzione di enunciati da un linguaggio all'altro porta a interpretazioni sorprendenti o utili. Ad esempio una topologia su un insieme X può essere definita indicando l'insieme α degli aperti; se X è finito, abbiamo una funzione booleana. Le funzioni booleane monotone (o piuttosto le funzioni antitone) possono essere identificate con i *complessi simpliciali*. Nell'algebra universale si studiano *sistemi di chiusi*, anch'essi non sono altro che sistemi di insiemi con particolari proprietà.

Spazi topologici finiti

Nota 12.1. In uno spazio topologico X due intorni di un punto x si intersecano sempre in un altro intorno di quel punto; per induzione segue immediatamente che un numero finito di intorni di x è ancora un intorno di x. Se lo spazio stesso è finito, l'intersezione U_x di tutti gli intorni di x è un intorno di x, necessariamente il più piccolo - infatti gli intorni di x sono esattamenti quei sottoinsiemi dello spazio che contengono U_x. Usando una notazione che introdurremo fra breve ciò significa che l'insieme degli intorni di x coincide con U_x^+.

Sempre nell'ipotesi che X sia finito, introduciamo su X una relazione $\leq$ ponendo

$$x \leq y : \iff U_y \subset U_x$$

È chiaro che ciò accade se e solo se $y \in U_x$ e che questa relazione è riflessiva e transitiva e costituisce quindi un quasiordine su X; inoltre (essendo $U_x^+ = U_y^+$ se e solo se $U_x = U_y$) la relazione $\leq$ è un ordine parziale (cioè anche antisimmetrica) se e solo se lo spazio topologico X soddisfa l'assioma di separazione T_0 che chiede che due punti che hanno gli stessi intorni sono uguali.

Sia viceversa dato un quasiordine $\leq$ su un insieme finito X. Se definiamo $U_x := \{y \in X \mid y \geq x\}$ e come intorni di x prendiamo tutti i sottoinsiemi di X che contengono U_x, otteniamo uno spazio topologico. Si dimostra adesso, con un po' di pazienza, che le due costruzioni sono una l'inversa dell'altra e che gli omomorfismi di insiemi quasiordinati, cioè le funzioni monotone, sono esattamente le funzioni continue nell'interpretazione topologica.

Troviamo così che spazi topologici finiti e insiemi quasiordinati finiti sono, a tutti gli effetti, la stessa cosa. Gli insiemi quasiordinati finiti sono a loro volta uno dei concetti più fondamentali della combinatoria (comprendono ad esempio i *reticoli*, ma anche strutture molto più generali).

Nota 12.2. Sia X un insieme finito. Un insieme di insiemi $\alpha \subset \mathcal{P}(X)$, cioè una funzione booleana espressa su X, coincide, come si impara nei corsi di topologia, con l'insieme degli aperti di una topologia su X se e solo se gode delle seguenti proprietà:

(1) $\emptyset \in \alpha$.
(2) $X \in \alpha$.
(3) $U, V \in \alpha \Longrightarrow U \cap V \in \alpha$.
(4) $U, V \in \alpha \Longrightarrow U \cup V \in \alpha$.

Attenzione: L'assioma (4) può essere formulato in questo modo solo se X è finito; nel caso generale bisogna chiedere che un'unione arbitraria di aperti sia ancora un aperto.

Vediamo quindi che la teoria delle funzioni booleane contiene (come una piccola parte!) la teoria degli spazi topologici finiti che, come abbiamo appena visto, non è banale ma di grande importanza.

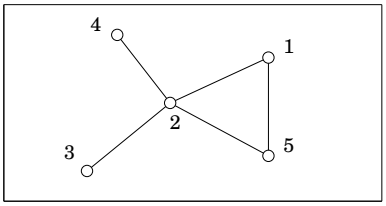
Se si rinuncia all'assioma (3) o all'assioma (4) si ottiene una teoria ancora più generale, quella dei *sistemi di aperti* o quella, equivalente (per dualità) e più spesso usata in molti contesti dell'algebra universale e della combinatoria, dei *sistemi di chiusi*. Queste teorie sono quindi, per insiemi finiti, anch'esse contenute nella teoria delle funzioni booleane. Oltre che nella matematica pura vengono da una quindicina di anni applicate ad esempio a problemi di classificazione in sociologia, biochimica e virologia.

Grafi

Un ipergrafo α tale che $|A| = 2$ per ogni $A \in \alpha$ si chiama un *grafo* (semplice). Ogni tale α dà luogo a un grafo nel senso comune come nell'esempio seguente:

$$X = \{1, 2, 3, 4, 5\}$$
$$\alpha = \{\{1, 2\}, \{1, 5\}, \{2, 3\}, \{2, 4\}, \{2, 5\}\}$$

Adesso congiungiamo x ed y con una linea del grafo se e solo se $\{x, y\} \in \alpha$:

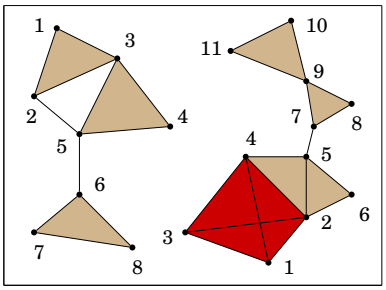


Quindi anche la teoria dei grafi fa parte della teoria delle funzioni booleane.

Complessi simpliciali

Uno dei concetti più importanti della *topologia combinatoria* o *geometrica* è il concetto di complesso simpliciale. Un *complesso simpliciale* è un sistema di insiemi α (cioè una funzione booleana) con la proprietà che ogni insieme contenuto in un elemento di α appartenga esso stesso ad α. L'importanza dei complessi simpliciali risiede nel fatto che mediante essi si possono descrivere le proprietà omotopiche di spazi topologici geometrici; per il loro studio è stata sviluppata una straordinariamente ricca teoria algebrica, detta *algebra omologica*; per questa ragione la teoria dei complessi simpliciali e le sue generalizzazioni fanno parte della *topologia algebrica*. In un certo senso quindi anche questo ramo molto sofisticato della matematica pura può essere inquadrato nella teoria delle funzioni booleane. Viceversa i risultati della topologia algebrica possono spesso essere applicati a problemi concreti o ingegneristici.

I complessi simpliciali possono essere rappresentati da figure in $\mathbb{R}^n$ come nei seguenti esempi:



Alla figura piana a sinistra possiamo associare il complesso simpliciale α che consiste degli insiemi $\{1, 2, 3\}$, $\{3, 4, 5\}$, $\{2, 5\}$, $\{5, 6\}$, $\{6, 7, 8\}$ e di tutti i loro sottoinsiemi, alla figura 3-dimensionale a destra, in cui il tetraedro con vertici 1,2,3,4 è pieno, il complesso simpliciale che consiste degli insiemi $\{1, 2, 3, 4\}$, $\{2, 4, 5\}$, $\{2, 5, 6\}$, $\{5, 7\}$, $\{7, 8, 9\}$, $\{9, 10, 11\}$ e di tutti i loro sottoinsiemi. Non è molto difficile dimostrare che ogni complesso simpliciale α può essere ottenuto da una figura come in questi esempi (detta talvolta *poliedro simpliciale*) in $\mathbb{R}^{2d+1}$, se $|A| \leq d + 1$ per ogni $A \in \alpha$.

Funzioni booleane monotone

Nota 13.1. Identifichiamo da ora in avanti sistematicamente le funzioni $\alpha : 2^X \longrightarrow 2$ con i sistemi di insiemi $\alpha \subset \mathcal{P}(X)$, usando la stessa lettera per una funzione booleana e il sistema di insiemi ad essa corrispondente. Quindi ad esempio

$$A \in \alpha \iff \alpha(A) = 1$$

Definizione 13.2. L'insieme $2 = \{0, 1\}$ è un insieme parzialmente ordinato con

$$\leq = \{(0,0), (0,1), (1,1)\}$$

Ciò corrisponde, nell'identificazione tra 2 e $\mathcal{P}(1)$, all'inclusione insiemistica in $\mathcal{P}(1)$. Si noti che 2 (come insieme), con le operazioni di addizione e moltiplicazione modulo 2 (come numero!) è anche un campo, ma non è un campo ordinato, perché in un campo ordinato $a, b > 0$ deve implicare $a + b > 0$, mentre in 2 si ha $1 > 0$, ma $1 + 1 = 0$.

Più in generale l'inclusione insiemistica in $\mathcal{P}(X)$ corrisponde a un ordine parziale altrettanto naturale su 2^X dato da

$$f \leq g \iff f(x) \leq g(x) \text{ per ogni } x \in X$$

Se $A, B \subset X$, abbiamo allora

$$A \subset B \iff 1_A \leq 1_B$$

Se $X = n$, gli elementi di 2^X possono anche essere scritti come n-ple $(u_1, \ldots, u_n)$ e allora abbiamo $(u_1, \ldots, u_n) \leq (v_1, \ldots, v_n)$ se e solo se $u_i \leq v_i$ per ogni $i = 1, \ldots, n$.

Definizione 13.3. Una funzione booleana $\alpha : 2^X \longrightarrow 2$ si dice *monotona* se

$$A \subset B \Longrightarrow \alpha(A) \leq \alpha(B)$$

e *antitona* se

$$A \subset B \Longrightarrow \alpha(A) \geq \alpha(B)$$

È chiaro che α è monotona se e solo se

$$A \in \alpha \text{ e } A \subset B \Longrightarrow B \in \alpha$$

quindi se e solo se α contiene con ogni insieme anche tutti gli insiemi che contengono quell'insieme, e che α è antitona se e solo se

$$A \subset B \in \alpha \Longrightarrow A \in \alpha$$

cioè se solo se α è un complesso simpliciale!

Definizione 13.4. Per $\alpha \subset \mathcal{P}(X)$ sia

$$\neg\alpha := \mathcal{P}(X) \setminus \alpha$$

In altre parole,

$$A \in \neg\alpha \iff A \notin \alpha$$

oppure, nell'interpretazione funzionale,

$$\neg\alpha(A) = 1 \iff \alpha(A) = 0$$
$$\neg\alpha(A) = 0 \iff \alpha(A) = 1$$

È chiaro che $\neg\neg\alpha = \alpha$.

Osservazione 13.5. *Sia $\alpha \subset \mathcal{P}(X)$. Allora α è antitona (e quindi un complesso simpliciale) se e solo se $\neg\alpha$ è monotona.*

Dimostrazione. Sia $A \subset B$ e $A \in \neg\alpha$, cioè $A \notin \alpha$. Se si avesse $B \notin \neg\alpha$, si avrebbe $B \in \alpha$ e quindi $A \in \alpha$, perché α è antitona, una contraddizione. Nello stesso modo o per simmetria (sfruttando $\neg\neg\alpha = \alpha$) si ottiene l'altra direzione dell'implicazione.

Intervalli e cointervalli in $\mathcal{P}(X)$

Definizione 13.6. Per $\alpha, \beta \subset \mathcal{P}(X)$ scriviamo spesso $\alpha\beta$ invece di $\alpha \cap \beta$. Questa non è solo un'abbreviazione, ma corrisponde al fatto che nell'interpretazione funzionale all'intersezione corrisponde la moltiplicazione nell'anello 2^{2^X}.

Definizione 13.7. Per $P, R \subset X$ siano

$$P^+ := \{A \subset X \mid P \subset A\}$$
$$R^- := \{A \subset X \mid A \subset X \setminus R\}$$

Abbiamo usato la prima di queste notazioni già nella nota 12.1.

P^+ è l'insieme dei sottoinsiemi di X che contengono P, mentre

$$P^+R^- = \{A \subset X \mid P \subset A \subset X \setminus R\}$$

è l'intervallo $[P, X \setminus R]$ nell'insieme parzialmente ordinato $\mathcal{P}(X)$ determinato da P ed $X \setminus R$; cfr. nota 13.9.

Gli intervalli di $\mathcal{P}(X)$ in logica sono anche detti *termini booleani* o *monomi*.

Un intervallo che contiene esattamente un elemento è detto *semplice*.

Nota 13.8. Per $R \subset X$

$$\begin{aligned} R^- &= \{A \subset X \mid A \subset X \setminus R\} \\ &= \{A \subset X \mid R \subset X \setminus A\} \\ &= \{A \subset X \mid A \cap R = \emptyset\} \end{aligned}$$

R^- è quindi l'insieme dei sottoinsiemi di X che hanno intersezione vuota con R.
Vediamo anche che, per $A \subset X$,

$$A \in R^- \iff X \setminus A \in R^+$$

Nota 13.9. a, b, c, d siano elementi di un insieme parzialmente ordinato. Denotiamo con $[a, b]$ e $[c, d]$ gli intervalli determinati da questi elementi. Allora:

(1) $[a, b] = [c, d] \neq \emptyset \Longrightarrow a = c, b = d$.
(2) $|[a, b]| = 1 \iff a = b$.

Si noti che l'enunciato (1) vale solo per intervalli non vuoti; infatti affinche $[a, b] = \emptyset$, è sufficiente che $a \not\leq b$.

Dimostrazione. (1) In primo luogo abbiamo $a \leq b$ e $c \leq d$, altrimenti l'intervallo sarebbe vuoto.

L'uguaglianza implica che $a, b \in [c, d]$, e quindi abbiamo $c \leq a \leq b \leq d$; ma per simmetria anche $a \leq c \leq d \leq b$. Siccome abbiamo presupposto che $\leq$ sia un ordine parziale, da ciò seguono $a = c$ e $b = d$.

(2) Chiaro; bisogna però usare anche qui l'ipotesi che si tratti di un ordine parziale.

Corollario 13.10. *Siano $P, Q, R, S \subset X$ tali che $P^+R^- = Q^+S^- \neq \emptyset$.*

Allora $P = Q$, $R = S$.

Osservazione 13.11. *Sia $A \subset X$. Allora*

$$\{A\} = [A, A] = A^+(X \setminus A)^-$$

Corollario 13.12. *Per una funzione booleana $\sigma \subset \mathcal{P}(X)$ sono equivalenti:*

(1) σ è un intervallo semplice.
(2) σ è della forma $A^+(X \setminus A)^-$ per un sottoinsieme $A \subset X$.

In questo caso $\sigma = \{A\}$.

Definizione 13.13. Un *cointervallo* in un insieme parzialmente ordinato è il complemento di un intervallo.

Un cointervallo si dice *semplice*, se è complemento di un intervallo semplice; un cointervallo è quindi semplice se e solo se consiste di tutti gli elementi dell'insieme parzialmente ordinato tranne uno.

In logica i cointervalli di $\mathcal{P}(X)$ prendono il nome di *clausole booleane*.

Nota 13.14. Per $P, R \subset X$ abbiamo:

(1) $\neg P^+ = \{A \subset X \mid P \not\subset A\}$.
(2) $\neg R^- = \{A \subset X \mid A \cap R \neq \emptyset\}$.
(3) $\neg(P^+R^-) = \neg P^+ \cup \neg R^-$.

I cointervalli di $\mathcal{P}(X)$ coincidono quindi con gli insiemi della forma $\neg P^+ \cup \neg R^-$ con $P, R \subset X$.

Dimostrazione. (1) Per definizione di P^+.
(2) Nota 13.8.
(3) Chiaro.

Corollario 13.15. *Per una funzione booleana $\tau \subset \mathcal{P}(X)$ sono equivalenti:*

(1) τ è un cointervallo semplice.
(2) τ è della forma $\neg B^+ \cup \neg(X \setminus B)^-$ per un sottoinsieme $B \subset X$.

In questo caso $\tau = \neg\{B\}$.

Osservazione 13.16. *Siano P, Q, R, S sottoinsiemi di X. Allora*

$$P^+Q^+ = (P \cup Q)^+$$
$$R^-S^- = (R \cup S)^-$$

Dimostrazione. Chiaro.

Proposizione 13.17. *Per $x \in X$ si ha*

$$x^- = \neg x^+$$

Dimostrazione. Immediata.

Esercizio 13.18. Sia $P \subset X$. Allora:

$$P \in P^+$$
$$\emptyset \in P^-$$

Esercizio 13.19. Abbiamo

$$X^+ = \{X\}$$
$$X^- = \{\emptyset\}$$
$$\emptyset^+ = \mathcal{P}(X)$$
$$\emptyset^- = \mathcal{P}(X)$$

Esercizio 13.20. Sia $P \subset X$. Allora:

$$\neg P^+ = P^- \iff |P| = 1$$

Esercizio 13.21. Siano $P, R \subset X$. Allora:

$$P^+X^- = \begin{cases} \{\emptyset\} & \text{per } P = \emptyset \\ \emptyset & \text{altrimenti} \end{cases}$$

$$P^+\emptyset^- = P^+$$

$$X^+R^- = \begin{cases} \{X\} & \text{per } R = \emptyset \\ \emptyset & \text{altrimenti} \end{cases}$$

$$\emptyset^+R^- = R^-$$

Forma normale disgiuntiva

Osservazione 14.1. *Siano dati* $x_1, \dots, x_p, y_1, \dots, y_r \in X$. *Allora*

$$x_1^+ \cdots x_p^+ y_1^- \cdots y_r^-$$

è l'insieme di quei sottoinsiemi di X *che contengono tutti i punti* $x_1, \dots, x_p$ *e nessuno dei punti* $y_1, \dots, y_r$. *Inoltre*

$$\neg(x_1^+ \cdots x_p^+ y_1^- \cdots y_r^-) = x_1^- \cup \ldots \cup x_p^- \cup y_1^+ \cup \ldots \cup y_r^+$$

Dimostrazione. Il primo enunciato deriva direttamente dalla definizione 13.7, nel secondo usiamo la proposizione 13.17.

Osservazione 14.2. *Sia* $\alpha \subset \mathcal{P}(X)$. *Allora*

$$\alpha = \bigcup_{A \in \alpha} \{A\} = \bigcup_{A \in \alpha} A^+ (X \setminus A)^-$$

Nota 14.3. Sia $A \subset X$. Possiamo scrivere A nella forma

$$A = \{a_1, \dots, a_m\}$$

con gli a_j tutti distinti. Allora

$$X \setminus A = \{u_1, \dots, u_{n-m}\}$$

con gli a_j e u_k tutti distinti perché $|X| = n$. Per l'osservazione 13.11 abbiamo allora

$$\{A\} = A^+ (X \setminus A)^- = a_1^+ \cdots a_m^+ u_1^- \cdots u_{n-m}^-$$

In questa rappresentazione ogni elemento di X appare esattamente una volta: con esponente positivo, se appartiene ad A, con esponente negativo in caso contrario.

Nota 14.4. Riformulando l'osservazione 14.2 con l'aiuto della nota 14.3, vediamo che una funzione booleana $\alpha \subset \mathcal{P}(X)$ può essere rappresentata come unione (compresa l'unione vuota nel caso che $\alpha = \emptyset$) di termini della forma $a_1^+ \cdots a_m^+ u_1^- \cdots u_{n-m}^-$.

Questa rappresentazione viene chiamata la *forma normale disgiuntiva* (FND) di α. Essa consiste in pratica semplicemente di un elenco degli elementi A di α, in cui ogni volta si elencano sia gli elementi di A che quelli del complemento $X \setminus A$ e si dimostra facilmente che è unica a parte ovvie possibilità di cambi nell'ordine in cui quegli elementi vengono elencati.

Esempio 14.5. Siano $X = \{x_1, x_2, x_3\}$ un insieme con tre elementi ed $\alpha \subset \mathcal{P}(X)$ data dalla seguente tabella, in cui rappresentiamo in modo naturale l'insieme $\{x_2\}$ mediante il vettore riga 010 ecc.

x_1	x_2	x_3	α
0	0	0	1
0	0	1	0
0	1	0	1
1	0	0	0
0	1	1	1
1	0	1	1
1	1	0	0
1	1	1	0

La forma normale disgiuntiva di α è

$$x_1^- x_2^- x_3^- \cup x_1^- x_2^+ x_3^- \cup x_1^- x_2^+ x_3^+ \cup x_1^+ x_2^- x_3^+$$

Forma normale congiuntiva

Nota 14.6. Nella forma normale disgiuntiva ogni funzione booleana è rappresentata come unione di intervalli semplici. Vogliamo adesso dimostrare che ogni funzione booleana può essere anche rappresentata come intersezione di cointervalli semplici: questa è la *forma normale congiuntiva*.

Anche stavolta la dimostrazione è molto facile e duale a quella precedente. Infatti le due forme normali sono in un certo senso *rappresentazioni tautologiche*: la forma normale disgiuntiva non è altro che un elenco degli insiemi che sono elementi della funzione booleana α data, la forma normale congiuntiva corrisponde a un elenco degli insiemi che non fanno parte di α.

Osservazione 14.7. *Sia* $\alpha \subset \mathcal{P}(X)$. *Allora*

$$\neg\alpha = \bigcup_{B \notin \alpha} \{B\}$$

e quindi

$$\alpha = \bigcap_{B \notin \alpha} \neg\{B\}$$

Nota 14.8. Sia $B \subset X$. Possiamo scrivere B nella forma

$$B = \{b_1, \dots, b_m\}$$

con i b_j tutti distinti. Allora

$$X \setminus B = \{v_1, \dots, v_{n-m}\}$$

con i b_j e v_k tutti distinti. Come nella nota 14.3 abbiamo allora

$$\{B\} = b_1^+ \cdots b_m^+ v_1^- \cdots v_{n-m}^-$$

cosicché, per l'osservazione 14.1,

$$\neg\{B\} = b_1^- \cup \ldots \cup b_m^- \cup v_1^+ \cup \ldots \cup v_{n-m}^+$$

Anche qui ogni elemento di X appare esattamente una volta e con esponente positivo se e solo se appartiene a B.

