



# ALGORITMI - CALCULABILITATE

Horia Georgescu

În cadrul acestui articol vom aborda proprietățile de închidere ale funcțiilor și codificarea programelor cu ajutorul câtorva demonstrații și vom trata problema opririi programelor cu ajutorul tezei lui Church. Vom insista asupra calculabilității funcțiilor.

## Proprietăți de închidere

Fie  $g_1, \dots, g_k$  funcții de aritate  $n$ , iar  $h$  o funcție de aritate  $k$ . Putem atunci defini funcția  $f$  de aritate  $n$  prin:

$$f(x_1, \dots, x_n) = h(g_1(x_1, \dots, x_n), \dots, g_k(x_1, \dots, x_n)).$$

## Propoziția 1

Dacă  $g_1, \dots, g_k, h$  sunt (parțial) calculabile, atunci și funcția  $f$  este (parțial) calculabilă.

Funcția  $f$  este calculată de următorul program:

```
z1 ← g1(x1, ..., xn)
...
zk ← gk(x1, ..., xn)
y ← h(z1, ..., zk)
```

Propoziția de mai sus arată că atât clasa funcțiilor parțial calculabile, cât și clasa funcțiilor calculabile, sunt închise la operația de compunere.

Fie  $n$  un număr natural și funcțiile  $f: N^m \rightarrow N, g: N^{m+2} \rightarrow N$ . Vom defini funcția  $h: N^{m+1} \rightarrow N$  prin:

$$(I) \begin{cases} h(x_1, \dots, x_n, 0) = f(x_1, \dots, x_n) \\ h(x_1, \dots, x_n, t+1) = g(t, h(x_1, \dots, x_n, t), x_1, \dots, x_n), t \geq 0 \end{cases}$$

## Propoziția 2

Dacă  $f$  și  $g$  sunt calculabile, atunci  $h$  este calculabilă. Într-adevăr, funcția  $h$  este calculată de programul:

```
y ← f(x1, ..., xn)
[A] if xn+1 = 0 goto E
y ← g(t, y, x1, ..., xn)
t ← t + 1
```

```
xn+1 ← xn+1
goto A
```

Această propoziție demonstrează închiderea clasei funcțiilor calculabile la recursii de tipul (1).

Pentru cazurile  $n = 0$  și  $n = 1$ , recursia (1) se scrie:

$$(I') \begin{cases} h(0) = k \in N \\ h(t+1) = g(t, h(t)) \end{cases}, \text{ respectiv}$$
$$(I'') \begin{cases} h(x, 0) = f(x) \\ h(x, t+1) = g(t, h(x, t), x) \end{cases}$$

unde în (1') s-a ținut cont că funcția de aritate 0 se identifică cu o constantă.

Proprietățile de închidere luate în considerare mai sus ne permit să prezentăm și alte exemple de funcții calculabile; fiecare dintre ele permite introducerea unei macroinstrucțiuni, conform exemplului 7. Noile exemple ne vor fi utile în continuare și în plus ne vor mări încrederea în limbajul S.

## Exemplul 9

Funcția  $b(n) = n!$  este calculabilă.

$$\text{Într-adevăr: } \begin{cases} h(0) = 1 \\ h(t+1) = h(t) \cdot (t+1) = g(t, h(t)), t > 0 \end{cases}$$

unde  $g(x_1, x_2) = (x_1 + 1) \cdot x_2$ .

## Exemplul 10

Funcția  $b: N \times N \rightarrow N$  dată de  $b(x_1, x_2) = x_1^{x_2}$  este calculabilă.



Pentru demonstrație, plecăm de la relațiile:  $x^0 = 1$ ;  $x^{y+1} = x^y \cdot x$  (am luat deci prin convenție  $0^0 = 1$ ), obținând:

$$\begin{cases} h(x_1, 0) = 1 \\ h(x_1, t+1) = h(x_1, t) \cdot x_1 = g(t, h(x_1, t), x_1) \end{cases}$$

unde  $g(x_1, x_2, x_3) = x_2 \cdot x_3$ .

### Exemplul 11

Funcția  $f: N \times N \rightarrow N$  dată de:

$$f(x_1, x_2) = x_1 \dot{-} x_2 = \begin{cases} x_1 - x_2, & x_1 \geq x_2 \\ 0, & x_1 < x_2 \end{cases}$$

este calculabilă (demonstrația este propusă ca exercițiu).

