SEMINAR 1

DECIDABILITATE

In cele ce urmeaza vom prezenta cateva notiuni importante din teoria calculabilitatii.

Prin *problema* intelegem o multime de intrebari referitoare la proprietatile dinamice sau structurale ale unor entitati (procese sau obiecte) care accepta raspunsuri precise, finite ca reprezentare si care pot fi riguros demonstrate.

O *problema de decizie* se caracterizeaza prin faptul ca multimea rezultatelor posibile este 1 sau 0 (true sau false) in functie daca problema are solutie sau nu pentru o anumita data de intrare valida. Un exemplu de problema de decizie este problema colorarii unui graf (*kColorare*): fiind date un graf G si un numar natural k, poate fi graful G colorat folosind k culori, astfel incat oricare doua noduri adiacente sa fie colorate diferit?

O *problema de optimizare* selecteaza solutiile optime din multimea de solutii posibile ale problemei. Optimizarea (minimizare sau maximizare) se face in raport cu o functie de cost. In general, problemele de optimizare pot fi reformulate sau rezolvate folosind problemele de decizie asociate. Problema colorarii unui graf ca problema de optimizare (*MinkColorare*) se poate enunta astfel: fiind dat un graf G, sa se determine numarul minim de culori necesar colorarii grafului G, astfel incat oricare doua noduri adiacente sa fie colorate diferit. Problema de optimizare *MinkColorare* poate fi solutionata apeland o rezolvare a problemei de decizie *kColorare*.

```
MinkColorare(G){
      for(i=0; i≤nr_noduri(G); i++)
            if(kColorare(G)) return i;
      return -1;
}
```

O *procedura efectiva* este o multime finita de operatii, cu reprezentare finita, a caror executie se termina in timp finit si implica o cantitate finita de date pentru a obtine un rezultat, mereu aceeasi pentru date identice. Specificarea unei proceduri efective, folosind o notatie conventionala, conform particularitatilor computationale ale unui tip de masina de calcul reprezinta un *algoritm.*

O *functie* f : I -> O este o mulţime de perechi $\{(x, f(x)) \mid x \in I \wedge f(x) \in O\}$ care asociaza fiecarei valori din domeniul functiei un singur rezultat, dar care nu arata in nici un fel modul in care rezultatul poate fi efectiv calculat. In schimb, un algoritm indica precis modul si ordinea in care datele de intrare sunt transformate pentru a obtine un rezultat. Daca consideram algoritmii care primesc ca date de intrare un numar si au ca rezultat un alt numar, putem privi relatia intre cele doua numere printr-o functie. Prin urmare, *putem construi algoritmi care sa* *calculeze functii*. Dar, nu orice functie este calculabila. Cu alte cuvinte, nu orice functie accepta algoritmi care sa o calculeze. Astfel, putem da urmatoarea definitie:

O *functie* f : I -> O este *efectiv calculabila (recursiva)* daca exista cel putin o procedura efectiva (un algoritm) capabila sa calculeze valoarea functiei pentru orice valoare din domeniul de definitiei al acesteia.

Un exemplu de functie efectiv calculabila este functia $test_{Prime}$: ℕ -> ℕ care testeaza apartenenta unui numar natural la multimea numerelor prime si poate fi calculata prin urmatorul algoritm:

```
testePrime(x){
     if(x ≤ 1) return 0;
     for(d=2; d ≤ isqr(x); d++)
      if(x % d = 0) return 0;
     return 1;
}
```

O *functie* f : I -> O este *calculabila in sens Turing* daca exista cel putin un program P ∈ $P_{1,1}$ astfel incat:

$$P(x) = \begin{cases} f(x), pentru\ x\ \in dom(f) \\ \perp, pentru\ x\ \notin dom(f) \end{cases}$$

unde ⊥ semnifica ca programul nu se termina (ruleaza la infinit) iar dom(f) este domeniul de definitie al functiei f.

Avand definite conceptele de functie efectiv calculabila si functie calculabila in sens Turing putem enunta *Teza Church-Turing: Orice functie este efectiv calculabila daca si numai daca este calculabila in sens Turing.* Teza Church-Turing sustine, ca orice proces de calcul poate fi redus la unul specific unei masini Turing. Cu alte cuvinte, alte modele computationale (precum calculul Lambda, algoritmii normali, etc.) sunt echivalente computational masinii Turing, adica, orice calcul efectuat intr-un model poate fi efectuat in toate celelalte.