Nota 14.9. Riformulando l'osservazione 14.7 con l'aiuto della nota 14.8, vediamo che una funzione booleana $\alpha \subset \mathcal{P}(X)$ può essere rappresentata come intersezione (compresa l'intersezione vuota nel caso che α coincida con $\mathcal{P}(X)$) di termini della forma

$$b_1^- \cup \ldots \cup b_m^- \cup v_1^+ \cup \ldots \cup v_{n-m}^+$$

Questa rappresentazione viene chiamata la *forma normale congiuntiva* (FNC) di α. Essa non è altro che la negazione della FND di $\neg\alpha$, consiste quindi semplicemente di un elenco degli elementi di $\neg\alpha$ a cui poi si applica l'osservazione 14.1.

Esempio 14.10. Nell'esempio 14.5 la forma normale disgiuntiva di $\neg\alpha$ è

$$x_1^- x_2^- x_3^+ \cup x_1^+ x_2^- x_3^- \cup x_1^+ x_2^+ x_3^- \cup x_1^+ x_2^+ x_3^+$$

la forma normale congiuntiva di α è perciò

$$\alpha = (x_1^+ \cup x_2^+ \cup x_3^-)(x_1^- \cup x_2^+ \cup x_3^+)(x_1^- \cup x_2^- \cup x_3^+)(x_1^- \cup x_2^- \cup x_3^-)$$

Nota 14.11. Siano $\alpha \subset \mathcal{P}(X)$ ed $x \in X$. Le seguenti formule (che seguono dalla proposizione 13.17) vengono spesso usate nella semplificazone di funzioni booleane:

$$x^+ \cup x^- = \mathcal{P}(X)$$
$$x^+ \alpha \cup x^- \alpha = \alpha$$

Le 16 funzioni booleane binarie

Nota 14.12. Quando $\alpha = \emptyset$, la FND di α è vuota. Nell'interpretazione funzionale $\emptyset$ corrisponde alla funzione costante 0; analogamente il caso $\alpha = \mathcal{P}(X)$ corrisponde alla funzione costante 1. Useremo queste e simili abbreviazioni nelle seguenti tabelle insieme alle formule insiemistiche.

Nota 14.13. Elenchiamo prima le 4 funzioni booleane unarie, in cui quindi $X = \{x\}$.

x	0 $\emptyset$	1 $\mathcal{P}(X)$	id x^+	NOT x^-
0	0	1	0	1
1	0	1	1	0

Vediamo che le prime due sono costanti (per ogni $n \geq 0$ esistono esattamente due funzioni costanti con i valori 0 e 1 che, come abbiamo visto, corrispondono a $\emptyset$ e $\mathcal{P}(X)$), la terza è l'identità, mentre l'unica funzione unaria non banale è la *negazione*.

Nota 14.14. Sono invece, come sappiamo, 16 le funzioni booleane binarie, in cui cioè $X = \{x, y\}$ con $x \neq y$.

x	y	0 $\emptyset$	1 $\mathcal{P}(X)$	p_x x^+	p_y y^+
0	0	0	1	0	0
0	1	0	1	0	1
1	0	0	1	1	0
1	1	0	1	1	1

Con p_x e p_y abbiamo denotato le proiezioni su x e y.

x	y	n_x x^-	n_y y^-	$=$ $x^- y^- \cup x^+ y^+$	$\neq$ (XOR) $x^+ y^- \cup x^- y^+$
0	0	1	1	1	0
0	1	1	0	0	1
1	0	0	1	0	1
1	1	0	0	1	0

n_x e n_y sono le proiezioni negate.

x	y	$\leq$ $x^- \cup y +$	$<$ $x^- y^+$	$\geq$ $x^+ \cup y^-$	$>$ $x^+ y^-$
0	0	1	0	1	0
0	1	1	1	0	0
1	0	0	0	1	1
1	1	1	0	1	0

Si osservi che nelle rappresentazioni insiemistiche stavolta abbiamo potuto usare semplificazioni che sono diverse dalle FND.

x	y	AND $x^+ y^+$	OR $x^+ \cup y^+$	NAND $x^- \cup y^-$	NOR $x^- y^-$
0	0	0	0	1	1
0	1	0	1	1	0
1	0	0	1	1	0
1	1	1	1	0	0

Abbiamo usato alcune abbreviazioni comuni della teoria dei circuiti: XOR (exclusive or), AND, OR, NAND, NOR.

LABORATORIO DI PROGRAMMAZIONE

Corso di laurea in matematica | Anno accademico 2004/05 | Numero 5

Preautomi cellulari

Situazione 15.1. Quando non indicato diversamente, K, N, E sono insiemi finiti.

Definizione 15.2. Un *preautoma cellulare* (PAC) è un'applicazione $K^N \times N \xrightarrow{\varepsilon} E$, dove K, N, E sono insiemi finiti (univocamente determinati da ε). Gli elementi di N si chiamano *nodi* del PAC, gli elementi di E *effetti*, gli elementi di K^N *stati*. K si chiama l'insieme dei *coefficienti* o valori del PAC. Il PAC si dice *binario*, se $K = 2$.

Per $x \in K^N$ ed $a \in N$ interpretiamo $\varepsilon(x, a)$ come l'effetto dello stato x sul nodo a. Scriviamo spesso x_a invece di $x(a)$.

Definizione 15.3. Un PAC *valutato* (PACV) è un diagramma

$$K^N \times N \xrightarrow{\varepsilon} E \xrightarrow{\alpha} K$$

dove $K^N \times N \xrightarrow{\varepsilon} E$ è un PAC e $E \xrightarrow{\alpha} K$ un'applicazione. $\alpha(e)$ è il *valore* dell'effetto e nel PACV; α si chiama la *valutazione* del PAC.

Definizione 15.4. Due PACV

$$K^N \times N \xrightarrow{\varepsilon} E \xrightarrow{\alpha} K$$

e

$$K^N \times N \xrightarrow{\delta} F \xrightarrow{\beta} K$$

si chiamano *equivalenti*, se il diagramma

$$\begin{array}{ccc} K^N \times N & \xrightarrow{\varepsilon} & E \\ {\scriptstyle\delta}\downarrow & & \downarrow{\scriptstyle\alpha} \\ F & \xrightarrow{\beta} & K \end{array}$$

è commutativo.

Automi cellulari

Definizione 15.5. Un *automa cellulare* (AC) è un'applicazione $K^N \longrightarrow K^N$, cioè un sistema dinamico finito sullo spazio K^N. Gli elementi di K^N si chiamano *stati* del sistema, gli elementi di N *nodi*, gli elementi di K *coefficienti*.

Nota 15.6. $K^N \times N \xrightarrow{\tau} K$ sia un'applicazione. Allora otteniamo un AC $K^N \xrightarrow{\tau^\#} K^N$ ponendo

$$\tau^\#(x)_a := \tau(x, a)$$

per $x \in K^N$ ed $a \in N$. In questo modo da ogni valutazione $E \xrightarrow{\alpha} K$ di un PAC $K^N \times N \xrightarrow{\varepsilon} E$ otteniamo un AC $K^N \xrightarrow{f} K^N$ con $f := (\alpha \circ \varepsilon)^\#$, in cui quindi

$$f(x)_a = \alpha(\varepsilon(x, a))$$

per ogni $x \in K^N$ ed $a \in N$.

Talvolta, quando per un PAC fissato ε consideriamo più valutazioni α, scriviamo f_α invece di f.

Segue direttamente dalla definizione che PACV equivalenti inducono lo stesso AC.

Nota 15.7. È chiaro viceversa che ogni AC può essere ottenuto da un PAC (naturalmente non univocamente determinato) nel modo descritto nella nota 15.6. Infatti in primo luogo ogni AC $K^N \xrightarrow{f} K^N$ induce un'applicazione $K^N \times N \xrightarrow{\tau} K$, data da $\tau(x, a) := f(x)_a$, che possiamo considerare come un PAC. È chiaro che ciò implica $f = \tau^\#$, se usiamo la notazione introdotta prima. Con la valutazione identica $K \xrightarrow{\mathrm{id}} K$ otteniamo un PACV $K^N \times N \xrightarrow{\tau} K \xrightarrow{\mathrm{id}} K$ da cui si ottiene l'AC di partenza.

Esempio 15.8. $N := \{a, b, c\}$ consista di tre elementi distinti che ci immaginiamo connessi in un cerchio orientato in cui, per uno stato x nel sistema dinamico che vogliamo costruire, il valore $(x + 1)_p$ in un nodo p dipende solo da x_p e $x_{p'}$, dove p' è il predecessore di p in questa configurazione circolare. Possiamo quindi definire un PAC con $K^N \times N \xrightarrow{\varepsilon} E$, dove $E = K \times K$ e

$$\begin{aligned} \varepsilon(x, a) &:= (x_c, x_a) \\ \varepsilon(x, b) &:= (x_a, x_b) \\ \varepsilon(x, c) &:= (x_b, x_c) \end{aligned}$$

Per $K = 2$ possiamo descrivere il PAC mediante una tabella

x_a	x_b	x_c	$\varepsilon(x,a)$		$\varepsilon(x,b)$		$\varepsilon(x,c)$	
0	0	0	0	0	0	0	0	0
0	0	1	1	0	0	0	0	1
0	1	0	0	0	0	1	1	0
1	0	0	0	1	1	0	0	0
0	1	1	1	0	0	1	1	1
1	0	1	1	1	1	0	0	1
1	1	0	0	1	1	1	1	0
1	1	1	1	1	1	1	1	1

A questo punto da ogni valutazione $2^2 \xrightarrow{\alpha} 2$, cioè da ogni funzione booleana (FB) di 2 variabili, otteniamo un AC binario $2^N \xrightarrow{f_\alpha} 2^N$.

Sia ad esempio $\alpha = \{10, 01\}$ (con la solita identificazione). Allora f_α è data dalla tabella

x			$f(x)$		
0	0	0	0	0	0
0	0	1	1	0	1
0	1	0	0	1	1
1	0	0	1	1	0
0	1	1	1	1	0
1	0	1	0	1	1
1	1	0	1	0	1
1	1	1	0	0	0

Numerare gli stati e disegnare il grafico del sistema.

In questo numero

Automi di Wolfram

Gli automi cellulari più famosi sono gli *automi di Wolfram* (AW).

Gli automi di Wolfram di *primo ordine* (AW_1) sono definiti nel modo seguente. Sia $n \geq 3$. Definiamo un PAC binario

$$2^n \times n \xrightarrow{\varepsilon} 2^3$$

con

$$\begin{aligned} \varepsilon(x, 0) &:= (x_{n-1}, x_0, x_1) \\ \varepsilon(x, 1) &:= (x_0, x_1, x_2) \\ \varepsilon(x, 2) &:= (x_1, x_2, x_3) \\ &\cdots \\ \varepsilon(x, n-2) &:= (x_{n-3}, x_{n-2}, x_{n-1}) \\ \varepsilon(x, n-1) &:= (x_{n-2}, x_{n-1}, x_0) \end{aligned}$$

L'effetto dello stato x sul nodo i dipende quindi solo da x_{i-1}, x_i, x_{i+1}, dove ai bordi gli indici vanno calcolati ciclicamente.

A questo punto per ogni valutazione $2^3 \xrightarrow{\alpha} 2$ (cioè per ogni FB di 3 variabili) otteniamo un AC $2^n \longrightarrow 2^n$.

Gli automi di Wolfram di secondo ordine (AW_2) sono definiti in modo analogo, solo che stavolta si sceglie $n \geq 5$ ed $E := 2^5$ con il PAC

$$2^n \times n \xrightarrow{\varepsilon} 2^5$$

definito da

$$\varepsilon(x, i) = (x_{i-2}, x_{i-1}, x_i, x_{i+1}, x_{i+2})$$

con gli indici ai bordi calcolati in modo ciclico. Adesso per ogni valutazione $2^5 \longrightarrow 2$ otteniamo un AC $2^n \longrightarrow 2^n$.

Come sappiamo dalla nota 11.11, possiamo definire 256 valutazioni $2^3 \longrightarrow 2$ e 2^{32} (cioè più di 4 miliardi) valutazioni $2^5 \longrightarrow 2$; siccome le applicazioni ε nella definizione 15.8 sono iniettive (come si vede facilmente), ciò significa che, fissato n, esistono 256 AW_1 e 2^{32} AW_2 su 2^n.

Stephen Wolfram (nato nel 1959 e inventore del software matematico commerciale *Mathematica*) ha studiato intensamente soprattutto gli AW_1 e raccolto i sui esperimenti nel suo nuovo libro *A new kind of science* (1250 pagine e più un libro d'arte che di matematica e caratterizzato da una certa esaltazione, ma piuttosto interessante).

Rappresentazione binaria

Ogni numero naturale n possiede una rappresentazione binaria, cioè una rappresentazione della forma

$$n = a_k 2^k + a_{k-1} 2^{k-1} + \cdots + a_2 2^2 + a_1 2 + a_0$$

con coefficienti (o cifre binarie) $a_i \in \{0, 1\}$. Per $n = 0$ usiamo $k = 0$ ed $a_0 = 0$; per $n > 0$ chiediamo che $a_k \neq 0$. Con queste condizioni k e gli a_i sono univocamente determinati. Sia $r_2(n) = (a_k, \ldots, a_0)$ il vettore i cui elementi sono queste cifre. Dalla rappresentazione binaria si deduce la seguente relazione ricorsiva:

$$r_2(n) = \begin{cases} (n) & \text{se } n \leq 1 \\ (r_2(\frac{n}{2}), 0) & \text{se } n \text{ è pari} \\ (r_2(\frac{n-1}{2}), 1) & \text{se } n \text{ è dispari} \end{cases}$$

A pagina 15 del corso di Programmazione abbiamo scritto una funzione in C per questo algoritmo. In R è ancora più facile. La funzione `M.rapp2` che definiamo accetta un secondo parametro facoltativo `cifre`; quando questo è maggiore del numero di cifre necessarie per la rappresentazione binaria di n, i posti iniziali vuoti vengono riempiti con zeri usando la funzione `rep` di R.

```
M.rapp2 = function (n, cifre=NA)
{if (n<=1) v=c(n)
else if (n%%2==0) v=c(M.rapp2(n/2),0)
else v=c(M.rapp2((n-1)/2),1)
if (is.na(cifre)) v else
{n=length(v);
if (n>=cifre) v
else c(rep(0,cifre-n),v)}}
```

Per provare la funzione scriviamo in *programma* queste istruzioni:

```
for (n in c(0:10,seq(19,240,29)))
cat(formatC(n,width=3),' ',
  M.rapp2(n,8),'\n')
```

Con `Alfa()` nel terminale di R otteniamo l'output

```
  0   0 0 0 0 0 0 0 0
  1   0 0 0 0 0 0 0 1
  2   0 0 0 0 0 0 1 0
  3   0 0 0 0 0 0 1 1
  4   0 0 0 0 0 1 0 0
  5   0 0 0 0 0 1 0 1
  6   0 0 0 0 0 1 1 0
  7   0 0 0 0 0 1 1 1
  8   0 0 0 0 1 0 0 0
  9   0 0 0 0 1 0 0 1
 10   0 0 0 0 1 0 1 0
 19   0 0 0 1 0 0 1 1
 48   0 0 1 1 0 0 0 0
 77   0 1 0 0 1 1 0 1
106   0 1 1 0 1 0 1 0
135   1 0 0 0 0 1 1 1
164   1 0 1 0 0 1 0 0
193   1 1 0 0 0 0 0 1
222   1 1 0 1 1 1 1 0
```

Lo schema di Horner ricorsivo (pag. 16 del corso di Programmazione) si traduce altrettanto facilmente in R. Si tratta di calcolare il valore $f(\alpha)$ di un polinomio

$$f = a_0 x^n + a_1 x^{n-1} + \ldots + a_n$$

Infatti, se $v = (a_0, \ldots, a_n)$ è il vettore dei coefficienti del polinomio, allora per $n = 0$ il valore è semplicemente a_0 (nello spirito vettoriale di R possiamo in questo caso identificare a_0 con v), altrimenti

$$f(\alpha) = \alpha(a_0 \alpha^{n-1} + a_1 \alpha^{n-2} + \ldots + a_{n-1}) + a_n$$

cosicché possiamo ricondurci al vettore v' che si ottiene da v togliendo l'ultimo elemento del vettore. Per il calcolo del valore di una rappresentazione binaria si usa $\alpha = 2$ e questa è la preimpostazione che scegliamo:

```
M.horner = function (v, alfa=2)
{n=length(v)-1; if (n==0) v
else alfa*M.horner(v[1:n],alfa)+v[n+1]}
```

Numerazione delle FB

Il comportamento degli AW dipende da n, ma soprattutto dalla valutazione scelta. Per gli AW_1 queste sono le FB $2^3 \longrightarrow 2$ che da Wolfram sono state numerate secondo lo schema seguente, in cui gli argomenti sono elencati nel'ordine usato da Wolfram, cioè da $7 = (111)_2$ a $0 = (000)_2$ dall'alto verso il basso:

111	0	0	0	0		0		1	1
110	0	0	0	0		1		1	1
101	0	0	0	0		0		1	1
100	0	0	0	0	...	0	...	1	1
011	0	0	0	0		1		1	1
010	0	0	0	0		1		1	1
001	0	0	1	1		0		1	1
000	0	1	0	1		1		0	1
k	0	1	2	3	...	77	...	254	255

Consideriamo, in altre parole, ogni colonna di valori come la rappresentazione binaria di un numero naturale k, con il coefficiente della potenza di 2 più alta (128) in alto e quella più bassa in basso, cosicché, come nella figura, $77 = (01001101)_2$ corrisponde alla FB

7	111	0	
6	110	1	2
5	101	0	
4	100	0	
3	011	1	5
2	010	1	6
1	001	0	
0	000	1	8

Vediamo così che le 256 FB $2^3 \longrightarrow 2$ possono essere numerate in modo naturale. Sia $0 \leq k \leq 255$. Come si ottiene la FB con il numero k? Supponiamo che $k = 77$. Se $v = (v_1, \ldots, v_8) = (0, 1, 0, 0, 1, 1, 0, 1)$, vediamo che gli indici i, per i quali $v_i = 1$, sono dati dal vettore $(2, 5, 6, 8)$, a cui corrispondono triple che, considerate come rappresentazioni binarie, hanno i valori $6, 3, 2, 0$ e che si ottengono dagli indici i come $8 - i$:

$$8 - 2 = 6, 8 - 5 = 3, 8 - 6 = 2, 8 - 8 = 0$$

Dobbiamo in altre parole saper determinare la FB la cui FND è data dalle triple che corrispondono ai numeri $6, 3, 2, 0$. E ciò è semplicissimo!

Sia v un vettore di numeri. Vogliamo calcolare la funzione α che applicata a un vettore booleano e calcola il numero s per il quale $s = (e)_2$ e restituisce 1 se s appartiene a v. I programmi per le FB faranno parte della sezione MFB della libreria, perciò definiamo

```
Mfb.alfa = function (v)
function (e)
is.element(M.horner(e),v)
```

Questa funzione può essere usata per FB $2^n \longrightarrow 2$ per ogni n.

Volevamo però partire da $k = 77$ e trovare le FB $2^3 \longrightarrow 2$ corrispondenti; dobbiamo quindi calcolare il vettore v che corrisponde alla FND di α. Usiamo il procedimento descritto sopra, calcolando prima il vettore booleano u di lunghezza 8 che corrisponde alla rappresentazione binaria di k, e raccogliendo da questo in v tutti gli $8 - i$ che corrispondono a un indice i per cui $u_i = 1$:

```
Dinac.alfa.aw1 = function (k)
{u=M.rapp2(k,8); v=c()
for (i in 1:8) if (u[i]) v=c(v,8-i)
Mfb.alfa(v)}
```

---

n incide molto meno sul comportamento di un AW, soprattutto se n è sufficientemente grande e gli effetti ciclicamente riflessi ai bordi si fanno meno sentire.

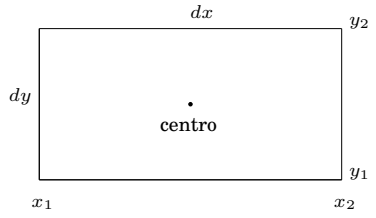
Esercizio 16.1. Consideriamo, per $K = 2$ e identificando $\{a, b, c\}$ con 3, il PAC $2^3 \times 3 \xrightarrow{\varepsilon} 2^2$ dell'esempio 15.8. Dimostrare che per ogni valutazione $2^2 \xrightarrow{\alpha} 2$ esiste una valutazione $2^3 \xrightarrow{\alpha'} 2$ tale che i PACV $2^3 \times 3 \xrightarrow{\varepsilon} 2^2 \xrightarrow{\alpha} 2$ e $2^3 \times 3 \xrightarrow{\varepsilon'} 2^3 \xrightarrow{\alpha'} 2$ sono equivalenti (nel senso della definizione 15.4), se $2^3 \times 3 \xrightarrow{\varepsilon'} 2^3$ è il PAC degli AW_1 (pagina 15). Molto facile, ma istruttivo.

Rettangoli

Nella rappresentazione grafica di automi cellulari, soprattutto di AW e dello loro generalizzazioni 2-dimensionali, si utilizzano spesso tabelle rettangolari, per le quali vogliamo preparare le necessarie funzioni in R.

Rettangoli sono definiti nella sezione GFR della nostra libreria. La funzione di base `Gfr` restituisce, nella forma di un vettore di quattro numeri complessi, gli angoli in basso a sinistra (`bs`), in basso a destra (`bd`), in alto a destra (`ad`) e in alto a sinistra (`as`) di un rettangolo. L'ordine `(bs,bd,ad,as)` è scelto in modo da poter applicare direttamente la funzione `polygon` per disegnare il rettangolo.