### Exemplul 12

Funcția  $f: N \times N \rightarrow N$ ,  $f(x_1, x_2) = |x_1 - x_2|$ , este calculabilă, deoarece  $|x_1 - x_2| = (x_1 \dot{-} x_2) + (x_2 \dot{-} x_1)$ .

### Exemplul 13

Funcția  $\alpha: N \rightarrow N$ , dată de:

$$\alpha(x) = \begin{cases} 1, & x = 0 \\ 0, & x \neq 0 \end{cases}$$

este calculabilă, așa cum s-a arătat în exemplul 8.

### Exemplul 14

Predicatul  $f: N \times N \rightarrow \{0, 1\}$  (predicatul "="), dat de:

$$f(x_1, x_2) = \begin{cases} 1, & x_1 = x_2 \\ 0, & \text{altfel} \end{cases}$$

este calculabil, deoarece  $f(x_1, x_2) = \alpha(|x_1 - x_2|)$ .

### Exemplul 15

Predicatul  $f: N \times N \rightarrow \{0, 1\}$  (predicatul " $\leq$ "), dat de:

$$f(x_1, x_2) = \begin{cases} 1, & x_1 \leq x_2 \\ 0, & \text{altfel} \end{cases}$$

este calculabil, deoarece  $f(x_1, x_2) = \alpha(x_1 \dot{-} x_2)$ ; într-adevăr,  $x_1 \leq x_2 \Leftrightarrow x_1 \dot{-} x_2 = 0$ .

### Propoziția 3

Dacă  $P$  și  $Q$  sunt predicate calculabile, atunci  $\neg P$ ,  $P \vee Q$ ,  $P \wedge Q$  sunt predicate calculabile.

### Demonstrație

Este suficient să observăm că:

$$\neg P = \alpha \circ P; P \wedge Q = P \cdot Q; P \vee Q = \neg((\neg P) \wedge (\neg Q)).$$

### Propoziția 4

Dacă  $g, h$  sunt funcții  $n$ -are (parțial) calculabile, iar  $P$  un predicat  $n$ -ar calculabil, atunci funcția  $n$ -ară:

$$f(x_1, x_2, \dots, x_n) = \begin{cases} g(x_1, x_2, \dots, x_n), & P(x_1, x_2, \dots, x_n) = 1 \\ h(x_1, x_2, \dots, x_n), & P(x_1, x_2, \dots, x_n) = 0 \end{cases}$$

este (parțial) calculabilă (observăm analogia cu construcția **if-then-else**).

### Demonstrație

Într-adevăr,  $f = g \cdot P + h \cdot (\alpha \circ P)$ .

### Propoziția 5

Fie  $f$  o funcție calculabilă de aritate  $n + 1$ . Atunci funcțiile  $g$  și  $h$  definite de:

$$\begin{aligned} g(x_1, x_2, \dots, x_n, x_{n+1}) &= \sum_{t=0}^{x_{n+1}} f(x_1, x_2, \dots, x_n, t) \\ h(x_1, x_2, \dots, x_n, x_{n+1}) &= \prod_{t=0}^{x_{n+1}} f(x_1, x_2, \dots, x_n, t) \end{aligned}$$

sunt calculabile.

### Demonstrație

Este suficient să observăm că:

$$\begin{cases} g(x_1, x_2, \dots, x_n, 0) = f(x_1, x_2, \dots, x_n, 0) \\ g(x_1, x_2, \dots, x_n, t+1) = g(x_1, x_2, \dots, x_n, t) + f(x_1, x_2, \dots, x_n, t+1) \\ h(x_1, x_2, \dots, x_n, 0) = f(x_1, x_2, \dots, x_n, 0) \\ h(x_1, x_2, \dots, x_n, t+1) = h(x_1, x_2, \dots, x_n, t) \cdot f(x_1, x_2, \dots, x_n, t+1) \end{cases}$$

și să aplicăm propoziția 2.

### Propoziția 6

Fie  $P$  un predicat calculabil de aritate  $n + 1$ . Atunci sunt calculabile și următoarele predicate de aritate  $n + 1$  cu definiție evidentă:

$$\begin{aligned} (\forall t)_{\leq n+1} P(x_1, x_2, \dots, x_n, t); & (\exists t)_{\leq n+1} P(x_1, x_2, \dots, x_n, t); \\ (\forall t)_{< n+1} P(x_1, x_2, \dots, x_n, t); & (\exists t)_{< n+1} P(x_1, x_2, \dots, x_n, t). \end{aligned}$$

