Si può rappresentare qualsiasi informazione se costruita da elementi elencabili in un insieme e a cui è possibile associare un numero (l’indice nell’elenco). Le parole sono rappresentate come costruite da lettere; lettere che vengono identificate dalla posizione che occupano nell’insieme, per esempio, ASCII.

Ci sono più modi di rappresentare i numeri con segno: rappresentazione modulo e segno (non canonica), la rappresentazione in complemento a 1 (sempre non canonica in realtà), la rappresentazione in complemento a 2.

Si usa principalmente la rappresentazione in complemento a 2 perché è semplice ed efficiente per far funzionare l’algoritmo di somma dei numeri con segno. Infatti poiché col complemento a 2 l’1 finale indica un -128 si può evitare di considerarlo durante l’operazione di somma (ed eventualmente riaggiungerlo alla fine). Per calcolare un cambio di segno in complemento a 2 basta ribaltare le cifre binarie e poi sommare la costante 00000001.

Un trucchetto per semplificare l’algoritmo di cambiamento di segno è scansionando le cifre da destra a sinistra e, tutte le volte che si incontra un valore 0 si copia il valore 0; quando si trova il primo 1 si copia 1 e tutte le cifre dopo il primo 1 vanno invertite.

L’unico problema che può sorgere dalla somma di due numeri con rappresentazione in complemento a 2 è un caso di overflow (ma questo è un problema comune di tutti i possibili algoritmi di somma per tutte le rappresentazioni possibili).

Un altro (quarto) tipo di rappresentazione per i numeri con segno è la rappresentazione Eccesso 2^m. m indica il numero di cifre binarie che si sta considerando -1.

Ad esempio, in una rappresentazione su 5 bit (con 1 bit di segno e 4 di mantissa) 2^m sarà il valore 2^4 -> 16. Questa costante la sommiamo ai valori che vogliamo rappresentare in modo da modificare il valore. Quindi se si vuole rappresentare v si rappresenta invece un valore v’ = v + 2^m = v +16. Nel caso v = 11 -> v’ = 27. Si codifica quindi la rappresentazione binaria del numero 27 (11011) anziché del numero 11. Per codificare il valore -2 si codifica invece la rappresentazione binaria di 14 (01110). Complemento a 2 ed Eccesso 2^m sono molto simili, si inverte però il valore del bit di segno.

Quando si ha necessità di rappresentare i numeri razionali viene adottata la Rappresentazione in virgola fissa. Si prende una rappresentazione binaria di un numero definito (per esempio 16) bit. Si potrebbe quindi prendere il valore codificato e lo si moltiplica per un certo valore costante compreso tra 0 e 1. Per esempio, si potrebbe assumere come costante 0.5: questo permetterebbe di rappresentare 0.5, 1.0, 1.5, 2.0, 2.5 ecc. Per fare ciò basta far partire la codifica dal valore 2^-1 anziché 2^0 (questo però riduce il numero massimo rappresentabile, poiché l’ultima casella codifica per 2^14 anziché 2^15). Questo metodo si dice “a virgola fissa” proprio perché si decide a priori il punto FISSO in cui impostare il 2^0 (rendendo tutte le caselle prima codificanti per potenze negative di 2, come 2^-1, 2^-2, 2^-3 ecc.) dopo il quale iniziano le cifre decimali. Se si scegliesse il valore 00…0010000 dopo aver impostato alla 4 cifra la virgola, questo verrebbe letto dal computer come 32 ma in fase di codifica e decodifica varrebbe come 2.0.

Questo metodo di rappresentazione può tornare utile per rappresentare delle probabilità: infatti basta piazzare la virgola subito dopo l’ultima casella/cifra disponibile (quella più a sinistra) e avere così molte cifre disponibili per la codifica di numeri anche “molto” piccoli. Certi numeri frazionari sono così rappresentabili esattamente, tuttavia altri non saranno rappresentati esattamente e verranno approssimati (come 1/3, poiché è irrazionale). La precisione con cui possiamo scrivere i nostri valori (da approssimare) dipende quindi dalla disponibilità di bit: tanti più bit sono dedicati alla rappresentazione della parte decimale, migliore sarà l’approssimazione.

Anche in questa rappresentazione si possono avere problemi di overflow. In genere il problema principale di queste rappresentazioni è però decidere la posizione migliore in cui piazzare la virgola: in alcuni casi è semplice, come in quello della probabilità, in altri no, poiché bisognerebbe prevedere il massimo numero che si andrà a trattare e piazzare la virgola in modo da evitare l’overflow.

Un’alternativa che risolva questo problema è la Rappresentazione in Virgola Mobile. Questa rappresentazione si differenzia perché, mentre in virgola fissa il coefficiente moltiplicativo è fisso/costante, in virgola mobile questo moltiplicatore diventa una variabile.

L’idea è quella di scrivere un valore v come il prodotto tra una mantissa (modulo) m e 2^e (Nota: non è il numero di Nepero). Poiché **e** varia e non è predeterminato esso viene rappresentato assieme subito dopo il valore m. Il valore di bit da dedicare ad e è però prefissato. Per esempio, in una rappresentazione su 16 bit si possono dedicare 4 bit ad all’esponente (e) e 12 bit alla mantissa (m). Se, sempre per esempio, si posiziona la virgola in m al penultimo (11°) bit il valore al suo interno potrebbe valere 1.1001… e sarebbe poi moltiplicato per 2^e.

Per e in genere si usa una rappresentazione dei numeri con segno, per poter identificare anche potenze negative di 2. Nel caso di rappresentazione a eccesso a 8 si potrebbe scrivere sia m \* 2^3 che m \* 2^-4. La dimensione finale del numero rappresentato (piccolo o grande) dipende quindi principalmente dal valore dell’esponente e per cui 2 viene elevato prima di essere moltiplicato alla mantissa.

Il risultato che si ottiene è quello di poter trattare sia numeri grandi che numeri piccoli con un elevata precisione a livello di cifre decimali utilizzando sempre lo stesso tipo di rappresentazione.

Prima di poter operare sui numeri a virgola mobile bisogna decodificarli.

Per evitare di rendere non canonica la rappresentazione in floating point (virgola mobile) si utilizza la condizione di Normalizzazione: essa consiste nell’eliminare tutte le possibilità che non sono quella che sta utilizzando al meglio i bit per la rappresentazione della mantissa. In pratica si dice che la rappresentazione è **normalizzata** se la prima cifra della mantissa è diversa da zero (perché così facendo sta, appunto, utilizzando al meglio i bit).

Ci sono dei numeri che non si possono rappresentare in forma normalizzata (per esempio 0, ma in generale i numeri molto piccoli): bisogna quindi determinare qual è il più piccolo numero rappresentabile in forma normalizzata.

Nella realtà viene utilizzata la Rappresentazione Standard, ideata da IEEE 802 (?).