`Gfr` prevede gli argomenti `dx`, `dy`, `x1`, `x2`, `y1`, `y2` e `centro`, tutti facoltativi e preimpostati a `NA`; il loro significato è illustrato nella figura:



Il rettangolo può essere quindi definito da una delle seguenti combinazioni di parametri, indicati in notazione matematica, con il centro uguale a (x, y), da essi vengono calcolati `x1,x2,y1,y2`, da cui si ottengono `bs,bd,ad,as`.

argomenti	parametri calcolati
x_1, x_2, y_1, y_2	$-$
$(x, y), dx, dy$	$x_1 = x - \frac{dx}{2}, x_2 = x + \frac{dx}{2}$, $y_1 = y - \frac{dy}{2}, y_2 = y + \frac{dy}{2}$
x_1, y_1, dx, dy	$x_2 = x_1 + dx, y_2 = y_1 + dy$
x_1, y_2, dx, dy	$x_2 = x_1 + dx, y_1 = y_2 - dy$
x_2, y_1, dx, dy	$x_1 = x_2 - dx, y_2 = y_1 + dy$
x_2, y_2, dx, dy	$x_1 = x_2 - dx, y_1 = y_2 - dy$

Per verificare che nessun elemento di una successione di valori sia NA, usiamo la seguente funzione:

```
P.def = function (...)
{a=list(...); !any(is.na(a))}
```

Possiamo così creare la funzione `Gfr`:

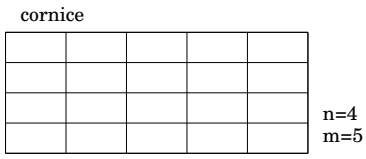
```
Gfr = function (dx=NA,dy=NA,
  x1=NA,x2=NA,y1=NA,y2=NA,centro=NA)
{if (P.def(centro))
{x=Re(centro); y=Im(centro)
x1=x-dx/2; x2=x+dx/2
y1=y-dy/2; y2=y+dy/2}
else if (P.def(x1,y1,dx,dy))
{x2=x1+dx; y2=y1+dy}
else if (P.def(x1,y2,dx,dy))
{x2=x1+dx; y1=y2-dy}
else if (P.def(x2,y1,dx,dy))
{x1=x2-dx; y2=y1+dy}
else if (P.def(x2,y2,dx,dy))
{x1=x2-dx; y1=y2-dy}
bs=x1+1i*y1; bd=x2+1i*y1
as=x1+1i*y2; ad=x2+1i*y2
c(bs,bd,ad,as)}
```

Definiamo inoltre funzioni che permettono di ricalcolare gli argomenti di `Gfr` dal vettore `c(bs,bd,ad,as)` che la funzione restituisce:

```
Gfr.centro = function (r) (r[1]+r[3])/2

Gfr.dx = function (r) Re(r[2]-r[1])

Gfr.dy = function (r) Im(r[3]-r[2])

Gfr.x1 = function (r) Re(r[1])

Gfr.x2 = function (r) Re(r[2])

Gfr.y1 = function (r) Im(r[1])

Gfr.y2 = function (r) Im(r[3])
```

Tabelle rettangolari

Tabelle rettangolari sono definite nella sezione GFT della libreria. La funzione di base `Gft` restituisce semplicemente la lista `list(n,m,r)`, in cui `r` è il rettangolo che costituisce la cornice della tabella, `n` è il numero di righe ed `m` il numero delle colonne.

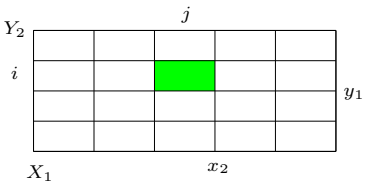


La funzione `Gft` ha `n` ed `m` come parametri obbligatori a cui seguono, nello stesso formato come per `Gfr`, i parametri che definiscono la cornice, oppure la cornice stessa come parametro:

```
Gft = function (n,m,r=NA, ...)
{if (!P.def(r)) r=Gfr(...)
list(n=n,m=m,r=r)}
```

Come si vede, il risultato `tab` della funzione è una *lista con nomi* i cui componenti sono `tab$n`, `tab$m` e `tab$r`.

Sia adesso data una tabella con n righe ed m colonne. Per $1 \le i \le n$ e $1 \le j \le m$ vogliamo determinare la casella che si trova nell'i-esima riga e j-esima colonna. Questa casella è un rettangolo per conto proprio e viene rappresentata mediante un vettore `c(bs,bd,ad,as)`; la sua larghezza dx è la larghezza della tabella divisa per m e la sua altezza è l'altezza della tabella divisa per n. x_1 e y_1 si calcolano, come si vede nella figura,



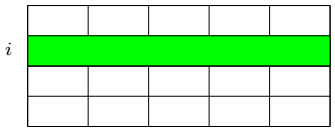
mediante le formule

$$x_2 = X_1 + j \cdot dx$$
$$y_1 = Y_2 - i \cdot dy$$

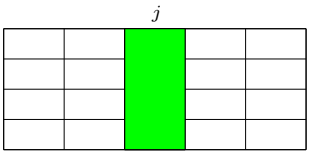
dove con X_1, Y_2 abbiamo indicato i parametri relativi alla cornice. Possiamo così definire la nostra funzione:

```
Gft.casella = function (tab,i,j)
{r=tab$r;
dx=Gfr.dx(r)/tab$m; dy=Gfr.dy(r)/tab$n
x2=Gfr.x1(r)+j*dx;
y1=Gfr.y2(r)-i*dy
Gfr(x2=x2,y1=y1,dx=dx,dy=dy)}
```

In modo simile calcoliamo la i-esima riga e la j-esima colonna di una tabella.



$$dx = DX, dy = \frac{DY}{n}$$
$$x_1 = X_1, y_1 = Y_2 - i \cdot dy$$



$$dx = \frac{DX}{m}, dy = DY,$$
$$x_2 = X_1 + j \cdot dx, y_1 = Y_1$$

```
Gft.riga = function (tab,i)
{r=tab$r; dx=Gfr.dx(r);
dy=Gfr.dy(r)/tab$n;
x1=Gfr.x1(r); y1=Gfr.y2(r)-i*dy
Gfr(x1=x1,y1=y1,dx=dx,dy=dy)}

Gft.colonna = function (tab,j)
{r=tab$r;
dx=Gfr.dx(r)/tab$m; dy=Gfr.dy(r)
x2=Gfr.x1(r)+j*dx; y1=Gfr.y1(r)
Gfr(x2=x2,y1=y1,dx=dx,dy=dy)}
```

Per disegnare tutte le caselle (cioè i loro bordi) di una tabella `tab` con `n` righe ed `m` colonne possiamo usare le istruzioni

```
for (i in 1:n) polygon(Gft.riga(tab,i))
for (j in 1:m) polygon(Gft.colonna(tab,j))
polygon(tab$r)
```

oppure definire una funzione apposita che contiene queste istruzioni:

```
Gd.tabella = function (tab)
{for (i in 1:tab$n)
  polygon(Gft.riga(tab,i))
for (j in 1:tab$m)
  polygon(Gft.colonna(tab,j))
polygon(tab$r)}
```

Questa funzione viene spesso utilizzata alla fine o all'inizio di un disegno.

Esercizio 17.1. Scrivere una funzione

```
Casella = function (tab,i,j,k=NA)
```

che, generalizzando `Gft.casella`, accetta anche un parametro facoltativo `k` che indica il numero della casella, se si comincia a contare le caselle della tabella da `1` a nm.

Come si ottengono n ed m?

Grafica per automi cellulari

Sia dato un AC binario $2^N \xrightarrow{f} 2^N$. Possiamo assumere che $N = \{1, \ldots, n\}$. Scegliamo uno stato iniziale $x \in 2^N$ e supponiamo che vogliamo esaminare la successione

$$x, x+1, \ldots, x+t-1$$

cioè i primi t elementi di $S(x)$. Creiamo allora una tabella con t righe ed n colonne (attenzione!). Rappresentiamo i valori $x_1, \ldots, x_n$ nella prima riga, colorando (ad esempio in rosso o nero su sfondo bianco o giallo) la casella nella j-esima colonna, se $x_j = 1$. Poi calcoliamo $x+1$ e rappresentiamo i valori di $x+1$ nella seconda riga nello stesso modo, e così via. Le funzioni per gli automi cellulari vanno inserite nella sezione DINAC della libreria.

```
Dinac = function (f,x,t,col="red")
{n=length(x)
tab=Gft(t,n,x1=0,y1=0,dx=n,dy=t)
polygon(tab$r)
for (i in 1:t) {for (j in 1:n) if (x[j])
polygon(Gft.casella(tab,i,j),col=col)
x=f(x)}}
```

Abbiamo visto nella nota 15.6 che ogni AC $K^N \xrightarrow{f} K^N$ può essere ottenuto da un PACV $K^N \times N \xrightarrow{\varepsilon} E \xrightarrow{\alpha} K$ ponendo

$$f(x)_a := \alpha(\varepsilon(x, a))$$

Limitandoci ad automi binari possiamo realizzare questa costruzione con la seguente funzione:

```
Dinac.fun = function (eps,alfa)
function (x)
{n=length(x); y=numeric(n)
for (a in 1:n) y[a]=alfa(eps(x,a))
y}
```

A pagina 15 abbiamo descritto la funzione $\varepsilon : 2^n \times n \longrightarrow 2^3$ che viene utilizzata negli AW_1. Tenendo conto che gli indici in R vanno da 1 ad n, creiamo la funzione

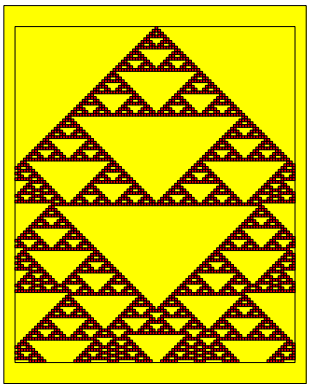
```
Dinac.eps.aw1 = function (x,a)
{n=length(x)
P.quale(a,1,c(x[n],x[1],x[2]),
  n,c(x[n-1],x[n],x[1]),
  c(x[a-1],x[a],x[a+1]))}
```

Esempi di AW_1

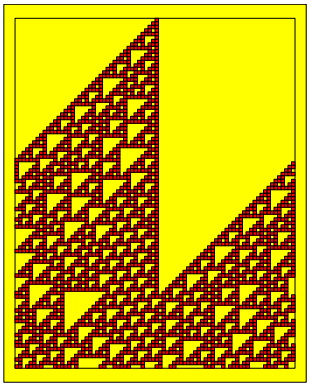
La regola 77 non corrisponde a una figura particolarmente interessante; proviamo con la regola 126:

```
alfa=Dinac.alfa.aw1(126)
eps=Dinac.eps.aw1

f=Dinac.fun(eps=eps,alfa=alfa)

n=100
x=c(rep(0,n/2),1,rep(0,n/2-1))

t=120
Gs.postscript("18-aw1-126.ps",4,5)
latox=c(0,n); latoy=c(0,t)
par(bg="yellow")
Gd.grafica(latox,latoy)
Dinac(f,x,t)
dev.off()
```

Otteniamo questa figura:

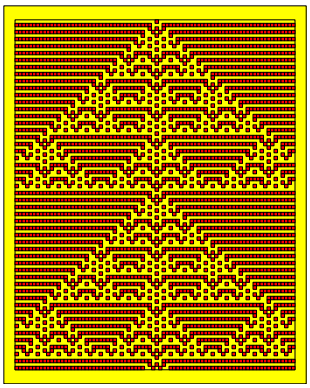


Piuttosto interessante è la regola 110 che otteniamo, sostituendo nella prima riga del programma precedente 126 con 110:



Questa volta abbiamo scelto `n=80, t=100`, perché la risoluzione nell'esempio precedente era un po' troppo fine. Lo spazio delle fasi è quindi adesso $X = 2^{80}$. Anche per uno spazio così grande valgono però i risultati generali che abbiamo ottenuto all'inizio del corso e quindi l'orbita di ogni punto dopo un numero finito di passi deve diventare periodica.

Un elenco grafico di tutti gli AW_1 si trova nel libro di Wolfram sulle pagine 55-56. Proviamo ancora la regola 105:



Piccoli cambiamenti nelle regole di generazione conducono spesso a figure molto diverse. A questo proposito si potrebbe leggere l'articolo *Come nasce una forma* a pagina 38 del corso di Algoritmi 2004/05.

Alcuni AW_2

Molto maggiore è la variabilità degli AW_2. Sappiamo che ne esistono 2^{32}, quindi più di 4 miliardi. Il numero delle immagini che possiamo ottenere è però molto maggiore, perché le immagini dipendono anche dal punto iniziale che scegliamo.

Mentre è quasi impossibile studiare in modo esauriente gli AW_2, non è difficile scrivere le funzioni necessarie:

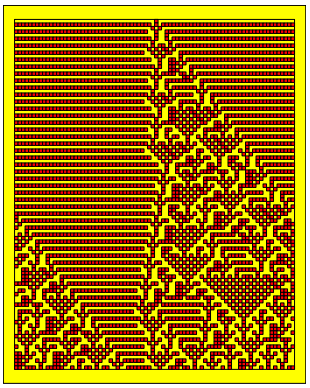
```
Dinac.alfa.aw2 = function (k)
{u=M.rapp2(k,32); v=c()
for (i in 1:32) if (u[i]) v=c(v,32-i)
Mfb.alfa(v)}

Dinac.eps.aw2 = function (x,a)
{n=length(x)
P.quale(a,1,c(x[n-1],x[n],x[1],x[2],x[3]),
  2,c(x[n],x[1],x[2],x[3],x[4]),
  n-1,c(x[n-3],x[n-2],x[n-1],x[n],x[1]),
  n,c(x[n-2],x[n-1],x[n],x[1],x[2]),
  c(x[a-2],x[a-1],x[a],x[a+1],x[a+2]))}
```

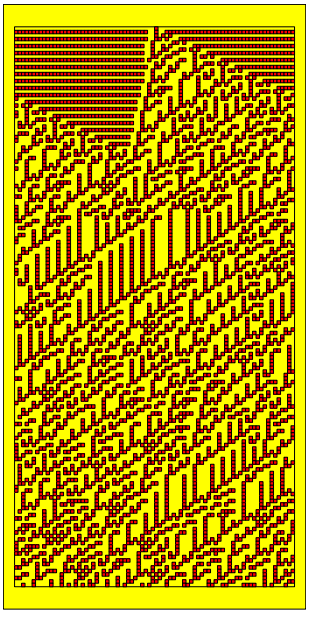
È sufficiente adesso sostituire `aw1` con `aw2` nel programma; nell'istruzione

```
alfa=Dinac.alfa.aw2(k)
```

si può usare ogni k con $0 \leq k < 429467696$; cfr. nota 11.11.

Proviamo con $k = 84110555$ e $k = 689209$:





LABORATORIO DI PROGRAMMAZIONE

Corso di laurea in matematica | Anno accademico 2004/05 | Numero 6

L'ipercubo

Situazione 19.1. Sia $X = \{1, \dots, n\}$ con $n \geq 0$. A partire dalla nota 21.1 X è un insieme finito qualsiasi con n elementi con $P, Q \subset X$.

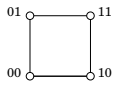
Identificando $\mathcal{P}(X)$ con 2^n, ogni funzione booleana di n variabili corrisponde a un sottoinsieme $\alpha \subset 2^n$.

2^n geometricamente è un *ipercubo* che può essere visualizzato nel modo seguente.

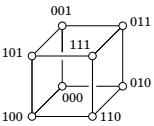
$n = 0$:

$n = 1$:

$n = 2$:



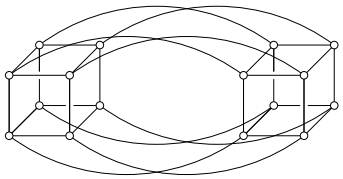
$n = 3$:



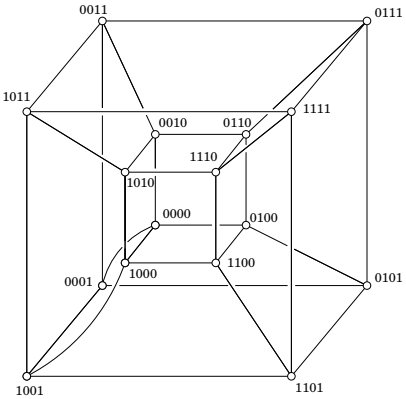
$n = 4$: L'ipercubo a 4 dimensioni si ottiene dal cubo 3-dimensionale attraverso la relazione

$$2^4 = 2^3 \times \{0, 1\}$$

Dobbiamo quindi creare due copie del cubo 3-dimensionale. Nella rappresentazione grafica inoltre sono adiacenti e quindi connessi con una linea quei vertici che si distinguono in una sola coordinata. Oltre ai legami all'interno dei due cubi dobbiamo perciò unire i punti $(x, y, z, 0)$ e $(x, y, z, 1)$ per ogni x, y, z.



La figura diventa molto più semplice, se si pone uno dei due cubi (quello con la quarta coordinata $= 0$) all'interno dell'altro (quello con la quarta coordinata $= 1$):



Una rappresentazione che rispecchia meglio la struttura reticolare di 2^n la otterremo fra poco mediante una proiezione $\mathbb{R}^n \longrightarrow \mathbb{R}^2$.

$n \geq 5$: Teoricamente anche qui si può usare la relazione

$$2^n = 2^{n-1} \times \{0, 1\}$$

ma la visualizzazione diventa difficoltosa.

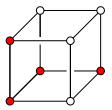
Ogni vertice dell'ipercubo corrisponde a un elemento di 2^n che nell'interpretazione insiemistica rappresenta a sua volta un sottoinsieme di X (il punto 0101 ad esempio l'insieme $\{2, 4\}$ se $X = \{1, 2, 3, 4\}$.

Per rappresentare una FB $\alpha \subset \mathcal{P}(X)$ coloreremo i suoi elementi sull'ipercubo in nero o in rosso, lasciando vuoti i cerchietti nei vertici che non appartengono ad α.

Rappresenteremo così ad esempio

$$\alpha = \{000, 010, 100, 101\} \subset 2^3$$

mediante la figura



Implicanti

Nota 19.2. Ogni intervallo σ di $\mathcal{P}(X)$ possiede un'unica rappresentazione della forma

$$\sigma = i_1^+ \dots i_p^+ j_1^- \dots j_r^-$$

con gli indici tutti distinti. Come si riconosce un tale intervallo nella rappresentazione grafica sull'ipercubo? Ma

$$i^+ = \{\text{punti dell'ipercubo con } i\text{-esima coordinata} = 1\}$$
$$i^- = \{\text{punti dell'ipercubo con } i\text{-esima coordinata} = 0\}$$

Ogni i^+ e ogni i^- corrisponde quindi a una faccia $(n-1)$-dimensionale di 2^n; chiedendo più di queste condizioni otteniamo tutte le facce di 2^n. Abbiamo perciò una biiezione naturale

$$\{\text{intervalli di } \mathcal{P}(X)\} \longleftrightarrow \{\text{facce dell'ipercubo } 2^n\}$$

Definizione 19.3. Sia $\alpha \subset \mathcal{P}(X)$ una funzione booleana. Un *implicante* di α è un intervallo di $\mathcal{P}(X)$ contenuto in α.

In questo numero

Implicanti massimali

Definizione 19.4. Sia $\alpha \subset \mathcal{P}(X)$ una funzione booleana. Un implicante di α non contenuto in nessun altro implicante di α è detto un *implicante massimale* (o *primo*) di α.

Nota 19.5. Sia $\alpha \subset \mathcal{P}(X)$ una funzione booleana. Allora $A = \bigcup_{A \in \alpha} \{A\}$. Siccome ogni $\{A\} = [A, A]$ è un intervallo, vediamo che α è l'unione (vuota se $\alpha = \emptyset$) dei sui implicanti. Inoltre, essendo X finito, ogni implicante di α è contenuto in (almeno) un implicante massimale di α, quindi α è anche uguale all'unione dei suoi implicanti massimali. Denotando con $\mathrm{Max}(\alpha)$ l'insieme degli implicanti massimali di α, abbiamo perciò

$$\alpha = \bigcup_{\sigma \in \mathrm{Max}(\alpha)} \sigma$$

Se α è rappresentata sull'ipercubo, gli implicanti di α sono esattamente le facce dell'ipercubo contenute in α.

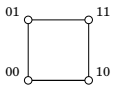
$\mathrm{Max}(\alpha)$ è l'insieme degli elementi massimali dell'insieme delle facce dell'ipercubo contenute in α.

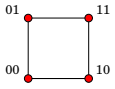
Nota 19.6. Una delle tecniche di semplificazione di una FB α consiste nella rappresentazione di α come unione di implicanti massimali utilizzando complessivamente il minor numero possibile di condizioni (cfr. l'inizio della nota 19.2). Ciò è teoricamente sempre possibile ma molto difficile, e mentre esiste un algoritmo (di Quine/McCluskey) che permette di individuare tutti gli implicanti primi di una funzione booleana data dalla sua FND, è probabilmente intrattabile il problema di scegliere in modo efficiente quegli implicanti primi che sono necessari per la rappresentazione minimale che peraltro non è univocamente determinata.

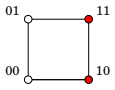
Oltre a ciò, la rappresentazione di una funzione booleane tramite i suoi implicanti può essere eseguita in pratica solo per un numero non troppo grande di argomenti (altrimenti non siamo nemmeno in grado di compilare la tabella per la FND, da cui parte ad esempio l'algoritmo di Quine/McCluskey). Bisogna perciò spesso utilizzare altre tecniche (metodi grafici che utilizzano i *diagrammi binari di decisione* oppure i metodi algebrici potenti ma difficili che si basano sulla teoria dei *campi finiti*).