### Demonstrație

Demonstrația se bazează pe următoarele echivalențe:

$$\begin{aligned} (\forall t)_{\leq n+1} P(x_1, x_2, \dots, x_n, t) &\Leftrightarrow \prod_{t=0}^{x_{n+1}} P(x_1, x_2, \dots, x_n, t) = 1 \\ (\exists t)_{\leq n+1} P(x_1, x_2, \dots, x_n, t) &\Leftrightarrow \sum_{t=0}^{x_{n+1}} P(x_1, x_2, \dots, x_n, t) \neq 0 \\ (\forall t)_{< n+1} P(x_1, \dots, x_n, t) &\Leftrightarrow (\forall t)_{\leq n+1} [t = x_{n+1} \vee P(x_1, \dots, x_n, t)] \\ (\exists t)_{< n+1} P(x_1, \dots, x_n, t) &\Leftrightarrow (\exists t)_{\leq n+1} [t \neq x_{n+1} \wedge P(x_1, \dots, x_n, t)] \end{aligned}$$

### Exemplul 16

Predicatul binar (notat prin " $|$ ") următor:

$$P(x_1, x_2) = 1 \Leftrightarrow x_1 | x_2,$$

este calculabil.

### Demonstrație

Într-adevăr,  $x_1 | x_2 \Leftrightarrow (\exists z)_{< x_2} (x_2 = x_1 \cdot z)$ .

### Exemplul 17

Predicatul unar *Prim*, definit prin:

$$Prim(x) = 1 \Leftrightarrow (x > 1) \wedge (\forall t)_{< x} [(t = 1) \vee \neg (t | x)],$$

este calculabil.



### Propoziția 7

Fie  $P$  un predicat calculabil de aritate  $n + 1$ . Atunci următoarea funcție de aritate  $n + 1$  este calculabilă:

$$\min_{t \leq x_{n+1}} P(x_1, x_2, \dots, x_n, t)$$

cu precizarea că dacă nu există un astfel de  $t$ , valoarea obținută va fi 0.

### Demonstrație

Într-adevăr, funcția de mai sus este calculată de programul:

```
[A] if P(x1, ..., xn, t) goto B
    t ← t + 1
    if t ≤ xn+1 goto A
    y ← 0
    goto E
[B] y ← t
```

Rățiunea pentru care am ales prin lipsă valoarea 0 pentru minim va apărea în secțiunea următoare.

### Exemplul 18

Funcția binară  $f$  ("partea întreagă din raport") dată de:

$$f(x_1, x_2) = \lfloor x_1/x_2 \rfloor$$

este calculabilă.

### Demonstrație

Într-adevăr:

$$\lfloor x_1/x_2 \rfloor = \min_{t \leq x_1} [(t+1) \cdot x_2 > x_1].$$

Să observăm că am acceptat că  $\lfloor x/0 \rfloor = 0$ .

### Exemplul 19

Funcția binară  $R$  care calculează restul împărțirii a două numere naturale este calculabilă.

### Demonstrație

Într-adevăr,  $R(x_1, x_2) = x_1 \div (\lfloor x_1/x_2 \rfloor \cdot x_2)$ . Să observăm că am acceptat că  $R(x, 0) = x$ .

### Exemplul 20

Funcția unară definită prin:

$f(n) =$  al  $n$ -lea număr prim (dacă  $n > 0$ ) sau 0 (dacă  $n = 0$ ) este calculabilă.

### Demonstrație

Vom nota  $p_n = f(n)$ . Deci  $p_0 = 0, p_1 = 2, p_2 = 3, p_3 = 5, \dots$

Într-adevăr, pentru  $n > 0$  avem:

$$p_{n+1} = \min_{t \leq p_n+1} [Prim(t) \wedge (t > p_n)].$$

Această egalitate se bazează pe inegalitatea  $p_{n+1} \leq p_n! + 1$ , care este adevărată deoarece fie  $p_n! + 1$  este prim, fie are un divizor prim mai mic decât el care nu poate fi nici unul dintre  $p_1, \dots, p_n$ .

Definim acum succesiv funcțiile  $h$  și  $k$ , evident calculabile, date de:

$$h(y, z) = \min_{t \leq z} [Prim(t) \wedge (t > y)]$$

$$k(x) = h(x, x!+1)$$

Rezultă:

$$\begin{cases} p_0 = 0 \\ p_{n+1} = k(p_n), \forall n \geq 0 \end{cases}$$

și, conform propoziției 2, funcția  $f$  este calculabilă.

### Codificarea programelor

În această secțiune vom vedea cum pot fi codificate programele.

