Indice

Capitolo 1

Game Theory, boolean functions, etc...

1.1 Formulario di Teoria dei Giochi

A, B, X, R, N, etc. for SETS a, b, x, r, n, etc. for ELEMENTS of above sets i, j, n, x, y, z, etc for INDICES of SETS or ELEMENTS (), for function application and tuples Indices are use to iterate over a set or to name the object to which belongs.

Doesn't exist a way to simulate hash map in mathematics.

Dobbiamo inventare un operatore o una struttura dati in grado di rappresentare in modo funzionale ai calcoli un hash map ovvero un oggetto che possiede, si porta dientro con se a sua volta un insieme.

Vi è quasi una certa ridondanza in questo fatto.

Ma in realtà si tratta solo di comprendere il significato dell'applicazione/funzione/mappa che assegna ad ogni elemento di un insieme, un altro insieme.

```
%% ATTENZIONE
l'indice n a volte seve per dare un nome e un numero, altre volte solo per dare un numero, per es
senza che faccia specificatamente ad n che rappresenta il numero dei giocatori. n indica due cose
intenda il numero di giocatori

%% VALUE
Si definisce value un qualunque numero reale.

    %% VOGLIAMO QUANTIFICARE (VALORIZZARE) - valori di R
    LET R be REAL SET
    LET v in R

    %% VOGLIAMO CONTARE (COUNT)            - valori di N
    LET n in N
    LET A be SET
    LET |A| := n


    %% VOGLIAMO RANDOMIZZARE. In questo caso occorre il value di un intero insieme.
    LET D be SET
    LET n INDICE of N
    LET D = {d_1, ..., d_n}
    LET [0,1] \in R
    LET d_1, ..., d_n \in [0,1]
    LET d_1 + ... + d_n = 1
    Ora che abbiamo costruito D (o DELTA) ovvero l'insieme randomizzatore lo possiamo utilizzare
```

```
LET A, B be SETS
LET B = {b}
LET b in R
LET {B} be an anonymous SET. In this case, {B} is a set with two elements: B (which is a set) and
LET a elements of A
LET value be a FUNCTION
LET value A \to {B} -- ABSTRACTION
LET value defined as a --> B in words: each elements of A is mapped to the same set B.
    This fact is useful for successive steps, counting how many of...
LET value(a) --> SUM of elements of B
```

Now we are able to define randomness

```
LET A, B_1, ... B_n , {B_1, ..., B_n} be SETS   %% Here n is just an INDICES
LET a elements of A
LET value be a FUNCTION
LET value A \to {B_1, ..., B_n} -- ABSTRACTION
LET value defined as
```

So we are conducted to the definition of value of an element of a set. Whenever I get an element of a set I can ask for its value.

In questo modo non basta più dire che A e B sono insiemi ma occorre anche specificare la funzione value che restituisce il valore di un suo elemento, cioè una funzione che dato in input un elemento dell'insieme restituisce un valore.

Per convenzione e per semplificare i calcoli assumiamo che le funzioni value restituiscano sempre un insieme e che il value è dato dalla somma dei valori presenti nell'insieme restituito.

Let's define *Pareto*

Prima però proviamo a fare il passaggio successivo, prima abbiamo imparata a calcolare il valore del payoff di un giocatore ma questa cosa può anche essere sbagliata dal punto di vista della teoria dei giochi cooperativi. Adesso vogliamo introdurre la nozione di payoff per coalizione che come si può immaginare è la somma dei payoff dei singoli giocatori.

Altra cosa importante sarà quello di attivare il meccanismo della partizione (partizionamento) e della conseguente enumerazione di sotto-insiemi che godono di determinate proprietà.

Ed infine occorrerà passare alle boolean function.