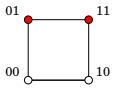
Le funzioni booleane binarie

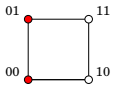
Rappresentiamo graficamente le 16 funzioni binarie della nota 14.14:

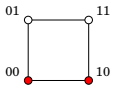
$0 = \{\}$ 

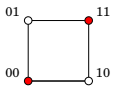
$1 = \{00, 01, 10, 11\}$ 

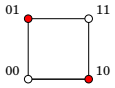
$p_x = \{10, 11\}$ 

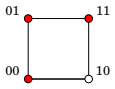
$p_y = \{01, 11\}$ 

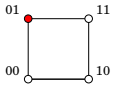
$n_x = \{00, 01\}$ 

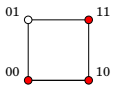
$n_y = \{00, 10\}$ 

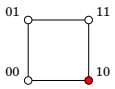
$== \{00, 11\}$ 

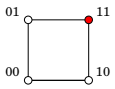
XOR $= \{01, 10\}$ 

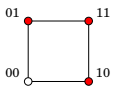
$\leq = \{00, 01, 11\}$ 

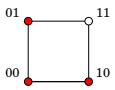
$< = \{01\}$ 

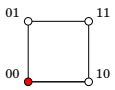
$\geq = \{00, 10, 11\}$ 

$> = \{10\}$ 

AND $= \{11\}$ 

OR $= \{01, 10, 11\}$ 

NAND $= \{00, 01, 10\}$ 

NOR $= \{00\}$ 

Le funzioni booleane unarie

Per completezza aggiungiamo le 4 funzioni unarie della nota 14.13:

$0 = \{\}$ 

$1 = \{0, 1\}$ 

id $= \{1\}$ 

NOT $= \{0\}$ 

lapply e sapply

Siano $x = (x_1, \ldots, x_n)$ un vettore o una lista ed f una funzione che (in R) non operi già in modo vettoriale. Allora con `lapply(x,f)` otteniamo la sequenza $(f(x_1), \ldots, f(x_n))$ in forma di una lista, con `sapply(x,f)` la stessa sequenza in forma di un vettore quando ciò è possibile.

Se la `f` ha argomenti addizionali, questi possono essere aggiunti ai parametri di `lapply` ed `sapply`, quindi il risultato di `sapply(x,f,a,b)` diventa il vettore $(f(x_1, a, b), \ldots, f(x_n, a, b))$. Talvolta è necessario applicare l'operatore `c` al risultato. Esempio:

```
f = function (x) x*x
u=sapply(2:5,f)
print(u)
# output: 4 9 16 25

f=function (x,rho=1) x*x+rho
u=sapply(2:5,f,rho=2)
print(u)
# output: 6 11 18 27
```

La funzione `Linden` a pagina 29 del corso di Programmazione diventa semplicemente

```
L = function (a,f) c(sapply(a,f))
```

Conosciamo già, come esempio semplice ma interessante di un sistema di Lindenmayer la successione di Morse, definita nel modo seguente:

$A = \{0, 1\}$
generatore: 0
leggi: $0 \longmapsto 01$, $1 \longmapsto 10$

con la seguente traduzione in R:

```
Ls.morse = function (x)
P.quale(x,0,c(0,1),1,c(1,0))
```

Per provare usiamo

```
u=0
for (k in 1:6)
{print(u); u=L(u,Ls.morse)}
```

con output

```
0
01
0110
01101001
0110100110010110
01101001100101101001011001101001
```

Le funzioni `lapply` e `sapply` sono caratteristiche di molti linguaggi ad alto livello e già presenti nel più antico di tutti, il *Lisp*. Nel *Perl* esse corrispondono alla funzione `map`.

Proiezioni lineari $\mathbb{R}^n \longrightarrow \mathbb{R}^2$

Un'applicazione lineare $\varphi : \mathbb{R}^n \longrightarrow \mathbb{R}^2$ è determinata dalle immagini

$$w_1 := \varphi(\delta_1), \ldots, w_n := \varphi(\delta_n)$$

dei vettori $\delta_1, \ldots, \delta_n$ della base standard. Per $x = (x_1, \ldots, x_n) \in \mathbb{R}^n$ si ha allora

$$x = x_1\delta_1 + \ldots + x_n\delta_n$$

e quindi

$$\varphi(x) = x_1 w_1 + \ldots + x_n w_n$$

La traduzione in R è immediata e incredibilmente semplice:

```
G.lin = function (w)
function (p) sum(p*w)
```

Abbiamo già visto a pagina 16 che gli elementi dell'ipercubo possono essere ottenuti mediante la rappresentazione binaria. La funzione `Mfb.cubo` restituisce l'ipercubo in forma di una lista:

```
Mfb.cubo = function (n)
lapply(0:(2^n-1),M.rapp2,cifre=n)
```

Non dimenticare le parentesi attorno a `2^n-1`.

Per disegnare l'ipercubo a 4 dimensioni, dopo

```
vertici=Mfb.cubo(4)

w1=-2+1i
w2=-1+1i
w3=1+1i
w4=3+1i
```

e

```
f=G.lin(c(w1,w2,w3,w4))
```

possiamo disegnare le proiezioni dei punti dell'ipercubo con

```
t=seq(0,2*pi,by=0.01)
cerchio=Gf.cerchio(t,0.1)
for (p in vertici)
polygon(cerchio+f(p),col="yellow")
```

La scelta dei w_j sarà discussa a pagina 21.

Quando proiettiamo gli elementi di 2^n sul piano, vogliamo però anche visualizzare gli *spigoli* che uniscono i punti adiacenti nell'ipercubo, cioè punti che si distinguono in una sola coordinata. Per determinare queste coppie di punti utilizziamo la *distanza di Hamming* che è definita come il numero delle coordinate in cui due elementi di 2^n differiscono e che in R può essere calcolata con la seguente semplicissima funzione:

```
Mfb.hamm = function (u,v)
sum(abs(u-v))
```

A questo punto, per disegnare gli spigoli, usiamo le istruzioni

```
r=length(vertici)
for (i in 1:(r-1)) {u=vertici[[i]]
for (j in (i+1):r) {v=vertici[[j]]
if (Mfb.hamm(u,v)==1)
Gdf(f(u),f(v),punta=0,d1=0.1,d2=0.1)}}
```

Come scegliere i w_j

Dalla scelta delle immagini w_j dei vettori δ_j nella proiezione dipende quali aspetti di una figura sono posti in risalto. Se, come nel nostro esempio, vogliamo che la figura rispecchi la struttura reticolare di 2^n, poniamo sulla stessa altezza punti che corrispondono a insiemi con lo stesso numero di elementi. In questo caso i w_j (che corrispondono ai sottoinsiemi con un solo punto) hanno tutti la stessa parte immaginaria. Abbiamo quindi ad esempio $w_j = a_j + i$ per ogni j. Adesso dobbiamo però evitare che punti distinti di 2^n vengano proiettati nello stesso punto del piano. Nel caso $n = 4$ ciò significa che i 6 numeri

$$\begin{matrix} a_1 + a_2, a_1 + a_3, a_1 + a_4, \\ a_2 + a_3, a_2 + a_4, a_3 + a_4 \end{matrix} \quad (*)$$

siano tutti distinti, così come devono essere distinti i 4 numeri

$$\begin{matrix} a_1 + a_2 + a_3, a_1 + a_2 + a_4, \\ a_1 + a_3 + a_4, a_2 + a_3 + a_4 \end{matrix} \quad (**)$$

Noi abbiamo scelto

$$a_1 = -2, a_2 = -1, a_3 = 1, a_4 = 3$$

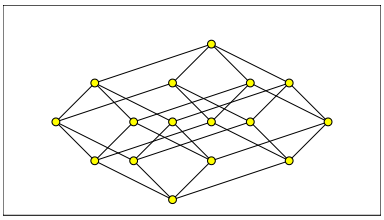
cosicché i numeri (*) sono

$$-3, -1, 1, 0, 2, 4$$

e i numeri (**) siano

$$-2, 0, 2, 3$$

Otteniamo così la figura



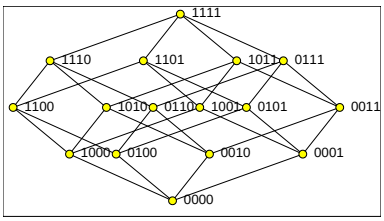
Per indicare anche gli insiemi a cui i punti corrispondono, possiamo aggiungere le istruzioni

```
for (p in vertici) {z=f(p)
text(Re(z)-0.2,Im(z),
  paste(p,collapse=''),
  cex=0.4,pos=4)}
```

Se, per ingrandire e spostare leggermente la figura, definiamo

```
f=G.lin(1.2*c(w1,w2,w3,w4)-0.25)
```

otteniamo



Contiamo gli intervalli

Nota 21.1. Consideriamo un intervallo $[P, Q]$ con $P \subset Q$ (altrimenti $[P, Q] = \emptyset$). Allora $[P, Q] \subset \mathcal{P}(Q)$, infatti

$$[P, Q] = \{A \in \mathcal{P}(Q) \mid P \subset A\}$$

Siccome ogni $A \in [P, Q]$ contiene l'insieme fissato P, ogni tale A è univocamente determinato dalla differenza $A \setminus P$, infatti

$$A = P \dot\cup (A \setminus P)$$

Abbiamo perciò una biiezione

$$[P, Q] \longleftrightarrow \mathcal{P}(Q \setminus P)$$

in cui $A \in [P, Q]$ corrisponde ad $A \setminus P$ ed $Y \in \mathcal{P}(Q \setminus P)$ a $P \cup Y$.

Corollario 21.2. *Sia $P \subset Q$. Allora*

$$|[P, Q]| = 2^{|Q \setminus P|} = 2^{|Q| - |P|}$$

Dimostrazione. Ciò segue direttamente dalla nota 21.1.

Nota 21.3. Siano $P \subset Q$ con $P = \{x_1, \dots, x_p\}$, $X \setminus Q = \{y_1, \dots, y_r\}$. Assumiamo che gli x_j e y_k siano tutti distinti tra di loro. Sappiamo che

$$\begin{aligned} [P, Q] &= P^+(X \setminus Q)^- \\ &= x_1^+ \cdots x_p^+ y_1^- \cdots y_r^- \end{aligned}$$

Gli elementi di X che non appaiono in questa rappresentazione sono esattamente gli elementi $x_{p+1}, \dots, x_q$ (che assumiamo tutti distinti) di $Q \setminus P$. All'intervallo $[P, Q]$ corrisponde nella tabella una riga

$$\underbrace{1 \dots 1}_{x_1 \cdots x_p} \quad \underbrace{* \dots *}_{x_{p+1} \cdots x_q} \quad \underbrace{0 \dots 0}_{y_1 \cdots y_r}$$

In ciascuna delle $q - p$ posizioni con * si può scegliere sia 0 che 1 con in tutto 2^{q-p} possibilità; ciò fornisce una seconda dimostrazione del corollario 21.2.

Nota 21.4.

(1) Sappiamo dal corollario 21.2 e dalla nota 21.3 che il numero degli elementi di un intervallo di 2^n è una potenza 2^d di 2; in questo caso diciamo che l'intervallo possiede la dimensione d. Come si vede dalla nota 21.3 un intervallo d-dimensionale si ottiene scegliendo $n - d$ indici e poi per ciascuno di questi $n - d$ indici j una delle funzioni j^+ o j^- (se assumiamo per semplicità che $X = n$). Ogni scelta determina univocamente un intervallo e viceversa.

(2) Esistono quindi $\binom{n}{d} 2^{n-d}$ intervalli d-dimensionali di 2^n.

(3) Il numero di tutti gli intervalli di 2^n è $\sum_{d=0}^{n} \binom{n}{d} 2^{n-d} = (1+2)^n = 3^n$.

(4) I vertici sono gli intervalli 0-dimensionali; il loro numero è uguale a $\binom{n}{0} 2^n = 2^n$, correttamente.

(5) Gli spigoli sono gli intervalli 1-dimensionali; il loro numero è uguale a $\binom{n}{1} 2^{n-1} = n 2^{n-1}$.

(6) Il numero degli intervalli di dimensione $n - 1$ è uguale a $\binom{n}{n-1} 2 = 2n$.

Esercizi

Esercizio 21.5. Nella tabella sia f_d il numero degli intervalli d-dimensionali. Verificare, mediante le formule della nota 21.4 e, per $n \leq 4$, nelle figure, i seguenti casi speciali.

n	f_0	f_1	f_2	f_3	f_4	f_5
0	1	0	0	0	0	0
1	2	1	0	0	0	0
2	4	4	1	0	0	0
3	8	12	6	1	0	0
4	16	32	24	8	1	0
5	32	80	80	40	10	1

Per il calcolo delle facce con R possiamo usare la seguente funzione:

```
Mfb.facce = function (n,d)
choose(n,d)*2^(n-d)
```

Con `choose` in R si ottengono i numeri binomiali. Con questa funzione calcoliamo ad esempio

n	f_4	f_7	f_{12}
10	13440	960	0
15	2795520	1647360	3640
20	317521920	635043840	32248320

Questi numeri enormi sono un'ulteriore dimostrazione della grande complessità e ricchezza delle funzioni booleane.

Esercizio 21.6. α sia data dalla tabella

1	2	3	4	α
0	0	0	0	0
0	0	0	1	1
0	0	1	0	1
0	1	0	0	0
1	0	0	0	0
0	0	1	1	1
0	1	0	1	1
1	0	0	1	0
0	1	1	0	0
1	0	1	0	1
1	1	0	0	1
0	1	1	1	0
1	0	1	1	1
1	1	0	1	1
1	1	1	0	1
1	1	1	1	1

Ridisegnare a mano una delle due figure nella prima colonna di questa pagina e colorare con colori diversi gli insiemi

$$\begin{aligned} \rho &:= \{1110, 1111, 1101, 1100\} \\ \sigma &:= \{1011, 0011, 1010, 0010\} \\ \tau &:= \{0001, 0101\} \end{aligned}$$

È abbastanza chiaro che ρ, σ e τ sono implicanti massimali di α e che $\alpha = \rho \cup \sigma \cup \tau$ è una rappresentazione minimale di α nel senso (non perfettamente precisato) della nota 19.6. Come abbiamo osservato allora, in altri casi però può essere molto difficile trovare una rappresentazione minimale.

LABORATORIO DI PROGRAMMAZIONE

Corso di laurea in matematica — Anno accademico 2004/05 — Numero 7

Caratteri di gruppi abeliani finiti

Situazione 22.1. $(G,+)$ sia un gruppo abeliano finito. Con 0 denotiamo l'elemento neutro di G. Identifichiamo 2 con $\mathbb{Z}/2$. Poniamo

$$\varepsilon := \varepsilon_G := e^{\frac{2\pi i}{|G|}}$$

ε è una $|G|$-esima radice dell'unità e genera il gruppo moltiplicativo delle $|G|$-esime radici dell'unita. Con S^1 denotiamo come sempre la circonferenza unitaria, cioè

$$S^1 = \{z \in \mathbb{C} \mid |z| = 1\}$$

Definizione 22.2. Un *carattere* di G è un'applicazione $\chi : G \longrightarrow \mathbb{C}$ tale che

$$\chi(0) = 1$$
$$\chi(a+b) = \chi(a)\chi(b)$$

per ogni $a, b \in G$.

Nota 22.3. $\chi : G \longrightarrow \mathbb{C}$ sia un'applicazione tale che $\chi(a+b) = \chi(a)\chi(b)$ per ogni $a, b \in G$. Se $\chi \neq 0$ (cioè se esiste almeno un $a \in G$ per cui $\chi(a) \neq 0$), allora χ è un carattere di G. Spesso è ovvio che $\chi \neq 0$ e quindi non dobbiamo dimostrare appositamente che $\chi(0) = 1$.

Dimostrazione. Sia $\chi(a) \neq 0$ per qualche $a \in G$. Allora $\chi(a) = \chi(a+0) = \chi(a)\chi(0)$ e vediamo che necessariamente $\chi(0) = 1$.

Nota 22.4. Siano χ un carattere di G ed $a \in G$. Allora $\chi(a)$ è una $|G|$-esima radice dell'unità; in particolare $\chi(a) \in S^1$. χ può quindi essere considerato come un omomorfismo di gruppi $G \longrightarrow (S^1, \cdot)$. Ciò implica inoltre che

$$\chi(-a) = \frac{1}{\chi(a)} = \overline{\chi(a)}$$

Dimostrazione. Per ipotesi $|G| < \infty$, per cui $|G|a = 0$. Perciò

$$(\chi(a))^{|G|} = \chi(|G|a) = \chi(0) = 1$$

La relazione $\chi(-a) = \frac{1}{\chi(a)}$ segue adesso perché χ è un omomorfismo di gruppi. Per $z \in S^1$ si ha inoltre $\frac{1}{z} = \overline{z}$.

Definizione 22.5. L'insieme $\hat{G}$ di tutti i caratteri di G è detto *gruppo dei caratteri* o *gruppo duale* di G.

$\hat{G}$ è infatti un gruppo abeliano con il prodotto (scritto in modo moltiplicativo) definito da

$$\chi\varphi := \bigcirc_x \chi(x)\varphi(x)$$

per $\chi, \varphi \in \hat{G}$; cfr. esercizio 22.6. Inoltre

$$\chi_0 := \bigcirc_x 1$$

è l'elemento neutro di $\hat{G}$ e

$$\chi^{-1} = \bigcirc_x \frac{1}{\chi(x)} = \bigcirc_x \overline{\chi(x)} = \overline{\chi}$$

per $\chi \in \hat{G}$. χ^{-1} naturalmente non è l'inversa di un'applicazione biiettiva, ma l'inverso nel gruppo $\hat{G}$.

Esercizio 22.6. Siano $\chi, \varphi \in \hat{G}$. Allora $\chi\varphi \in \hat{G}$ e $\overline{\chi} \in \hat{G}$.

Il gruppo dei caratteri di un gruppo ciclico

Lemma 22.7. *G sia ciclico e g un generatore di G. Allora l'applicazione $\chi_1 : G \longrightarrow \mathbb{C}$ definita da*

$$\chi_1(ng) := \varepsilon^n$$

per $n \in \mathbb{Z}$ è ben definita e un carattere di G. In particolare abbiamo $\chi_1(g) = \varepsilon$.

Si noti che χ_1 dipende dalla scelta del generatore g; solo nel caso $|G| \leq 2$ abbiamo un unico generatore e quindi anche χ_1 è univocamente determinato.

Dimostrazione. (1) Dimostriamo prima che l'applicazione χ_1 è ben definita. Ogni elemento di G è della forma ng per qualche $n \in \mathbb{Z}$. Sia $ng = mg$. Allora $m = n + j|G|$ per qualche $j \in \mathbb{Z}$, per cui

$$\varepsilon^m = \varepsilon^{n+j|G|} = \varepsilon^n.$$

(2) Dimostriamo che χ_1 soddisfa le condizioni della definizione 22.2.

$$\chi_1(0) = \varepsilon^0 = 1$$
$$\chi_1(ng+mg) = \chi_1((n+m)g) = \varepsilon^{n+m} = \varepsilon^n\varepsilon^m = \chi_1(ng)\chi_1(mg)$$

Proposizione 22.8. *G sia ciclico e g un generatore fissato di G. χ_1 sia definito come nel lemma 22.7. Allora i caratteri $\chi_0, \chi_1, \chi_1^2, \ldots, \chi_1^{|G|-1}$ sono tutti distinti tra di loro e*

$$\hat{G} = \{\chi_0, \chi_1, \chi_1^2, \ldots, \chi_1^{|G|-1}\}$$

$\hat{G}$ è quindi un gruppo ciclico dello stesso ordine di G, generato da χ_1. Notiamo di nuovo che χ_1 dipende dalla scelta del generatore g. Naturalmente $\chi_1 = \chi_0$ se $G = 0$.

Dimostrazione. (1) I numeri $\chi_0(g) = 1$, $\chi_1(g) = \varepsilon$, $\chi_1^2(g) = \varepsilon^2$,..., $\chi_1^{|G|-1} = \varepsilon^{|G|-1}$ sono tutti distinti tra di loro, perciò lo sono anche le applicazioni $\chi_0, \chi_1, \chi_1^2, \ldots, \chi_1^{|G|-1}$; queste sono caratteri come si vede dall'esercizio 22.6.

(2) Dobbiamo ancora dimostrare che ogni carattere di G è uno dei caratteri elencati. Sia $\chi \in \hat{G}$. Dalla nota 22.4 sappiamo che $\chi(g)$ è una $|G|$-esima radice dell'unità, e quindi esiste $k \in \{0, 1, \ldots, |G|-1\}$ tale che $\chi(g) = \varepsilon^k$. Ma allora $\chi = \chi_1^k$.

Corollario 22.9. *G sia ciclico. Allora $G \cong \hat{G}$. Questo isomorfismo però non è canonico, perché dipende dalla scelta del generatore g, tranne nel caso $|G| \leq 2$ in cui esiste un solo generatore.*

In questo numero

I caratteri di 2 e di $\mathbb{Z}/4$

Esempio 22.10. Sia $G = 2$. Allora $\varepsilon = -1$. L'unico generatore di 2 è 1 e con il procedimento della proposizione 22.8 otteniamo il carattere χ_1 definito da $\chi_1(1) = -1$.