### Reprezentarea perechilor și n-uplelor

Pentru orice două numere naturale  $x, y \in N$ , definim  $\langle x, y \rangle = 2^x \cdot (2 \cdot y + 1) - 1$ .

Funcția  $f: N \times N \rightarrow N$  dată de  $f(x, y) = \langle x, y \rangle$  este bijectivă.

Într-adevăr pentru orice  $z$ , există și sunt unice numerele  $x, y$  cu  $\langle x, y \rangle = z$ :

- $x$  este cea mai mare putere a lui 2 care divide pe  $z + 1$ ;
- $y$  poate fi determinat de relația  $2 \cdot y + 1 = (z + 1)/2^x$  (membrul drept al acestei relații fiind impar).

Notând prin  $l$  și  $r$  funcțiile prin care se obțin  $x$  și  $y$  din  $z$  (adică  $\langle l(z), r(z) \rangle = z$ ), observăm că ele sunt calculabile deoarece:

$$\begin{cases} l(z) = \min_{x \leq z} [-(2^{x+1} | (z+1))] \\ r(z) = \lfloor ((z+1)/2^x - 1)/2 \rfloor. \end{cases}$$

Pentru reprezentarea  $n$ -uplului  $(a_1, \dots, a_n)$  definim numărul Gödel atașat astfel:

$$[a_1, \dots, a_n] = \prod_{i=1}^n p_i^{a_i},$$

unde  $p_1 = 2, p_2 = 3, p_3 = 5, \dots$  este șirul numerelor prime. De exemplu  $[2, 0, 1] = 2^2 \cdot 5^1 = 20 = [2, 0, 1, 0, \dots, 0]$ .

Pentru orice  $n \geq 1$ , funcția  $n$ -ară  $f$  dată de  $f(x_1, \dots, x_n) = [x_1, \dots, x_n]$  este evident calculabilă.

Din teorema fundamentală a aritmeticii deducem că:

- pentru orice  $x$  nenul, există  $n, a_1, \dots, a_n$  cu  $[a_1, \dots, a_n] = x$  (deci funcția  $f$  este surjectivă);
- din  $[a_1, \dots, a_n] = [b_1, \dots, b_n]$  rezultă  $a_i = b_i, \forall i = 1, \dots, n$ .

Mai observăm că  $[a_1, \dots, a_n] = [a_1, \dots, a_n, 0]$ . Prin urmare funcția care atașează oricărui program numărul său Gödel nu este injectivă.

De exemplu:  $1 = [0] = [0, 0] = [0, 0, 0] = \dots$ ; acest exemplu ne permite să definim numărul Gödel atașat secvenței vide (cu  $n = 0$ ) ca fiind 1.

Pentru orice  $i \geq 1$ , considerăm funcția  $(\cdot)_i: N \rightarrow N$  care pentru fiecare număr  $x = [a_1, a_2, \dots]$  calculează  $(x)_i = a_i$ . Aceste funcții sunt calculabile deoarece:

$$(x)_i = \min_{t \leq x} [-(p_i^{t+1} | x)].$$

Să observăm că  $(0)_i = (1)_i = 0, \forall i$ .



Mai introducem funcția  $Lt: N \rightarrow N$  care atașează fiecărui număr natural  $x$  numărul de ordine al celui mai mare număr prim care îl divide pe  $x$ . În exemplul de mai sus pentru  $x = 20 = [2, 0, 1, 0, \dots, 0]$  avem  $Lt(x) = 3$ .

Funcția  $Lt$  este calculabilă deoarece:

$$Lt(x) = \min_{i \leq x} \left[ (x)_i \neq 0 \wedge (\forall j)_{j \leq x} ((j \leq i) \vee ((x)_j = 0)) \right].$$

(se caută cel mai mic  $i$  cu  $x_i \neq 0$  și  $(x)_j = 0, \forall j > i$ ). Să observăm că:

- $Lt(0) = Lt(1) = 0$ ;
- $Lt([a_1, \dots, a_n]) = n \Leftrightarrow a_n \neq 0$  (unde  $n \geq 1$ );
- pentru orice  $z \in N \setminus \{0, 1\}$ , există și sunt unice  $n, a_1, \dots, a_n$  cu  $a_n \neq 0$  și  $[a_1, \dots, a_n] = z$  (pentru  $z = 1$  se obține  $n = 0$ , adică secvența vidă).