O *functie* f : B -> C este *partiala peste multimea A* daca B ⊂ A, respectiv *totala peste multimea A* daca B = A.

O *multime* A⊆ ℕ este *recursiva* (prescurtat R) daca exista o functie recursiva si totala f: ℕ -> ℕ, adica exista un program P ∈ $P_{1,1}$ astfel incat:

$$P(x) = \begin{cases} 1, pentru\ x\ \in A \\ 0, pentru\ x\ \notin A \end{cases}$$

O *multime* A ⊆ ℕ este *recursiv-enumerabila* (prescurtat RE) daca:

a) exista o functie recursiva si partiala f: A -> ℕ , adica exista un program P ∈ $P_{1,1}$ astfel incat:

$$P(x) = \begin{cases} 1, pentru\ x\ \in A \\ \perp, pentru\ x\ \notin A \end{cases}$$

sau

b) exista un program Q ∈ $P_{0,1}$ (numit *program generator* al multimii A) care la fiecare apel calculeaza si intoarce ca rezultat un element din A.

Fie $T : \mathbb{N} \to \{0, 1\}$ un predicat si $M_T = \{n \in \mathbb{N} \mid T(n)=1\}$ multimea de adevar a lui T. Spunem ca:

a) T este decidabil daca M_T este recursiva;
b) T este semidecidabil daca M_T este recursiv enumerabila;
c) T este nedecidabil daca M_T nu este recursiva.

In concluzie, putem spune ca perechile de adjective recursiv-decidabil, respectiv recursiv enumerabil–semidecidabil sunt echivalente, dar se prefera folosirea pentru multimi a adjectivelor recursiv/recursiv enumerabil iar pentru probleme a adjectivelor decidabil/ semidecidabil.

Bibliografie

Giumale, Cristian A., Introducere in Analiza Algoritmilor, Editura Polirom, 2004.

Probleme de Rezolvat

1) Aratati ca:
   a) orice multime recursiva este recursiv enumerabila;
   b) pentru o submultime $A \subseteq \mathbb{N}$ recursiv enumerabila si $B= \mathbb{N} \setminus A$ recursiv enumerabila, rezulta ca A este recursiva.
2) Daca A si B sunt doua multimi recursive, demonstrati ca $A \cup B$ si $A \cap B$ sunt recursive.
3) Daca A si B sunt doua multimi recursiv enumerabile, demonstrati ca $A \cup B$ si $A \cap B$ sunt recursiv enumerabile.
4) Fie multimea $A = \{x \mid \exists y, z, t,\ y \neq z \neq t, \text{ a.i. } (x,y), (x,z), (x,t) \in B \text{ si } y = z^2\}$, unde $A \subset \mathbb{N}$, $B \subset \mathbb{N} \times \mathbb{N}$ este recursiv enumerabila. Demonstrati ca A este recursiv enumerabila.
5) Fie A o multime recursiv enumerabila. Ce puteti spune despre multimile B, C, D si E?
   $B = \{x \mid \exists y, (x , y) \in A\}$
   $C = \{x \mid \forall y, (x , y) \in A\}$
   $D = \{x \mid \exists y, (x , y) \notin A\}$
   $E = \{x \mid \forall y, (x , y) \notin A\}$
6) Fie *A* o multime recursiva, *B* o multime recursiv enumerabila dar care nu este recursiva. Stabiliti valoarea de adevar a afirmatiei urmatoare (intotdeauna adevarat, uneori adevarat si in ce situatii sau intotdeauna fals):
   $C = \{x \mid x \in A \setminus B\}$ este recursiv enumerabila.
7) Fie predicatele P1, P2, Q1, Q2 : $\mathbb{N} \to \{0,1\}$. Stiind ca predicatele P1 v P2, Q1 v Q2 sunt semidecidabile si P2 = ~Q2. Ce puteti spune despre predicatul P1 v Q1?
8) Fie predicatele decidabile A, B : $\mathbb{N} \to \{0,1\}$. Ce puteti spune despre predicatul A->B?