2 possiede perciò esattamente i due caratteri χ_0 e χ_1 con i valori dati dalla tabella

	0	1
χ_0	1	1
χ_1	1	-1

È facile verificare già adesso che con

$$0 \mapsto \chi_0$$
$$1 \mapsto \chi_1$$

si ottiene un isomorfismo tra 2 e $\hat{2}$.

Nota 22.11. Sia $a + a = 0$ per ogni $a \in G$ (dimostreremo nel prossimo numero che ciò accade se e solo se $G \cong 2^n$). Allora ogni carattere di G assume solo i valori 1 e -1 e può quindi essere considerato come un omomorfismo di gruppi $G \longrightarrow (\{1,-1\}, \cdot)$.

Dimostrazione. Per $\chi \in \hat{G}$ ed $a \in G$

$$(\chi(a))^2 = \chi(a+a) = \chi(0) = 1$$

$\chi(a)$ è perciò una seconda radice dell'unità e ciò implica che $\chi(a) = \pm 1$.

Esempio 22.12. Sia $G = \mathbb{Z}/4 = \{0,1,2,3\}$. In questo caso $\varepsilon = i$. Scegliamo 1 come generatore di G (ma lo sarebbe anche 3); allora $\hat{G} = \{\chi_0, \chi_1, \chi_2, \chi_3\}$ con $\chi_k(1) = i^k$. Si ottiene così la tabella dei caratteri

	0	1	2	3
χ_0	1	1	1	1
χ_1	1	i	-1	$-i$
χ_2	1	-1	1	-1
χ_3	1	$-i$	-1	i

Nota 22.13. Nella teoria delle funzioni booleane utilizzeremo soprattutto il gruppo duale di 2^n. Vedremo che si ha un isomorfismo naturale tra $\widehat{2^n}$ e $(\hat{2})^n$ e quindi essenzialmente è sufficiente conoscere $\hat{2}$.

Si osservi che i gruppi $2^2 = \mathbb{Z}/2 \times \mathbb{Z}/2$ e $\mathbb{Z}/4$ non sono isomorfi.

Relazioni di ortogonalità

Proposizione 23.1. *Per ogni $\chi \in \hat{G}$ vale*

$$\sum_{x\in G} \chi(x) = |G| \cdot (\chi = \chi_0)$$

In altre parole:

$$\sum_{x\in G} \chi(x) = \begin{cases} |G| & \text{se } \chi = \chi_0 \\ 0 & \text{se } \chi \neq \chi_0 \end{cases}$$

Dimostrazione. Per $\chi = \chi_0$ l'enunciato è ovvio. Sia $\chi \neq \chi_0$.

Ciò significa che esiste un $y \in G$ tale che $\chi(y) \neq 1$. Quindi

$$\sum_{x\in G} \chi(x) = \sum_{x\in G} \chi(x+y) = \sum_{x\in G} \chi(x)\chi(y) = \chi(y) \sum_{x\in G} \chi(x)$$

Questa è una relazione tra numeri complessi e per ipotesi $\chi(y) \neq 1$, perciò necessariamente $\sum_{x\in G} \chi(x) = 0$.

Teorema 23.2. *Siano $\chi, \varphi \in \hat{G}$. Allora*

$$\sum_{x\in G} \chi(x)\overline{\varphi(x)} = |G| \cdot (\chi = \varphi)$$

In altre parole:

$$\sum_{x\in G} \chi(x)\overline{\varphi(x)} = \begin{cases} |G| & \text{se } \chi = \varphi \\ 0 & \text{se } \chi \neq \varphi \end{cases}$$

Queste sono le relazioni di ortogonalità per i caratteri.

Dimostrazione. Il carattere $\chi\overline{\varphi} = \chi\varphi^{-1}$ è uguale a χ_0 se e solo se $\chi = \varphi$.

Nota 23.3. L'insieme $\mathbb{C}^G$ di tutte le applicazioni $G \longrightarrow \mathbb{C}$, con le operazioni vettoriali definite componente per componente, è uno spazio vettoriale su $\mathbb{C}$ di dimensione uguale a $|G|$. In $\mathbb{C}^G$ si può definire un *prodotto scalare* ponendo

$$\|f, g\| := \sum_{x\in G} f(x)\overline{g(x)}$$

per $f, g \in \mathbb{C}^G$, da cui con $\|f\| := \sqrt{\|f, f\|}$ si ottiene una norma per la quale

$$\|f\|^2 = \sum_{x\in G} |f(x)|^2$$

Ovviamente $\hat{G} \subset \mathbb{C}^G$. Per il teorema 23.2 i caratteri di G sono fra di loro ortogonali rispetto a questo prodotto scalare e perciò linearmente indipendenti su $\mathbb{C}$.

Ma $\dim \mathbb{C}^G = |G|$, e ciò implica che

$$|\hat{G}| \leq |G|$$

Dimostreremo adesso che in verità si ha $|\hat{G}| = |G|$ e che quindi i caratteri formano una base (ortogonale, ma non ortonormale) di $\mathbb{C}^G$.

Il teorema di estensione

Osservazione 23.4. *Siano H un sottogruppo di G ed $y \in G$. Allora $H + \mathbb{Z}y$ è il sottogruppo di G generato da H e da y.*

Lemma 23.5. *Siano H un sottogruppo di G ed $y \in G$. Sia*

$$\lambda := \min\{n \in \mathbb{N} + 1 \mid ny \in H\}$$

Siccome $|G|y = 0 \in H$, λ è ben definito. Chiaramente $y \in H \iff \lambda = 1$.

Ogni $a \in H + \mathbb{Z}y$ possiede una rappresentazione della forma

$$a = h + ry$$

con $h \in H$ ed $r \in \{0, 1, \ldots, \lambda - 1\}$ univocamente determinati.

Dimostrazione. Consideriamo il gruppo quoziente

$$Q := \frac{H + \mathbb{Z}y}{H}$$

e denotiamo con $[a]$ la classe di equivalenza di $a \in G$. È chiaro che $[y]$ è un generatore di Q e che $\lambda = |Q|$. Per ogni $a \in G$ abbiamo una rappresentazione $[a] = r[y] = [ry]$ con $0 \leq r < \lambda$ univocamente determinati e quindi $a = ry + h$ con $h \in H$. Unicità di h: Esercizio, simile alla dimostrazione della proposizione 2.17; naturalmente bisogna anche usare che H è un sottogruppo di G.

Lemma 23.6. *Siano H un sottogruppo di G e $\varphi \in \hat{H}$. Sia $y \in G$.*
Allora esiste $\chi \in \widehat{H + \mathbb{Z}y}$ tale che $(\chi = \varphi, in\ H)$.

Dimostrazione. Come nel lemma 23.5 sia

$$\lambda := \min\{n \in \mathbb{N} + 1 \mid ny \in H\}$$

In particolare $\lambda y \in H$ e quindi $\varphi(\lambda y) \in S^1$ è definito, ad esempio $\varphi(\lambda y) = e^{2\pi i\alpha}$ con $\alpha \in \mathbb{R}$. Sia $\tau := e^{\frac{2\pi i\alpha}{\lambda}}$. Allora $\varphi(\lambda y) = \tau^\lambda$.

Per $h \in H$ e $0 \leq r < \lambda - 1$ sia adesso

$$\chi(h + ry) := \tau^r \cdot \varphi(h)$$

Per il lemma 23.5 otteniamo un'applicazione $\chi : H + \mathbb{Z}y \longrightarrow S^1$ ben definita.
È chiaro che $(\chi = \varphi, \text{in } H)$.

Dobbiamo però ancora dimostrare che χ è un carattere.

Siano $a = h + ry$ e $b = k + sy$ con $h, k \in H$ ed $r, s \in \{0, 1, \ldots, \lambda - 1\}$.

(1) Sia $0 \leq r + s < \lambda$. In questo caso

$$\begin{aligned}\chi(a+b) &= \chi(h + k + (r+s)y) \\ &= \tau^{r+s}\varphi(h+k) = \tau^r\varphi(h)\tau^s\varphi(k) \\ &= \chi(a)\chi(b)\end{aligned}$$

(2) Altrimenti $r + s = \lambda + \rho$ con $0 \leq \rho \leq \lambda - 2 < \lambda$. Allora, siccome $\lambda y \in H$,

$$\begin{aligned}\chi(a+b) &= \chi(h + k + \lambda y + \rho y) \\ &= \varphi(h + k + \lambda y)\tau^\rho \\ &= \varphi(h)\varphi(k)\varphi(\lambda y)\tau^\rho \\ &= \tau^\lambda\tau^\rho\varphi(h)\varphi(k)\end{aligned}$$

mentre

$$\begin{aligned}\chi(a)\chi(b) &= \tau^r\varphi(h)\tau^s\varphi(k) \\ &= \tau^{r+s}\varphi(h)\varphi(k)\end{aligned}$$

Ma $\tau^{r+s} = \tau^{\lambda+\rho} = \tau^\lambda\tau^\rho$, perciò anche in questo caso $\chi(a+b) = \chi(a)\chi(b)$.

Teorema 23.7. *Siano H un sottogruppo di G e $\varphi \in \hat{H}$. Allora esiste $\chi \in \hat{G}$ tale che $(\chi = \varphi, in\ H)$.*

Dimostrazione. Siccome G è finito, esistono $y_1, \ldots, y_m \in G$ tali che

$$G = H + \mathbb{Z}y_1 + \ldots + \mathbb{Z}y_m$$

Per il lemma 23.6 possiamo estendere φ prima da H ad $H + \mathbb{Z}y_1$, poi da $H + \mathbb{Z}y_1$ ad $H + \mathbb{Z}y_1 + \mathbb{Z}y_2$, ecc.

L'isomorfismo canonico $G \cong \hat{\hat{G}}$

Proposizione 23.8. *Per ogni elemento $x \neq 0$ di G esiste $\chi \in \hat{G}$ tale che $\chi(x) \neq 1$.*

Dimostrazione. Sia $H = \mathbb{Z}x$ il gruppo generato da x. H è ciclico ed $|H| \geq 2$, poiché $x \neq 0$. Se applichiamo la proposizione 22.8 ad H, otteniamo un carattere $\varphi \in \hat{H}$ che corrisponde al carattere χ_1 di quella proposizione e per il quale quindi $\varphi(x) \neq 1$.

Per il teorema 23.7 possiamo estendere φ a un carattere $\chi \in \hat{G}$ e naturalmente $\chi(x) = \varphi(x) \neq 1$.

Proposizione 23.9. *Siano $x, y \in G$ ed $x \neq y$. Allora esiste $\chi \in \hat{G}$ con $\chi(x) \neq \chi(y)$.*

Dimostrazione. L'ipotesi implica $x - y \neq 0$. Per la proposizione 23.8 esiste $\chi \in \hat{G}$ con $\chi(x-y) \neq 1$. Ma $\chi(x-y) = \dfrac{\chi(x)}{\chi(y)}$ e vediamo che $\chi(x) \neq \chi(y)$.

Definizione 23.10. Per ogni $a \in G$ sia $\aleph_a : \hat{G} \longrightarrow S^1$ l'applicazione definita da

$$\aleph_a(\chi) := \chi(a)$$

per ogni $\chi \in \hat{G}$.

Osservazione 23.11. *Per ogni $a \in G$ si ha $\aleph_a \in \hat{\hat{G}}$, cioè $\aleph_a$ è un carattere di $\hat{G}$. Dalla definizione 23.10 otteniamo così un'applicazione $G \longrightarrow \hat{\hat{G}}$.*

Dimostrazione. Siano $\chi, \varphi \in \hat{G}$. Allora

$$\begin{aligned}\aleph_a(\chi\varphi) &= (\chi\varphi)(a) = \chi(a)\varphi(a) \\ &= \aleph_a(\chi)\aleph_a(\varphi)\end{aligned}$$

Teorema 23.12. *L'applicazione $\aleph : G \longrightarrow \hat{\hat{G}}$ è un isomorfismo di gruppi.*

Dimostrazione. (1) Siano $a, b \in G$ e $\chi \in \hat{G}$. Allora

$$\begin{aligned}\aleph_{a+b}(\chi) &= \chi(a+b) = \chi(a)\chi(b) \\ &= \aleph_a(\chi)\aleph_b(\chi) = (\aleph_a\aleph_b)(\chi)\end{aligned}$$

(2) La proposizione 23.9 implica che $\aleph$ è iniettiva.

(3) Per la nota 23.3 $|\hat{\hat{G}}| \leq |\hat{G}| \leq |G|$ e quindi $\aleph$ deve essere biiettiva.

Corollario 23.13. $|\hat{G}| = |G|$.

Dimostrazione. Per il teorema 23.12 e la nota 23.3 abbiamo

$$|G| = |\hat{\hat{G}}| \leq |\hat{G}| \leq |G|$$

Nota 23.14. Il corollario 23.13 non implica ancora che $G \cong \hat{G}$. Per dimostrare questa isomorfia dovremo prima dimostrare il teorema di struttura per i gruppi abeliani finiti che, nel nostro approccio, sarà infatti una conseguenza della teoria dei caratteri.

Corollario 23.15. *I caratteri formano una base ortogonale di $\mathbb{C}^G$.*

Dimostrazione. Ciò segue dal corollario 23.13, come visto nella nota 23.3.

Vedremo adesso che l'analisi armonica su un gruppo abeliano finito si deduce dalla rappresentazione di una funzione arbitraria in $\mathbb{C}^G$ come combinazione lineare degli elementi della base ortogonale costituita dai caratteri e si inquadra quindi nella teoria dei sistemi di funzioni ortogonali dell'analisi.

L'identità di uguaglianza

Proposizione 24.1. *Per ogni $a \in G$ si ha*

$$\sum_{\chi\in\hat G}\chi(a) = |G|\cdot(a=0)$$

In altre parole

$$\sum_{\chi\in\hat G}\chi(a) = \begin{cases}|G| & \text{se } a=0\\ 0 & \text{se } a\neq 0\end{cases}$$

Dimostrazione. Ciò segue direttamente dalla proposizione 23.1 e dal teorema 23.12.

Teorema 24.2. *Siano $a, b \in G$. Allora*

$$\sum_{\chi\in\hat G}\chi(a)\overline{\chi(b)} = |G|\cdot(a=b)$$

In altre parole

$$\sum_{\chi\in\hat G}\chi(a)\overline{\chi(b)} = \begin{cases}|G| & \text{se } a=b\\ 0 & \text{se } a\neq b\end{cases}$$

Dimostrazione. $a=b \iff a-b=0$.

Nota 24.3. L'identità del teorema 24.2 può essere anche letta nella direzione opposta,

$$(a=b) = \frac{1}{|G|}\sum_{\chi\in\hat G}\chi(a)\overline{\chi(b)}$$

fornendo quindi una *formula* per l'uguaglianza fra due elementi di G che è spesso usata ad esempio nel calcolo del numero di soluzioni di equazioni complicate del calcolo combinatorio e della teoria dei numeri.

La trasformata di Fourier

Lemma 24.4. *V sia uno spazio vettoriale su $\mathbb{C}$ con $\dim V =: n < \infty$. $\|\|$ sia un prodotto scalare su V ed $e_1, \ldots, e_n$ una base (ordinata) ortogonale (rispetto a questo prodotto scalare) di V.*

Per ogni $v \in V$ si ha allora lo sviluppo di Fourier

$$v = \alpha_1 e_1 + \ldots + \alpha_n e_n$$

con $\alpha_k = \dfrac{\|v, e_k\|}{\|e_k, e_k\|}$ per ogni k.

Se esiste una costante (necessariamente reale e $\neq 0$) C tale che $\|e_k, e_k\| = C$ per ogni k, e se poniamo $\|v\| = \sqrt{\|v, v\|}$, allora abbiamo

$$\|v\|^2 = C\sum_{k=1}^{n}|\alpha_k|^2$$

Dimostrazione. (1) Abbiamo

$$v = \alpha_1 e_1 + \ldots + \alpha_n e_n$$

con $\alpha_1, \ldots, \alpha_n \in \mathbb{C}$. Allora $\|v, e_1\| = \alpha_1\|e_1, e_1\|$ e quindi $\alpha_1 = \dfrac{\|v, e_1\|}{\|e_1, e_1\|}$.

Analoghe espressioni si trovano per gli altri α_k.

(2) Sia adesso $\|e_k, e_k\| = C$ per ogni k. Dall'ortogonalità abbiamo

$$\|v\|^2 = \|\sum_{k=1}^{n}\alpha_k e_k, \sum_{k=1}^{n}\alpha_k e_k\|$$

$$= \sum_{k=1}^{n}\alpha_k\overline{\alpha_k}\|e_k, e_k\|$$

$$= C\sum_{k=1}^{n}\alpha_k\overline{\alpha_k} = C\sum_{k=1}^{n}|\alpha_k|^2$$

Corollario 24.5. *Per ogni funzione $f \in \mathbb{C}^G$ vale lo sviluppo di Fourier*

$$f = \frac{1}{|G|}\sum_{\chi\in\hat G}\|f,\chi\|\chi$$

Dimostrazione. Ciò segue dal lemma 24.4 perché $\|\chi, \chi\| = |G|$ per ogni $\chi \in \hat G$.

Definizione 24.6. La *trasformata di Fourier* di $f \in \mathbb{C}^G$ è la funzione $\hat f : \hat G \longrightarrow \mathbb{C}$ che si ottiene ponendo

$$\hat f(\chi) := \|f, \chi\|$$

per $\chi \in \hat G$. Abbiamo in altre parole

$$\hat f(\chi) = \sum_{x\in G} f(x)\overline{\chi(x)}$$

per ogni $\chi \in G$, mentre dal corollario 24.5 segue la formula di inversione

$$f(x) = \frac{1}{|G|}\sum_{\chi\in\hat G}\hat f(\chi)\chi(x)$$

per ogni $x \in G$ e quindi

$$f = \frac{1}{|G|}\sum_{\chi\in\hat G}\hat f(\chi)\chi$$

Osservazione 24.7. *Sia $f \in \mathbb{C}^G$. Allora*

$$\hat f(\chi_0) = \sum_{x\in G} f(x)$$

$$f(0) = \frac{1}{|G|}\sum_{\chi\in\hat G}\hat f(\chi)$$

Osservazione 24.8. *La trasformata di Fourier, considerata come applicazione*

$$\begin{aligned}\mathbb{C}^G &\longrightarrow \mathbb{C}^{\hat G}\\ f &\longmapsto \hat f\end{aligned}$$

è un isomorfismo di spazi vettoriali su $\mathbb{C}$.

Dimostrazione. La linearità è immediata e la biiettività segue dalla formula di inversione.

Osservazione 24.9. *Il prodotto scalare della nota 23.3 su $\mathbb{C}^{\hat G}$ diventa*

$$\|A, B\| := \sum_{\chi\in\hat G}A(\chi)\overline{B(\chi)}$$

per $A, B \in \mathbb{C}^{\hat G}$. Con $\|A\| := \sqrt{\|A, A\|}$ otteniamo una norma per cui

$$\|A\|^2 = \sum_{\chi\in\hat G}|A(\chi)|^2$$

Proposizione 24.10. *Sia $f \in \mathbb{C}^G$. Allora*

$$\|f\|^2 = \frac{1}{|G|}\|\hat f\|^2$$

Questa è l'identità di Plancherel, che può essere scritta anche nella forma

$$\sum_{x\in G}|f(x)|^2 = \frac{1}{|G|}\sum_{\chi\in\hat G}|\hat f(\chi)|^2$$

Dimostrazione. Per la nota 23.3 abbiamo $\|\chi, \chi\| = |G|$ per ogni $\chi \in \hat G$, possiamo quindi applicare il lemma 24.4 al corollario 24.5 con $C = |G|$, ottenendo

$$\|f\|^2 = |G|\sum_{\chi\in\hat G}\left|\frac{\hat f(\chi)}{|G|}\right|^2 = \frac{1}{|G|}\sum_{\chi\in\hat G}|\hat f(\chi)|^2$$

La convoluzione

Definizione 24.11. Siano $f, g \in \mathbb{C}^G$.
La *convoluzione* $f * g \in \mathbb{C}^G$ è definita da

$$(f*g)(x) := \sum_{a\in G} f(a)g(x-a)$$

Teorema 24.12. *Siano $f, g \in \mathbb{C}^G$. Allora*

$$\widehat{f*g} = \hat f\hat g$$

Dimostrazione. Sia $\chi \in \hat G$. Allora

$$\begin{aligned}\widehat{f*g}(\chi) &= \sum_{x\in G}(f*g)(x)\overline{\chi(x)}\\ &= \sum_{x\in G}\sum_{a\in G} f(a)g(x-a)\overline{\chi(x)}\\ &= \sum_{x\in G}\sum_{a\in G} f(a)g(x-a)\overline{\chi(a)}\,\overline{\chi(x-a)}\\ &= \sum_{a\in G} f(a)\overline{\chi(a)}\sum_{x\in G} g(x-a)\overline{\chi(x-a)}\\ &= \hat f(\chi)\hat g(\chi)\end{aligned}$$

Le formule per un gruppo ciclico

Nota 24.13. G sia ciclico e g un generatore fissato. Come nel lemma 22.7 e nella proposizione 22.8 definiamo i caratteri $\chi_0, \chi_1, \ldots, \chi_{|G|-1}$ con $\chi_k = \chi_1^k$ e quindi

$$\chi_k(g) = \varepsilon^k, \quad \overline{\chi_k(g)} = \varepsilon^{-k}$$

cosicché

$$\chi_k(ng) = \varepsilon^{nk}, \quad \overline{\chi_k(ng)} = \varepsilon^{-nk}$$

per ogni k. Sappiamo che

$$\hat G = \{\chi_0, \chi_1, \ldots, \chi_{|G|-1}\}$$

e naturalmente, scrivendo $x_n := ng$,

$$G = \{x_0, x_1, \ldots, x_{|G|-1}\}$$

Applicando il corollario 24.5 abbiamo

$$\begin{aligned}\hat f(\chi_k) &= \sum_{n=0}^{|G|-1} f(x_n)\overline{\chi_k(x_n)}\\ &= \sum_{n=0}^{|G|-1} f(x_n)\varepsilon^{-nk}\end{aligned}$$

$$\begin{aligned}|G|f(x_n) &= \sum_{k=0}^{|G|-1}\hat f(\chi_k)\chi_k(x_n)\\ &= \sum_{k=0}^{|G|-1}\hat f(\chi_k)\varepsilon^{nk}\end{aligned}$$

Le formule per $\mathbb{Z}/2, \mathbb{Z}/4$ e $\mathbb{Z}/3$

Esempio 25.1. Sia $G = 2$. In questo caso $\varepsilon = -1$. L'unico generatore di G è 1. Come nell'esempio 22.10 abbiamo la tabella dei caratteri

	0	1
χ_0	1	1
χ_1	1	-1

Siccome entrambi i caratteri sono reali, per $f \in \mathbb{C}^2$ abbiamo

$$\hat{f}(\chi) = \sum_{x \in 2} f(x)\overline{\chi(x)} = \sum_{x \in 2} f(x)\chi(x)$$

per ogni $\chi \in \hat{2}$ e quindi

$$\hat{f}(\chi_0) = f(0) + f(1)$$
$$\hat{f}(\chi_1) = f(0) - f(1)$$

percorrendo nella sommazione ogni volta una riga della tabella, e

$$2f(x) = \sum_{\chi \in \hat{2}} \hat{f}(\chi)\chi(x)$$

e quindi

$$2f(0) = \hat{f}(\chi_0) + \hat{f}(\chi_1)$$
$$2f(1) = \hat{f}(\chi_0) - \hat{f}(\chi_1)$$

percorrendo nella sommazione ogni volta una colonna della tabella. Si osservi che la seconda coppia di equazioni è anche visibilmente una conseguenza della prima.

Nel caso $G = 2^n$ la trasformata di Fourier si chiama anche *trasformata di Walsh*. Abbiamo così visto il primo esempio di una trasformata di Walsh.

Esempio 25.2. Sia $G = \mathbb{Z}/4 = \{0, 1, 2, 3\}$. In questo caso $\varepsilon = i$. Come nell'esempio 22.12 abbiamo la tabella dei caratteri (numerati in modo non canonico)

	0	1	2	3
χ_0	1	1	1	1
χ_1	1	i	-1	$-i$
χ_2	1	-1	1	-1
χ_3	1	$-i$	-1	i

e la tabella dei caratteri coniugati

	0	1	2	3
$\overline{\chi_0}$	1	1	1	1
$\overline{\chi_1}$	1	$-i$	-1	i
$\overline{\chi_2}$	1	-1	1	-1
$\overline{\chi_3}$	1	i	-1	$-i$

Per $f \in \mathbb{C}^G$ abbiamo

$$f(\chi) = \sum_{x \in G} f(x)\overline{\chi(x)}$$

per ogni $\chi \in \hat{G}$ e quindi

$$\hat{f}(\chi_0) = f(0) + f(1) + f(2) + f(3)$$
$$\hat{f}(\chi_1) = f(0) - f(1)i - f(2) + f(3)i$$
$$\hat{f}(\chi_2) = f(0) - f(1) + f(2) - f(3)$$
$$\hat{f}(\chi_3) = f(0) + f(1)i - f(2) - f(3)i$$

percorrendo nella sommazione ogni volta una riga della tabella dei caratteri coniugati, e

$$4f(0) = \hat{f}(\chi_0) + \hat{f}(\chi_1) + \hat{f}(\chi_2) + \hat{f}(\chi_3)$$
$$4f(1) = \hat{f}(\chi_0) + \hat{f}(\chi_1)i - \hat{f}(\chi_2) - \hat{f}(\chi_3)i$$
$$4f(2) = \hat{f}(\chi_0) - \hat{f}(\chi_1) + \hat{f}(\chi_2) - \hat{f}(\chi_3)$$
$$4f(3) = \hat{f}(\chi_0) - \hat{f}(\chi_1)i - \hat{f}(\chi_2) + \hat{f}(\chi_3)i$$

percorrendo nella sommazione ogni volta una colonna della tabella dei caratteri.

Esempio 25.3. Sia $G = \mathbb{Z}/3 = \{0, 1, 2\}$. In questo caso $\varepsilon = e^{\frac{2\pi i}{3}}$. Come negli esempi precedenti con l'aiuto della proposizione 22.8 otteniamo le tabelle dei caratteri (numerati in modo non canonico) e dei caratteri coniugati:

	0	1	2
χ_0	1	1	1
χ_1	1	ε	ε^2
χ_2	1	ε^2	ε

	0	1	2
$\overline{\chi_0}$	1	1	1
$\overline{\chi_1}$	1	ε^2	ε
$\overline{\chi_2}$	1	ε	ε^2

Per $f \in \mathbb{C}^G$ abbiamo perciò

$$\hat{f}(\chi_0) = f(0) + f(1) + f(2)$$
$$\hat{f}(\chi_1) = f(0) + f(1)\varepsilon^2 + f(2)\varepsilon$$
$$\hat{f}(\chi_2) = f(0) + f(1)\varepsilon + f(2)\varepsilon^2$$

percorrendo nella sommazione ogni volta una riga della tabella dei caratteri coniugati, e

$$3f(0) = \hat{f}(\chi_0) + \hat{f}(\chi_1) + \hat{f}(\chi_2)$$
$$3f(1) = \hat{f}(\chi_0) + \hat{f}(\chi_1)\varepsilon + \hat{f}(\chi_2)\varepsilon^2$$
$$3f(2) = \hat{f}(\chi_0) + \hat{f}(\chi_1)\varepsilon^2 + \hat{f}(\chi_2)\varepsilon$$

percorrendo nella sommazione ogni volta una colonna della tabella dei caratteri.

Il duale del prodotto

Proposizione 25.4. *Siano G ed H gruppi abeliani finiti. Allora l'applicazione*

$$\begin{aligned} \hat{G} \times \hat{H} &\longrightarrow \widehat{G \times H} \\ (\chi, \varphi) &\longmapsto \bigcirc_{(x,y)} \chi(x)\varphi(y) \end{aligned}$$

è ben definita ed un isomorfismo di gruppi. Un elemento $\psi \in \widehat{G \times H}$ corrisponde alla coppia $(\chi, \varphi) \in \hat{G} \times \hat{H}$ con $\chi(x) = \psi(x, 1)$ per $x \in G$ e $\varphi(y) = \psi(1, y)$ per $y \in H$.

Dimostrazione. Facile verifica.

Nota 25.5. G sia prodotto diretto di gruppi ciclici. Allora $\hat{G} \cong G$.

Quando dal teorema di struttura sapremo che ogni gruppo abeliano finito è prodotto diretto di gruppi ciclici, avremo così dimostrato che $\hat{G} \cong G$ per ogni gruppo abeliano finito (benché questo isomorfismo non sia canonico).

Dimostrazione. Proposizione 25.4 e corollario 22.9.

Nota 25.6. G ed H siano gruppi ciclici, g e h generatori fissati di G ed H, ed

$$\varepsilon := \varepsilon_G = e^{\frac{2\pi i}{|G|}}, \quad \delta := \varepsilon_H = e^{\frac{2\pi i}{|H|}}$$

Per $0 \le k < |G|$ e $0 \le l < |H|$ definiamo i caratteri χ_{kl} ponendo

$$\chi_{kl}(g, h) := \varepsilon^k \delta^l$$

per cui

$$\chi_{kl}(mg, nh) = \varepsilon^{mk}\delta^{nl}$$

Per la proposizione 25.4 in questo modo otteniamo ogni carattere di $G \times H$ esattamente una volta.

Gli elementi di $G \times H$ possono invece essere scritti nella forma $x_{mn} := (mg, nh)$. Come nella nota 24.13 per $f \in \mathbb{C}^{G \times H}$ abbiamo allora

$$\hat{f}(\chi_{kl}) = \sum_{m=0}^{|G|-1} \sum_{n=0}^{|H|-1} f(x_{mn})\varepsilon^{-mk}\delta^{-nl}$$

$$f(x_{mn}) = \frac{1}{|G||H|} \sum_{k=0}^{|G|-1} \sum_{l=0}^{|H|-1} \hat{f}(\chi_{kl})\varepsilon^{mk}\delta^{nl}$$

Esempio 25.7. Sia $G = 2^2 = \mathbb{Z}/2 \times \mathbb{Z}/2$. Come generatori nella nota 25.6 possiamo scegliere $g = 1, h = 1$. Inoltre $\varepsilon = \delta = -1$. Otteniamo così

$$\hat{G} = \{\chi_{00}, \chi_{01}, \chi_{10}, \chi_{11}\}$$

con la tabella dei caratteri

	00	01	10	11
χ_{00}	1	1	1	1
χ_{01}	1	-1	1	-1
χ_{10}	1	1	-1	-1
χ_{11}	1	-1	-1	1

che si ottiene applicando la proposizione 25.4 o la nota 25.6 alla tabella dei caratteri di 2. Siccome i caratteri sono tutti reali, otteniamo direttamente le rappresentazioni

$$\hat{f}(\chi_{00}) = f(00) + f(01) + f(10) + f(11)$$
$$\hat{f}(\chi_{01}) = f(00) - f(01) + f(10) - f(11)$$
$$\hat{f}(\chi_{10}) = f(00) + f(01) - f(10) - f(11)$$
$$\hat{f}(\chi_{11}) = f(00) - f(01) - f(10) + f(11)$$

sommando attraverso le righe e

$$4f(00) = \hat{f}(\chi_{00}) + \hat{f}(\chi_{01}) + \hat{f}(\chi_{10}) + \hat{f}(\chi_{11})$$
$$4f(01) = \hat{f}(\chi_{00}) - \hat{f}(\chi_{01}) + \hat{f}(\chi_{10}) - \hat{f}(\chi_{11})$$
$$4f(10) = \hat{f}(\chi_{00}) + \hat{f}(\chi_{01}) - \hat{f}(\chi_{10}) - \hat{f}(\chi_{11})$$
$$4f(11) = \hat{f}(\chi_{00}) - \hat{f}(\chi_{01}) - \hat{f}(\chi_{10}) + \hat{f}(\chi_{11})$$

sommando attraverso le colonne. Questo è il secondo esempio di una trasformata di Walsh.

Esercizio 25.8. Calcolare le tabelle dei caratteri e dei caratteri coniugati di $\mathbb{Z}/2 \times \mathbb{Z}/3$ e di $\mathbb{Z}/2 \times \mathbb{Z}/4$. Il primo gruppo è isomorfo a $\mathbb{Z}/6$; confrontare le formule che si ottengono con il metodo della nota 25.6 con quelle ottenute con la nota 24.13.

LABORATORIO DI PROGRAMMAZIONE

Corso di laurea in matematica | Anno accademico 2004/05 | Numero 8

L'ordine di $a+b$ in un gruppo abeliano finito

Situazione 26.1. $(G,+)$ sia un gruppo abeliano finito. $a,b,\dots$ siano elementi di G, quando non diversamente indicato.

Le ipotesi cambiano dalla situazione 28.7.

Definizione 26.2. $\mathbb{P}$ sia l'insieme dei numeri primi.

Nota 26.3. Sia $n\in\mathbb{Z}$. Allora

$$n(a+b)=na+nb$$

È molto importante notare che questa relazione vale solo in un gruppo abeliano. In un gruppo non commutativo invece in genere $(ab)^n\neq a^nb^n$. Ciò si verifica già in S_3, il più piccolo gruppo non commutativo:

Siano $a=(12)$ e $b=(23)$. Allora $a^2=b^2=1$, mentre $ab=(123)$ ha ordine 3 e quindi $(ab)^2\neq 1$.

Definizione 26.4. $\mathfrak{o}_a := |\mathbb{Z}a|$ sia l'*ordine* dell'elemento a.

Osservazione 26.5. $\mathfrak{o}_a\mathfrak{o}_b(a+b)=0$

perciò $\mathfrak{o}_{a+b}|\mathfrak{o}_a\mathfrak{o}_b$. *Inoltre* $\mathfrak{o}_{a+b}a=-\mathfrak{o}_{a+b}b$.

Dimostrazione. G è abeliano e quindi

$$\begin{aligned}\mathfrak{o}_a\mathfrak{o}_b(a+b)&=\mathfrak{o}_a\mathfrak{o}_ba+\mathfrak{o}_a\mathfrak{o}_bb\\&=\mathfrak{o}_a\mathfrak{o}_ba=\mathfrak{o}_b\mathfrak{o}_aa=0\end{aligned}$$

Nota 26.6. Anche l'enunciato analogo a quello dell'osservazione 26.5 non vale per gruppi non commutativi, ad esempio non è più vero in S_3: Siano di nuovo $a=(12)$ e $b=(23)$ come nella nota 26.3. Allora $a^2=b^2=1$, ma $(ab)^4=(ab)\neq 1$.

Teorema 26.7. *Sia* $\operatorname{mcd}(\mathfrak{o}_a,\mathfrak{o}_b)=1$. *Allora*

$$\mathfrak{o}_{a+b}=\mathfrak{o}_a\mathfrak{o}_b$$

Dimostrazione. L'ipotesi implica che esistono $n,m\in\mathbb{Z}$ tali che $n\mathfrak{o}_a+m\mathfrak{o}_b=1$. Siccome $\mathfrak{o}_aa=0$, ciò implica $a=m\mathfrak{o}_ba$.

Per l'osservazone 26.5 $\mathfrak{o}_{a+b}a=-\mathfrak{o}_{a+b}b$ e quindi

$$\begin{aligned}\mathfrak{o}_{a+b}a&=\mathfrak{o}_{a+b}m\mathfrak{o}_ba=m\mathfrak{o}_b\mathfrak{o}_{a+b}a\\&=-m\mathfrak{o}_b\mathfrak{o}_{a+b}b=0\end{aligned}$$

Ciò implica $\mathfrak{o}_a|\mathfrak{o}_{a+b}$. Nello stesso modo si trova che $\mathfrak{o}_b|\mathfrak{o}_{a+b}$. Usando di nuovo che $\operatorname{mcd}(\mathfrak{o}_a,\mathfrak{o}_b)=1$ vediamo che $\mathfrak{o}_a\mathfrak{o}_b|\mathfrak{o}_{a+b}$.

Dalla prima parte dell'osservazione 26.5 sappiamo però che $\mathfrak{o}_{a+b}|\mathfrak{o}_a\mathfrak{o}_b$. Ciò implica $\mathfrak{o}_{a+b}=\mathfrak{o}_a\mathfrak{o}_b$.

Osservazione 26.8. *Siano* $n,m\in\mathbb{N}+2$ *e* $\operatorname{mcd}(n,m)=1$. *Allora* $\mathbb{Z}/n\times\mathbb{Z}/m$ *è ciclico e quindi isomorfo a* $\mathbb{Z}/nm$.

Dimostrazione. È chiaro che in $G:=\mathbb{Z}/n\times\mathbb{Z}/m$ si hanno $\mathfrak{o}_{(1,0)}=n$ e $\mathfrak{o}_{(0,1)}=m$. Per il teorema 26.7 $\mathfrak{o}_{(1,1)}=nm$.

Siccome $|G|=nm$, ciò implica che $(1,1)$ è un generatore di G.

Osservazione 26.9. *Siano* $n,m\in\mathbb{N}+2$ *e* $\operatorname{mcd}(n,m)>1$. *Allora* $\mathbb{Z}/n\times\mathbb{Z}/m$ *non è ciclico.*

Dimostrazione. Sia $d:=\operatorname{mcd}(n,m)$. Per ipotesi $\frac{nm}{d}<nm$. Sia (a,b) un elemento qualsiasi di $G:=\mathbb{Z}/n\times\mathbb{Z}/m$. Allora $\frac{n}{d}$ e $\frac{m}{d}$ sono interi e

$$\begin{aligned}\frac{nm}{d}(a,b)&=(\frac{nm}{d}a,\frac{nm}{d}b)\\&=(\frac{m}{d}na,\frac{n}{d}mb)=(0,0)\end{aligned}$$

Ciò implica che ogni elemento di G ha ordine $<nm$, per cui G non può essere ciclico.

Osservazione 26.10. *H e K siano gruppi e $\varphi:H\longrightarrow K$ un omomorfismo suriettivo. H sia ciclico. Allora anche K è ciclico.*

Dimostrazione. h sia un generatore di H e x un elemento qualsiasi di K. Allora esiste un $a\in H$ tale che $x=\varphi(a)$. Per ipotesi $a=h^n$ per qualche $n\in\mathbb{N}$. Quindi

$$x=\varphi(a)=\varphi(h^n)=(\varphi(h))^n$$

Ciò mostra che $\varphi(h)$ è un generatore di K.

Osservazione 26.11. *G_1 e G_2 siano gruppi e $G_1\times G_2$ ciclico. Allora anche G_1 e G_2 sono ciclici.*

Dimostrazione. Applichiamo l'osservazione 26.10 alle proiezioni $G_1\times G_2\longrightarrow G_i$.

Proposizione 26.12. *Siano* $n_1,\dots,n_s\in\mathbb{N}+2$. *Allora* $\mathbb{Z}/n_1\times\dots\times\mathbb{Z}/n_s$ *è ciclico se e solo se gli* n_j *sono relativamente primi a due a due, cioè se* $\operatorname{mcd}(n_i,n_j)=1$ *per ogni* i,j *con* $i\neq j$. *In tal caso*

$$\mathbb{Z}/n_1\times\dots\times\mathbb{Z}/n_s\cong\mathbb{Z}/n_1\cdots n_s$$

Dimostrazione. (1) Sia $\operatorname{mcd}(n_i,n_j)=1$ per ogni $i\neq j$. Per l'osservazione 26.8 allora $\mathbb{Z}/n_1\times\mathbb{Z}/n_2\cong\mathbb{Z}/n_1n_2$. È chiaro che $\operatorname{mcd}(n_1n_2,n_j)=1$ per ogni $j\geq 3$ e quindi possiamo ripetere il ragionamento.

(2) Sia $\operatorname{mcd}(n_i,n_j)>1$ per qualche i,j con $i\neq j$. Per l'osservazione 26.9 allora $\mathbb{Z}/n_i\times\mathbb{Z}/n_j$ non è ciclico e per l'osservazione 26.11 anche $\mathbb{Z}/n_1\times\dots\times\mathbb{Z}/n_s$ non può essere ciclico.

Lemma 26.13. *Sia* $n\in\mathbb{N}$ *tale che* $n|\mathfrak{o}_a$. *Allora* $\mathfrak{o}_{na}=\frac{\mathfrak{o}_a}{n}$.

Dimostrazione. $\mathfrak{o}_{na}na=0$, perciò esiste $s\in\mathbb{N}$ tale che $\mathfrak{o}_{na}n=s\mathfrak{o}_a$.

D'altra parte $\frac{\mathfrak{o}_a}{n}$ è un numero naturale perché $n|\mathfrak{o}_a$, e $\frac{\mathfrak{o}_a}{n}na=0$; ciò implica che esiste $t\in\mathbb{N}$ con $\frac{\mathfrak{o}_a}{n}=t\mathfrak{o}_{na}$. Abbiamo così $\mathfrak{o}_{na}n=snt\mathfrak{o}_{na}$. Siccome $\mathfrak{o}_{na},n,s,t$ sono numeri naturali $\neq 0$ ciò implica $t=1$ e quindi l'enunciato.

Nota 26.14. Non è difficile dimostrare che per un elemento g di ordine finito di un gruppo non necessariamente commutativo e per $n\in\mathbb{N}$ si ha sempre $\mathfrak{o}_{g^n}=\frac{\mathfrak{o}_g}{\operatorname{mcd}(\mathfrak{o}_g,n)}$.

Il lemma 26.13 è un caso speciale di questa formula spesso utilizzata.

In questo numero

Un lemma misterioso sul mcm

Definizione 26.15. Siano $n\in\mathbb{N}+1$ e p un numero primo. n_p sia allora la massima potenza di p che divide n.

Lemma 26.16. *Siano* $n,m\in\mathbb{N}+1$. *Allora esistono* $\alpha,\beta\in\mathbb{N}$ *tali che* $\operatorname{mcm}(n,m)=\alpha\beta$, $\operatorname{mcd}(\alpha,\beta)=1$, $\alpha|n$ *e* $\beta|m$.

Dimostrazione. (1) Se $n=1$, possiamo scegliere $\alpha=1$ e $\beta=m$. Se $m=1$, possiamo scegliere $\alpha=n$ e $\beta=1$.

(2) Assumiamo quindi che $n,m\geq 2$. Siano

$$\begin{aligned}\mathbb{P}_1&:=\{p\in\mathbb{P}\mid n_p>m_p\}\\\mathbb{P}_2&:=\{p\in\mathbb{P}\mid m_p>n_p\}\\\mathbb{P}_0&:=\{p\in\mathbb{P}\mid n_p=m_p\geq 1\}\\\mathcal{P}_j&:=\{n_p\mid p\in\mathbb{P}_j\}\text{ per }j=0,1,2\\n'&:=\prod_{p\in\mathbb{P}_1}n_p,\quad m':=\prod_{p\in\mathbb{P}_2}m_p\\r&:=\prod_{p\in\mathbb{P}_0}n_p\end{aligned}$$

Allora $\operatorname{mcm}(n,m)=n'm'r$. Poniamo

$$\alpha:=n',\beta:=m'r$$

Esempio 26.17. Siano

$$\begin{aligned}n&=2^3\cdot3^2\cdot5\cdot7\cdot11^2\cdot13\cdot17\\m&=2\cdot3^2\cdot5^2\cdot7^4\cdot11^2\cdot17^4\end{aligned}$$

Nella notazione del lemma 26.16 abbiamo

$$\begin{aligned}\mathcal{P}_1&=\{2^3,13\},\quad\mathcal{P}_2=\{5^2,7^4,17^4\}\\\mathcal{P}_0&=\{3^2,11^2\}\\\alpha&=2^3\cdot13\\\beta&=3^2\cdot5^2\cdot7^4\cdot11^2\cdot17^4\\m'&=5^2\cdot7^4\cdot17^4,\quad r=3^2\cdot11^2\end{aligned}$$

$+$					$+$	
2^3	3^2	5	7	11^2	13	17
2	3^2	5^2	7^4	11^2		17^4
	$=$	$+$	$+$	$=$		$+$

Come si usa lo schema?

Esistenza di sommandi ciclici

Lemma 27.1. *Per ogni coppia di elementi $a, b \in G$ esiste $c \in G$ tale che*

$$\mathfrak{o}_c = \mathrm{mcm}(\mathfrak{o}_a, \mathfrak{o}_b)$$

Dimostrazione. Per il lemma 26.16 esistono $\alpha, \beta \in \mathbb{N}$ tali che $\mathrm{mcm}(\mathfrak{o}_a, \mathfrak{o}_b) = \alpha\beta$, $\mathrm{mcd}(\alpha, \beta) = 1$, $\alpha | \mathfrak{o}_a$ e $\beta | \mathfrak{o}_b$. Siano

$$j := \frac{\mathfrak{o}_a}{\alpha}, \quad k := \frac{\mathfrak{o}_b}{\beta}$$

Per il lemma 26.13 $\mathfrak{o}_{ja} = \alpha$ e $\mathfrak{o}_{kb} = \beta$. Essendo $\mathrm{mcd}(\alpha, \beta) = 1$, dal teorema 26.7 segue che $\mathfrak{o}_{ja+kb} = \alpha\beta$.

Esempio 27.2. Siano n, m, α, β come nell'esempio 26.17 ed $\mathfrak{o}_a = n$, $\mathfrak{o}_b = m$.
Allora $\mathfrak{o}_{3^2 \cdot 5 \cdot 7 \cdot 11^2 \cdot 17a+2b} = \mathrm{mcm}(\mathfrak{o}_a, \mathfrak{o}_b)$.

Definizione 27.3. Per $n \in \mathbb{N}$ siano

$$G[n] := \{a \in G \mid na = 0\}$$
$$nG := \{na \mid a \in G\}$$

Proposizione 27.4. *Sia v un elemento di ordine massimo di G. Allora:*

(1) $\mathfrak{o}_a | \mathfrak{o}_v$ *per ogni* $a \in G$.
(2) $\mathfrak{o}_v a = 0$ *per ogni* $a \in G$.
(3) $G = G[\mathfrak{o}_v]$.

Dimostrazione. I tre enunciati sono equivalenti, perciò è sufficiente dimostrare il primo. Sia $a \in G$.
Per il lemma 27.1 esiste $c \in G$ tale che $\mathfrak{o}_c = \mathrm{mcm}(\mathfrak{o}_a, \mathfrak{o}_v)$.
La massimalità di $\mathfrak{o}_v$ implica $\mathfrak{o}_c = \mathfrak{o}_v$, per cui $\mathfrak{o}_v = \mathrm{mcm}(\mathfrak{o}_a, \mathfrak{o}_v)$. Ciò può accadere solo se $\mathfrak{o}_a | \mathfrak{o}_v$.

Definizione 27.5. Un sottogruppo H di G si chiama un *fattore diretto* (nel caso nostro di un gruppo abeliano anche un *sommando diretto*) di G, se esiste un sottogruppo K di G tale che $H + K = G$ e $H \cap K = 0$.
In tal caso K si chiama un *complemento* di H in G. Naturalmente allora anche K è un fattore diretto di G ed H è un complemento di K in G.

Nota 27.6. Vogliamo adesso dimostrare che ogni gruppo abeliano finito G è somma diretta di gruppi ciclici. È sufficiente dimostrare che G possiede dei sommandi ciclici, perché allora possiamo applicare lo stesso ragionamento al complemento di quel sommando e procedere così fino a quando quel complemento non è il gruppo 0. La proposizione 27.7 indica come trovare un sommando ciclico: possiamo prendere il sottogruppo generato da un elemento di ordine massimale. La dimostrazione usa da un lato i risultati sulle relazioni aritmetiche per gli ordini degli elementi di G che abbiamo dedotto in queste ultime due pagine, in particolare la proposizione 27.4 (che a sua volta discende soprattutto dal lemma 27.1), e dall'altro lato il teorema di estensione di caratteri e il fatto che $e^{\frac{2\pi i}{n}}$ è un generatore del gruppo delle n-esime radici dell'unità.

Proposizione 27.7. *Sia v un elemento di ordine massimo di G. Allora $\mathbb{Z}v$ è un fattore diretto di G.*

Dimostrazione. La dimostrazione è un bellissimo esempio della potenza della teoria dei caratteri.
Siano $n := \mathfrak{o}_v$ ed $\varepsilon := \varepsilon_{\mathbb{Z}v} = e^{\frac{2\pi i}{n}}$. Per il lemma 22.7 esiste $\varphi \in \widehat{\mathbb{Z}v}$ tale che $\varphi(v) = \varepsilon$.
Per il teorema 23.7 esiste $\chi \in \hat{G}$ con $(\chi = \varphi, \text{in } \mathbb{Z}v)$. In particolare $\chi(v) = \varepsilon$.
Sia $K := \ker \chi$. Allora K è un sottogruppo di G. Dimostriamo che K è un complemento di $\mathbb{Z}v$ in G.

(1) Sia $a \in G$. Per la proposizione 27.4 $na = 0$, perciò $\chi(a)$ è una n-esima radice dell'unità. Ma ogni n-esima radice dell'unità è una potenza di ε, quindi esiste $k \in \mathbb{N}$ con $0 \le k < n$ tale che $\chi(a) = \varepsilon^k$. Possiamo scrivere a nella forma

$$a = kv + (a - kv)$$

Però

$$\chi(a - kv) = \chi(a)\chi(-kv) = \varepsilon^k \varepsilon^{-k} = 1$$

e ciò mostra che $a - kv \in K$, per cui $a \in \mathbb{Z}v + K$.

(2) Rimane da dimostrare che $\mathbb{Z}v \cap K = 0$.
Sia $a \in \mathbb{Z}v \cap K$. Allora $a = kv$ per qualche $k \in \mathbb{N}$ con $0 \le k < n$, e inoltre

$$1 = \chi(a) = \chi(kv) = \varepsilon^k$$

Ma allora $k = 0$! Abbiamo infatti supposto che $0 \le k < n$ e l'ordine di ε è n. Ciò implica $a = kv = 0$.

Il teorema di struttura

Osservazione 27.8. *Per sottogruppi $H_1, \ldots, H_s$ di G l'applicazione*

$$H_1 \times \ldots \times H_s \longrightarrow G$$
$$(a_1, \ldots, a_s) \longmapsto a_1 + \ldots + a_s$$

è un omomorfismo di gruppi.

Dimostrazione. G è abeliano!

Proposizione 27.9. *Per sottogruppi $H_1, \ldots, H_s$ di G sono equivalenti:*

(1) L'applicazione $H_1 \times \ldots \times H_s \xrightarrow{m} G$ dell'osservazione 27.8 è un isomorfismo di gruppi.
(2) Si ha $H_1 + \ldots + H_s = G$ e inoltre $h_1 + \ldots + h_s = 0$ con $h_i \in H_i$ per ogni i implica $h_1 = \ldots = h_s = 0$.
(3) Ogni elemento a di G possiede un'unica rappresentazione nella forma

$$a = h_1 + \ldots + h_s$$

con $h_i \in H_i$ per ogni i.
(4) $H_1 + \ldots + H_s = G$ *e* $H_i \cap \sum_{j \ne i} H_j = 0$ *per ogni i.*

In tal caso diciamo che G è la somma diretta dei sottogruppi $H_1, \ldots, H_s$ e scriviamo

$$G = H_1 \oplus \ldots \oplus H_s$$

Per la condizione (1) allora

$$G \cong H_1 \times \ldots \times H_s$$

Dimostrazione. (1) $\iff$ (2): Chiaro. Infatti la prima condizione in (2) significa che m è suriettiva, la seconda che $\ker m = 0$.

(2) $\implies$ (3): Sia $a \in G$. Per ipotesi $a = h_1 + \ldots + h_s$ con $h_i \in H_i$ per ogni i. Sia anche $a = h_1' + \ldots + h_s'$ con $h_i' \in H_i$ per ogni i. Allora

$$(h_1 - h_1') + \ldots + (h_s - h_s') = 0$$

e dall'ipotesi (2) segue $h_i' = h_i$ per ogni i.

(3) $\implies$ (2): Chiaro.

(2) $\implies$ (4): L'ipotesi implica

$$H_1 + \ldots + H_s = G$$

Sia adesso ad esempio

$$h_1 \in H_1 \cap (H_2 + \ldots + H_s)$$

Allora $h_1 = h_2 + \ldots + h_s$ con $h_i \in H_i$ per ogni $i \ge 2$ (ma anche $h_1 \in H_1$), cosicché $h_1 - h_2 - \ldots - h_s = 0$. Per ipotesi si ha $h_i = 0$ per ogni i.

(4) $\implies$ (2): Sia $h_1 + \ldots + h_s = 0$ con $h_i \in H_i$ per ogni i. Allora $h_1 \in H_2 + \ldots + H_s$ e per ipotesi $h_1 = 0$. Nello stesso modo si dimostra $h_i = 0$ per ogni i.

Osservazione 27.10. *H sia un fattore diretto di G e K un complemento di H in G.*
Allora $G = H \oplus K$.

Osservazione 27.11. *H e K siano sottogruppi di G e $G = H \oplus K$. M ed N siano sottogruppi di K e $K = M \oplus N$.*
Allora $G = H \oplus M \oplus N$.

Dimostrazione. (1) In primo luogo

$$H + M + N = H + K = G.$$

(2) Siano $a \in H, b \in M, c \in N$ e $a + b + c = 0$. Siccome $b + c \in K$, abbiamo $a = b + c = 0$. Però anche $K = M \oplus N$ e quindi $b = c = 0$.

Teorema 27.12. *Esistono elementi $v_1, \ldots, v_s \in G$ tali che:*

(1) $G = \mathbb{Z}v_1 \oplus \ldots \oplus \mathbb{Z}v_s$.
(2) $\mathfrak{o}_{v_{i+1}} | \mathfrak{o}_{v_i}$ *per* $i = 1, \ldots, s-1$.

Dimostrazione. v_1 sia un elemento di ordine massimale di G ed $n_1 := \mathfrak{o}_{v_1}$. Per la proposizione 27.7 esiste un sottogruppo H_1 di G tale che $G = \mathbb{Z}v \oplus H_1$. Per la proposizione 27.4 $n_1 G = 0$ e quindi anche $n_1 H_1 = 0$.
Lo stesso ragionamento può essere applicato ad H_1. Scegliamo un elemento v_2 di ordine massimale in H_1 e poniamo $n_2 := \mathfrak{o}_{v_2}$. Allora $n_1 v_2 = 0$ e quindi $n_2 | n_1$.
Di nuovo per la proposizione 27.7 H_1 possiede un sottogruppo H_2 con $H_1 = \mathbb{Z}v_2 \oplus H_2$. Per la proposizione 27.4 $n_2 H_2 = 0$.
Per l'osservazione 27.11 a questo punto $G = \mathbb{Z}v_1 \oplus \mathbb{Z}v_2 \oplus H_2$. In questo modo andiamo avanti. Siccome

$$|G| > |H_1| > |H_2| > \ldots$$

a un certo punto avremo $H_i = \mathbb{Z}v_i$.

Corollario 27.13. *Sia $G \ne 0$. Allora esistono numeri naturali $n_1, \ldots, n_s \ge 2$ con $n_{i+1} | n_i$ per ogni $i = 1, \ldots, s-1$ tali che*

$$G \cong \mathbb{Z}/n_1 \times \ldots \times \mathbb{Z}/n_s$$

Osservazione 27.14. *Sia $n = p_1^{\alpha_1} \cdots p_k^{\alpha_k}$ con $p_1, \ldots, p_k$ primi distinti ed $\alpha_1, \ldots, \alpha_k \ge 1$. Allora*

$$\mathbb{Z}/n \cong \mathbb{Z}/p_1^{\alpha_1} \times \ldots \times \mathbb{Z}/p_k^{\alpha_k}$$

Dimostrazione. Proposizione 26.12.

Nota 27.15. I numeri $n_1, \ldots, n_s$ nel corollario 27.13 sono univocamente determinati. G è isomorfo a un prodotto diretto di gruppi ciclici della forma $\mathbb{Z}/p^m$ con p primo. Questi fattori sono univocamente determinati. Così a meno di isomorfia esistono esattamente 6 gruppi abeliani di ordine $600 = 2^3 \cdot 3 \cdot 5^2$:

$$\mathbb{Z}/2 \times \mathbb{Z}/2 \times \mathbb{Z}/2 \times \mathbb{Z}/3 \times \mathbb{Z}/5 \times \mathbb{Z}/5$$
$$\mathbb{Z}/2 \times \mathbb{Z}/2 \times \mathbb{Z}/2 \times \mathbb{Z}/3 \times \mathbb{Z}/25$$
$$\mathbb{Z}/2 \times \mathbb{Z}/4 \times \mathbb{Z}/3 \times \mathbb{Z}/5 \times \mathbb{Z}/5$$
$$\mathbb{Z}/2 \times \mathbb{Z}/4 \times \mathbb{Z}/3 \times \mathbb{Z}/25$$
$$\mathbb{Z}/8 \times \mathbb{Z}/3 \times \mathbb{Z}/5 \times \mathbb{Z}/5$$
$$\mathbb{Z}/8 \times \mathbb{Z}/3 \times \mathbb{Z}/25$$

Per la prop. 26.12 solo l'ultimo è ciclico.

Dimostrazione. Risultati finora ottenuti e corso di Algebra.

G è isomorfo a $\hat{G}$

Teorema 28.1. $G \cong \hat{G}$.

Dimostrazione. Corollario 27.13 (o teorema 27.12) e nota 25.5. Ricordiamo ancora una volta che questo isomorfismo in genere è canonico soltanto se $G = 2^n$ per qualche $n \in \mathbb{N}$.

Caratteri reali

Osservazione 28.2. *Applicando la definizione 27.3 a $\hat{G}$ per $n \in \mathbb{N}$ abbiamo*

$$\begin{aligned}\hat{G}[n] &= \{\chi \in \hat{G} \mid \chi^n = \chi_0\} \\ &= \{\chi \in \hat{G} \mid \chi^n(a) = 1 \textit{ per ogni } a \in G\}\end{aligned}$$

Siccome per $\hat{G}$ usiamo la notazione moltiplicativa, scriviamo (non è un prodotto cartesiano!)

$$\hat{G}^n = \{\chi^n \mid \chi \in \hat{G}\}$$

Definizione 28.3. Gli elementi di $\hat{G}[2]$ si chiamano *caratteri quadratici* di G, gli elementi di $\hat{G}[3]$ si chiamano *caratteri cubici*.

Per ogni $\chi \in \hat{G}[2]$ si ha

$$\chi(a) \in \{1, -1\}$$

per ogni $a \in G$ e per ogni $\chi \in \hat{G}[3]$ si ha

$$\chi(a) \in \{1, e^{\frac{2\pi i}{3}}, e^{\frac{4\pi i}{3}}\}$$

per ogni $a \in G$.

Caratteri quadratici e cubici sono molto importanti nella teoria dei numeri e nella teoria dei campi finiti.

Osservazione 28.4. *I caratteri quadratici di G sono esattamente i caratteri reali di G, cioè quei caratteri per cui $\chi(a) \in \mathbb{R}$ per ogni $a \in G$.*

Dimostrazione. Per $\chi \in \hat{G}$ sono equivalenti:

(1) χ è reale.
(2) $\overline{\chi} = \chi$.
(3) $\chi^{-1} = \chi$.
(4) $\chi^2 = \chi_0$.

Proposizione 28.5. *Sono equivalenti:*

(1) $2G = 0$.
(2) $G[2] = G$.
(3) Ogni carattere di G è reale.
(4) $\hat{G}^2 = 1$.
(5) $G \cong 2^n$ *per qualche* $n \in \mathbb{N}$.

Dimostrazione. (1) $\iff$ (2): Chiaro.

(3) $\iff$ (4): Osservazione 28.4.

(1) $\implies$ (4): Siano $a \in G$ e $\chi \in \hat{G}$. Allora $\chi^2(a) = \chi(2a) = \chi(0) = 1$.

(4) $\implies$ (1): Sia $a \in G$. Assumiamo che $2a \neq 0$. Per la proposizione 23.8 esiste $\chi \in \hat{G}$ tale che $\chi(2a) \neq 1$. Ma per ipotesi

$$\chi(2a) = \chi^2(a) = \chi_0(a) = 1$$

una contraddizione.

(1) $\implies$ (5): Per il corollario 27.13

$$G \cong \mathbb{Z}/n_1 \times \ldots \times \mathbb{Z}/n_s$$

con $n_1, \ldots, n_s \geq 2$ (se $G \neq 0 = 2^0$). Per ipotesi $G[2] = 0$. Ciò è possibile solo se $n_j = 2$ per ogni j.

(5) $\implies$ (1): Chiaro.

Nota 28.6. La proposizione 28.5 è un esempio di una *dualità* tra G e $\hat{G}$ che si può elaborare molto più sistematicamente.

La trasformata di Walsh

Situazione 28.7. $n \in \mathbb{N}$ ed $f : 2^n \longrightarrow \mathbb{C}$ un'applicazione. Tralasciamo talvolta il caso banale $n = 0$ e assumiamo $n > 0$.

Nota 28.8. Applicheremo adesso la teoria generale al gruppo 2^n. La trasformata di Fourier in questo caso si chiama anche *trasformata di Walsh* (o *trasformata di Hadamard*), come avevamo già osservato a pagina 25. Per la proposizione 28.5 i caratteri di 2^n sono tutti reali, per cui le formule della definizione 24.6 diventano:

$$\hat{f}(\chi) = \sum_{x \in 2^n} f(x)\chi(x)$$

per ogni $\chi \in \widehat{2^n}$, ed

$$f(x) = \frac{1}{2^n} \sum_{\chi \in \widehat{2^n}} \hat{f}(\chi)\chi(x)$$

per ogni $x \in 2^n$.

Nota 28.9. Siccome $2^n = \mathbb{Z}/2 \times \ldots \times \mathbb{Z}/2$, per la proposizione 25.4 e la nota 25.6 abbiamo, come negli esempi 25.5 e 25.7, concretamente

$$\widehat{2^n} = \{\chi_{u_1 \ldots u_n} \mid (u_1, \ldots, u_n) \in 2^n\}$$

con

$$\chi_{u_1 \ldots u_n}(x_1, \ldots, x_n) = (-1)^{u_1 x_1 + \ldots + u_n x_n}$$

per $u_1, \ldots, u_n, x_1, \ldots, x_n \in \{0, 1\}$.

Se scriviamo $u = (u_1, \ldots, u_n)$ ed $x = (x_1, \ldots, x_n)$ e quindi anche $\chi_u(x)$, abbiamo

$$\chi_u(x) = (-1)^{\|u,x\|}$$

dove $\|\|$ è il comune prodotto scalare in $\mathbb{R}^n$, in cui comunque appaiono solo vettori i cui componenti sono elementi di $\{0, 1\}$, per cui $\|u, x\| \in \mathbb{N}$. Inoltre

$$(-1)^{\|u,x\|} = (-1)^{\|u,x\|_{\mathrm{mod}\,2}}$$

e quindi possiamo sostituire $\|\|$ con il prodotto scalare $\|\|_{\mathrm{mod}\,2}$ in 2^n. Abbiamo perciò

$$\chi_u(x) = (-1)^{\|u,x\|_{\mathrm{mod}\,2}}$$

D'altra parte, se consideriamo u ed x come insiemi,

$$\|u, x\| = |u \cap x|$$

e

$$\begin{aligned}\|u, x\|_{\mathrm{mod}\,2} &= |u \cap x|_{\mathrm{mod}\,2} \\ &= \begin{cases} 0 & \text{se } |u \cap x| \text{ è pari} \\ 1 & \text{se } |u \cap x| \text{ è dispari} \end{cases}\end{aligned}$$

cosicché

$$\begin{aligned}\chi_u(x) = (-1)^{|u \cap x|} &= (-1)^{|u \cap x|_{\mathrm{mod}\,2}} \\ &= \begin{cases} 1 & \text{se } |u \cap x| \text{ è pari} \\ -1 & \text{se } |u \cap x| \text{ è dispari} \end{cases}\end{aligned}$$

Ad esempio per

$$u = 01101110$$
$$x = 00101101$$

si ha $|u \cap x| = 3$ e quindi $\chi_u(x) = -1$.

Le formule di trasformazione diventano

$$\hat{f}(\chi_u) = \sum_{x \in 2^n} f(x)(-1)^{|u \cap x|}$$

$$f(x) = \frac{1}{2^n} \sum_{u \in 2^n} \hat{f}(\chi_u)(-1)^{|u \cap x|}$$

Siano adesso

$$u^{\mathrm{pari}} := \{x \in 2^n \mid |u \cap x| \text{ pari}\}$$
$$u^{\mathrm{disp}} := \{x \in 2^n \mid |u \cap x| \text{ dispari}\}$$

Allora

$$\hat{f}(\chi_u) = \sum_{x \in u^{\mathrm{pari}}} f(x) - \sum_{x \in u^{\mathrm{disp}}} f(x)$$

Questa formula mostra particolarmente bene che la trasformata di Fourier può essere considerata come *un'operazione di sommazione*; cfr. esempio 25.7. Come tale viene interpretata in statistica nell'analisi dei disegni di esperimenti.

Nota 28.10. Come nell'esempio 22.10 si ha un isomorfismo canonico

$$\begin{aligned} 2^n &\longrightarrow \widehat{2^n} \\ u &\longmapsto \chi_u \end{aligned}$$

Possiamo perciò elencare gli elementi di 2^n e $\widehat{2^n}$ nello stesso modo, identificando $x \in 2^n$ con il numero naturale j di cui x è la rappresentazione binaria (completata ad n cifre come a pagina 16), cioè quel $j \in \mathbb{N}$ per cui $(x)_2 = j$, e nello stesso modo $\chi_u \in \widehat{2^n}$ con il numero naturale i per cui $(u)_2 = i$. In questo modo otteniamo un ordine naturale per 2^n e $\widehat{2^n}$ che ci permette di scrivere la tavola dei caratteri di 2^n come matrice con 2^n righe e 2^n colonne, esattamente come nell'esempio 25.7. Denotiamo questa matrice con

$$W := W_n$$

e la chiamiamo la n-esima *matrice di Walsh*. Si ha quindi $W \in \{1, -1\}_{2^n}^{2^n}$ e dalle relazioni di ortogonalità segue che

$$WW^t = W^t W = 2^n \delta$$

Da ciò deriva $W^{-1} = \frac{1}{2^n} W$.

Nel calcolo combinatorio una matrice $A \in \{1, -1\}_s^s$ con $AA^t = A^t A = s\delta$ si chiama una *matrice di Hadamard*; la matrice di Walsh è quindi un esempio di matrice di Hadamard.

Per poter distinguere tra il gruppo 2^n e il numero 2^n poniamo $G = 2^n$ (come gruppo). Possiamo allora identificare una funzione $f \in \mathbb{C}^G$ con il vettore corrispondente $v \in \mathbb{C}^{|G|}$, se gli elementi di G vengono elencati nell'ordine naturale appena descritto, e la trasformata $\hat{f} \in \mathbb{C}^{\hat{G}}$ con un vettore $\hat{v}$ in $\mathbb{C}^{|G|}$ (perché sappiamo che $|\hat{G}| = |G|$), elencando anche gli elementi di $\hat{G}$ nell'ordine naturale. La formula per la trasformata di Walsh diventa allora

$$\hat{v} = Wv$$

come si vede anche dall'esempio 25.7.

La formula di inversione diventa

$$v = \frac{1}{2^n} W\hat{v}$$

La trasformata di Walsh veloce

Nota 29.1. Esiste un modo molto semplice per ottenere W_{n+1} da W_n. In primo luogo $W_0 = 1$ e, come si vede dall'esempio 22.10,

$$W_1 = \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} = \begin{pmatrix} W_0 & W_0 \\ W_0 & -W_0 \end{pmatrix}$$

Questa relazione vale per ogni n:

$$W_{n+1} = \begin{pmatrix} W_n & W_n \\ W_n & -W_n \end{pmatrix}$$

per ogni $n \in \mathbb{N}$.

Dimostrazione. Induzione su n.

$n = 0, 1$: Già visto.

$n \longrightarrow n+1$: Osserviamo che ogni elemento di 2^{n+1} può essere scritto nella forma $(0, x)$ oppure $(1, x)$ con $x \in 2^n$. Un elemento $(0, x)$ appartiene alla prima metà degli elementi di 2^{n+1} nell'ordine naturale, un elemento $(1, x)$ alla seconda metà. Ricordando dalla nota 28.10 il modo in cui abbiamo elencato i coefficienti della matrice di Walsh e dalla nota 28.9 la formula per i caratteri, W_{n+1} può essere scritta nel modo seguente:

$$\begin{pmatrix} * & * & * & * & * \\ * & \chi_{(0,u)}(0,x) & * & \chi_{(0,u)}(1,x) & * \\ * & * & * & * & * \\ * & \chi_{(1,u)}(0,x) & * & \chi_{(1,u)}(1,x) & * \end{pmatrix}$$

Abbiamo però, con $w := (-1)^{\|u,x\|}$,

$$\chi_{(0,u)}(0,x) = (-1)^{\|u,x\|} = w$$
$$\chi_{(0,u)}(1,x) = (-1)^{\|u,x\|} = w$$
$$\chi_{(1,u)}(0,x) = (-1)^{\|u,x\|} = w$$
$$\chi_{(1,u)}(1,x) = (-1)^{1+\|u,x\|} = -w$$

Ciò mostra che W_{n+1} ha veramente la forma indicata nell'enunciato.

Possiamo quindi calcolare la matrice di Walsh direttamente dalla relazione della nota 29.1 senza usare le formule delle nota 28.8 o 28.9. Nella traduzione in R usiamo le funzioni `rbind` e `cbind` che uniscono le righe rispettivamente le colonne di due matrici:

$$\mathrm{rbind}(A, B) = \begin{pmatrix} A \\ B \end{pmatrix}$$

$$\mathrm{cbind}(A, B) = \begin{pmatrix} A & B \end{pmatrix}$$

Abbiamo così la semplicissima funzione

```
Mw.matrice = function (n)
{if (n==0) 1
else {W=Mw.matrice(n-1)
rbind(cbind(W,W),cbind(W,-W))}}
```

Con `Mw.matrice(3)` otteniamo adesso la matrice W_3, cioè la tavola dei caratteri di 2^3,

```
 1  1  1  1  1  1  1  1
 1 -1  1 -1  1 -1  1 -1
 1  1 -1 -1  1  1 -1 -1
 1 -1 -1  1  1 -1 -1  1
 1  1  1  1 -1 -1 -1 -1
 1 -1  1 -1 -1  1 -1  1
 1  1 -1 -1 -1 -1  1  1
 1 -1 -1  1 -1  1  1 -1
```

e con `Mw.matrice(4)` la tavola dei caratteri di 2^4:

```
 1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1
 1 -1  1 -1  1 -1  1 -1  1 -1  1 -1  1 -1  1 -1
 1  1 -1 -1  1  1 -1 -1  1  1 -1 -1  1  1 -1 -1
 1 -1 -1  1  1 -1 -1  1  1 -1 -1  1  1 -1 -1  1
 1  1  1  1 -1 -1 -1 -1  1  1  1  1 -1 -1 -1 -1
 1 -1  1 -1 -1  1 -1  1  1 -1  1 -1 -1  1 -1  1
 1  1 -1 -1 -1 -1  1  1  1  1 -1 -1 -1 -1  1  1
 1 -1 -1  1 -1  1  1 -1 -1  1  1 -1  1 -1 -1  1
 1  1  1  1  1  1  1  1 -1 -1 -1 -1 -1 -1 -1 -1
 1 -1  1 -1  1 -1  1 -1 -1  1 -1  1 -1  1 -1  1
 1  1 -1 -1  1  1 -1 -1 -1 -1  1  1 -1 -1  1  1
 1 -1 -1  1 -1  1  1 -1 -1  1  1 -1  1 -1 -1  1
 1  1  1  1 -1 -1 -1 -1 -1 -1 -1 -1  1  1  1  1
 1 -1  1 -1 -1  1 -1  1 -1  1 -1  1  1 -1  1 -1
 1  1 -1 -1 -1 -1  1  1 -1 -1  1  1  1  1 -1 -1
 1 -1 -1  1 -1  1  1 -1 -1  1  1 -1  1 -1 -1  1
```

Nota 29.2. Nelle applicazioni pratiche dobbiamo calcolare, nella notazione vettoriale della nota 28.10, la trasformata di Walsh

$$\hat{v} = Wv$$

Se calcoliamo $\hat{v}$ come prodotto Wv, dobbiamo però, per ciascuno dei 2^n componenti di v eseguire 2^n addizioni e quindi $2^n \cdot 2^n = 2^{2n}$ addizioni. È quindi importante che la semplice forma di W_n ci permette di calcolare $\hat{v}$ in modo molto più rapido senza addirittura eseguire un'operazione matriciale.

Siccome la lunghezza di v è una potenza di 2, possiamo, per $n > 0$, sempre scrivere

$$v = \begin{pmatrix} v_1 \\ v_1 \end{pmatrix}$$

con due vettori $v_1, v_2 \in 2^{n-1}$. Allora

$$\hat{v} = W_n v = \begin{pmatrix} W_{n-1} & W_{n-1} \\ W_{n-1} & -W_{n-1} \end{pmatrix} \begin{pmatrix} v_1 \\ v_2 \end{pmatrix}$$

Se poniamo $A := W_{n-1}$, abbiamo quindi

$$\hat{v} = \begin{pmatrix} Av_1 + Av_2 \\ Av_1 - Av_2 \end{pmatrix}$$

Possiamo perciò porre

$$p_1 := Av_1, \quad p_2 := Av_2$$

avendo così

$$\hat{v} = \begin{pmatrix} p_1 + p_2 \\ p_1 - p_2 \end{pmatrix} \quad (*)$$

p_1 e p_2 si calcolano però di nuovo mediante una matrice di Walsh (perché $A = W_{n-1}$) e quindi possiamo ripetere il procedimento fino a quando arriviamo a vettori di lunghezza 1. Si noti che la matrice di Walsh non viene mai calcolato in questo algoritmo che usa soltano la relazione (*) e, come è facile vedere, richiede soltanto $O(n2^n)$ operazioni. $n2^n$ è però molto più piccolo di $(2^n)^2$. Questo algoritmo si chiama la *trasformata veloce di Walsh*. In R lo realizziamo mediante la seguente funzione:

```
Mw = function (v)
{N=length(v)
if (N==1) v
else {M=N/2
v1=v[1:M]; v2=v[(M+1):N]
p1=Mw(v1); p2=Mw(v2)
c(p1+p2,p1-p2)}}
```

Questa funzione può essere applicata direttamente a ogni vettore la cui lunghezza è una potenza di 2, ad esempio

```
v=c(2,5,8,3)
print(Mw(v))
# output: 18 2 -4 -8
```

Verifichiamo che il risultato coincide con quello che si ottiene col metodo dell'esempio 25.7, ricordandoci che dobbiamo sommare per righe:

$$\hat{v} = \begin{pmatrix} 2+5+8+3 \\ 2-5+8-3 \\ 2+5-8-3 \\ 2-5-8+3 \end{pmatrix} = \begin{pmatrix} 18 \\ 2 \\ -4 \\ -8 \end{pmatrix}$$

Esercizio: Provare

```
v=c(2,1,3,4,2,5,8,7)
print(Mw(v))
# output: 32 -2 -12 -2 -12 2 4 6
```

Controllare il risultato a mano utilizzando la tavola dei caratteri di 2^3 nella prima colonna di questa pagina.

La funzione `Mw` ha come argomento un vettore. Se la funzione che vogliamo trasformare è data invece veramente come funzione su 2^n, ad esempio

```
f=function (x)
x[1]+x[1]*x[3]-x[2]
```

dobbiamo prima calcolare il vettore che ad essa corrisponde. A questo scopo serve la funzione

```
Mw.vettore = function (f,n)
{X=Mfb.cubo(n)
sapply(X,f)}
```

Esempio:

```
f=function (x)
x[1]+x[1]*x[3]-x[2]

v=Mw.vettore(f,3)

print(v)
# output: 0 0 -1 -1 1 2 0 1

w=Mw(v)
print(w)
# output: 2 -2 4 0 -6 2 0 0
```

Nota 29.3. La trasformata di Walsh ha molte applicazioni nell'elaborazione dei segnali e delle immagini, nella teoria dei codici, nel disegno di esperimenti statistici, nella teoria delle funzioni booleane. Per quanto riguarda queste ultime, la trasformata di Walsh viene impiegata sia per studiare la *sensitività* di una funzione booleana, cioè la misura in cui la funzione cambia se in un argomento $x \in 2^n$ si modifica uno dei coefficienti, che nelle *applicazioni crittografiche*. Per poter utilizzare una funzione booleana α in crittografia bisogna che la funzione sia il più irregolare possibile e la trasformata di Walsh della sua polarizzazione fornisce uno strumento per verificare e stimare questa irregolarità.

Definizione 29.4. Sia α una funzione booleana. Allora la *polarizzazione* p_α di α è definita come

$$p_\alpha(x) := (-1)^{\alpha(x)} = 1 - 2\alpha(x)$$

È abbastanza naturale che si studi la trasformata $\widehat{p_\alpha}$ piuttosto che $\hat{\alpha}$, perché p_α assume valori in $\{1, -1\}$ e quindi può essere confrontata direttamente con i caratteri di 2^n che assumono anch'essi valori in $\{1, -1\}$.

La FFT

Nota 30.1. L'algoritmo per la trasformazione veloce di Walsh presentato nella nota 29.2 è particolarmente semplice. Se guardiamo la tabella dei caratteri di un altro gruppo abeliano finito, ad esempio $\mathbb{Z}/n$, una decomposizione simile non è più così immediata. Nonostante ciò una tale decomposizione è possibile anche nel caso generale, benché sia più nascosta e più complicata, e ciò porta all'algoritmo per il calcolo veloce della trasformata di Fourier (in inglese *fast Fourier transform, FFT*) che risale addirittura a Gauß ed è stato riscoperto varie volte fino a diventare universalmente noto nel 1965 in seguito a una pubblicazione di Jim Cooley (un giovane programmatore) e John Tukey (un famoso matematico e statistico) come *algoritmo di Cooley-Tukey*. La trasformata veloce di Fourier ha avuto un impatto immenso nella tecnologia della comunicazione e dell'informatica perché permette l'esecuzione in tempo reale delle operazioni di base dell'elaborazione dei segnali e delle immagini.

La trasformata di Fourier di $\mathbb{Z}/n$ in R è fornita dalla funzione `fft` che, mediante l'opzione `inverse=T`, permette anche di calcolare la trasformata inversa. Il primo argomento è nella versione più semplice un vettore v di numeri complessi; viene calcolata la trasformata in $\mathbb{Z}/n$ se la lunghezza di v è uguale ad n. Proviamo per $n = 4$ con il vettore

$$v = \begin{pmatrix} 1 \\ 2 \\ 3 \\ 5 \end{pmatrix}$$

Secondo quanto visto nell'esempio 25.2 dovremmo avere (ricordandoci che dobbiamo utilizzare la tabella dei caratteri coniugati e sommare attraverso le righe):

$$\hat{v} = \begin{pmatrix} 1+2+3+5 \\ 1-2i-3+5i \\ 1-2+3-5 \\ 1+2i-3-5i \end{pmatrix} = \begin{pmatrix} 11 \\ -2+3i \\ -3 \\ -2-3i \end{pmatrix}$$

Infatti ciò è quanto otteniamo con la funzione `fft` di R:

```
v=c(1,2,3,5)
w=fft(v)
print(w)
# output semplificato:
# 11+0i -2+3i -3+0i -2-3i
```

La funzione `fft` può essere anche usata per la trasformata di Fourier in $G := \mathbb{Z}/n_1 \times \ldots \times \mathbb{Z}/n_s$. Per $s = 2$ possiamo rappresentare $v \in \mathbb{C}^{|G|}$ in forma di matrice come in

```
v=M.matrice(c(2,5,1,4,1,3,2,1),nrow=2)
w=fft(v); print(w)
# output semplificato:
# 18 2
# -4 -8
```

Il risultato coincide (a parte la rappresentazione matriciale) con quello ottenuto nella nota 29.2. Infatti in questo caso $n_1 = n_2 = 2$ e quindi $G = 2^2$. La funzione `fft` fornisce quindi anche la trasformata di Walsh.

Esercizio 30.2. Provare

```
v=M.matrice(c(2,5,1,4,1,3,2,1),righe=2)
w=fft(v); print(w)
# output semplificato:
# 19 -3i -7 3i
# 5 2+i -5 2-i
```

Calcolare $\hat{v}$ con l'aiuto della tabella dei caratteri calcolata nell'esercizio 25.8 e confrontare i risultati.

Per $s > 2$ si può usare la funzione `array` per creare elementi di $\mathbb{C}^G$ con

$G = \mathbb{Z}/n_1 \times \ldots \times \mathbb{Z}/n_s$.

Esercizio 30.3. Sia $G = 2^3$. Con

```
v=array(c(4,1,3,2,5,8,1,1),dim=c(2,2,2))
```

si crea un elemento $v \in \mathbb{C}^G$. Calcolare $\hat{v}$ con l'aiuto della tabella alla fine della prima colonna a pagina 29. Poi in R provare `fft` e confrontare i risultati.

Calcolo della convoluzione

Nota 30.4. In molte applicazioni si utilizza la convoluzione. La formula della definizione 24.11 richiede però un calcolo impegnativo o e quindi si usa il teorema 24.12 che permette di ottenere la convoluzione $f * g$ con il seguente algoritmo:

(1) Calcolo di $\hat{f}$ e $\hat{g}$.
(2) Calcolo di $\hat{f} \cdot \hat{g}$. Questa operazione è semplicissima.
(3) $\hat{f} \cdot \hat{g}$ coincide però con $\widehat{f * g}$. Mediante l'inversa della trasformazione di Fourier otteniamo adesso $f * g$.

Usando la trasformata veloce di Fourier (e l'algoritmo veloce quasi identico per l'inversa) le operazioni (1) e (3) sono così veloci che nella somma questo modo di calcolare la convoluzione risulta notevolmente più rapido del calcolo diretto secondo la definizione 24.11.

Con la funzione `convolve` di R si ottiene la convoluzione. Verifichiamo il teorema 24.12 in un esempio:

```
u=c(1,2,4,2)
v=c(3,1,2,5)

p=convolve(u,v)
P=fft(p)

U=fft(u)
V=fft(v)

print(P)
print(U*V)
# output semplificato:
# 99 -3+12i -1 -3-12i
# 99 -3+12i -1 -3-12i
```

La FFT in $\mathbb{Z}/2^d$

Per un gruppo ciclico la formula

$$\hat{f}(\chi_k) = \sum_{n=0}^{|G|-1} f(x_n)\varepsilon^{-nk}$$

della nota 24.13 mostra che la trasformata di Fourier si riconduce al calcolo dei valori $P(\varepsilon^{-k})$ del polinomio

$$P := \sum_{n=0}^{|G|-1} f(x_n)x^n$$

Quando $N := |G|$ è una potenza di 2, il calcolo di questi valori può essere eseguito con l'algoritmo che adesso presentiamo.

Siano N una potenza di 2 e

$$P = a_0 + a_1 x + \ldots + a_{N-1}x^{N-1}$$

con gli indici elencati in ordine crescente.

Per $N = 1$ abbiamo $P = a_0$, per $N = 2$ invece $P = a_0 + a_1 x$. Sia $N > 2$. In questo caso possiamo decomporre P nella forma

$$P = P_1(x^2) + xP_2(x^2)$$

$$P_1 = a_0 + a_2 x + \ldots + a_{N-2}x^{\frac{N}{2}-1}$$

$$P_2 = a_1 + a_3 x + \ldots + a_{N-1}x^{\frac{N}{2}-1}$$

Se, per un numero α reale o complesso, denotiamo il valore di $P(\alpha)$ con $V(\alpha, a_0, a_1, \ldots, a_{N-1})$, abbiamo quindi

$$\begin{aligned} &V(\alpha, a_0, a_1, \ldots, a_{N-1}) \\ &\quad = V(\alpha^2, a_0, a_2, \cdots, a_{N-2}) \\ &\qquad + \alpha \cdot V(\alpha^2, a_1, a_3, \ldots, a_{N-1}) \end{aligned}$$

La realizzazione in R non sarebbe difficile:

```
M.valic = function (v, x)
{N=length(v)
if (N==1) v
else if (N==2) v[1]+x*v[2]
else {x2=x^2
indici=1:N
pari=indici[indici%%2==1]
# Perche' in R si inizia da 1.
dispari=pari+1
a=v[pari]; b=v[dispari]
M.valic(a,x2)+x*M.valic(b,x2)}}
```

risulta però piuttosto lenta, almeno ai fini della trasformata di Fourier. La riprogrammiamo quindi in C, per coefficienti reali (`Valicreale`) e complessi (`Valic`), usando il tipo `nc` introdotto nel corso di Programmazione. Il suffisso `ic` ricorda che gli indici sono crescenti.

```
double Valicreale  (Double A, int N, double x)
{if (N==1) return A[0];
if (N==2) return A[0]+x*A[1];
int M=N/2; int k; double x2;
double a1[M],a2[M]; Double X,X1,X2;
for (k=0,X=A,X1=a1,X2=a2;k<M;k++)
{*(X1++)=*(X++); *(X2++)=*(X++);}
x2=x*x;
return Valicreale (a1,M,x2)+
  x*Valicreale(a2,M,x2);}

nc Valic (nc *A, int N, nc z)
{if (N==1) return A[0];
if (N==2) return Ncadd(A[0],Ncmolt(z,A[1]));
int M=N/2; int k; nc z2;
nc a1[M],a2[M]; nc *X,*X1,*X2;
for (k=0,X=A,X1=a1,X2=a2;k<M;k++)
{*(X1++)=*(X++); *(X2++)=*(X++);}
z2=Ncmolt(z,z);
return Ncadd(Valic(a1,M,z2),
  Ncmolt(z,Valic(a2,M,z2)));}
```

Ancora più veloce, ma più difficile da programmare, è la variante iterativa.