### Numărul atașat unui program

Începem prin a așeza variabilele de intrare  $(x_1, x_2, \dots)$ , variabila de ieșire  $y$  și variabilele de lucru  $(z_1, z_2, \dots)$  în următoarea ordine:  $y, x_1, z_1, x_2, z_2, \dots$

Variabilelor le atașăm pozițiile lor în acest șir prin funcția # astfel:  $\#(y) = 1; \#(x_i) = 2 \cdot i; \#(z_i) = 2 \cdot i + 1, \forall i \geq 1$ .

Fiecare instrucțiune poate avea atașată o etichetă. Vom numerota etichetele folosite în program cu  $E_1, E_2, \dots$ . Extindem funcția # la etichete astfel:  $\#(E_i) = i$ .

În continuare observăm că o instrucțiune  $I$  în limbajul  $S$  este bine determinată de:

- etichetarea ei;
- tipul instrucțiunii;
- unica variabilă  $v$  care intervine în ea.

Corespunzător, pentru fiecare instrucțiune definim trei numere  $a, b$  și  $c$  astfel:

$$a = \begin{cases} 0, & \text{dacă } I \text{ nu este etichetată} \\ \#(L), & \text{dacă } I \text{ este etichetată cu } L \end{cases}$$

$$b = \begin{cases} 0, & \text{dacă } I \text{ este de tipul } v \leftarrow v \\ 1, & \text{dacă } I \text{ este de tipul } v \leftarrow v + 1 \\ 2, & \text{dacă } I \text{ este de tipul } v \leftarrow v - 1 \\ \#(L) + 2, & \text{dacă } I \text{ este de tipul } \mathbf{if } v \neq 0 \mathbf{ goto } L \end{cases}$$

$$c = \#(v) - 1,$$

unde  $v$  este unica variabilă care apare în instrucțiunea  $I$ .

Putem extinde acum funcția # la instrucțiuni astfel:

$$\#(I) = \langle a, \langle b, c \rangle \rangle \text{ (numărul atașat instrucțiunii } I).$$

### Exemplul 21

Fie  $I$  următoarea instrucțiune:

$$[E_1] \quad x_1 \leftarrow x_1 + 1.$$

Atunci  $a = 1, b = 1, c = 1$  și deci  $\#(I) = \langle 1, \langle 1, 1 \rangle \rangle = \langle 1, 5 \rangle = 21$ .

Restricția funcției # la mulțimea instrucțiunilor, funcție având codomeniul  $N$ , este bijectivă, deoarece funcția  $\langle \cdot, \cdot \rangle$

este bijectivă. Să mai observăm că unica instrucțiune  $I$  cu  $\#(I) = 0$  este instrucțiunea neetichetată:  $y \leftarrow y$ .

Fie acum un program  $P$  constând, în ordine, din instrucțiunile  $I_1, \dots, I_n$ . Extindem atunci funcția # la programe astfel:  $\#(P) = [\#(I_1), \dots, \#(I_n)] - 1$ .

$\#(P)$  este numărul atașat programului  $P$ . Funcția # definită pe mulțimea programelor este calculabilă.

Ținând cont de observațiile făcute anterior asupra numărului Gödel atașat unei secvențe, rezultă că restricția funcției # la programe devine bijectivă, dacă ultima instrucțiune a programelor este diferită de instrucțiunea neetichetată  $y \leftarrow y$ ; cum efectul acesteia este nul, vom impune (fără a scădea din generalitate) regula: *Nici un program nu se poate termina cu instrucțiunea neetichetată  $y \leftarrow y$ .*

În acest mod fiecare număr natural poate fi privit ca un (unic) program în limbajul  $S$ .

O consecință imediată este că mulțimea programelor în limbajul  $S$  este numărabilă; acest fapt se putea deduce și în alte moduri, dar cel de mai sus permite chiar "numerotarea programelor".

### Exemplul 22

Căutăm programul  $P$  al cărui număr atașat este  $\#(P) = 199$ .

Observăm că:  $199 + 1 = 200 = 2^3 \cdot 5^2 = [3, 0, 2]$ , ceea ce arată că  $P$  este format din 3 instrucțiuni, ale căror numere atașate sunt în ordine 3, 0, 2:

$$3 = \langle a, \langle b, c \rangle \rangle \Rightarrow a = 2 \text{ și } \langle b, c \rangle = 0, \text{ deci } b = c = 0;$$

$$2 = \langle a, \langle b, c \rangle \rangle \Rightarrow a = 0 \text{ și } \langle b, c \rangle = 1, \text{ deci } b = 1, c = 0.$$

Rezultă că programul  $P$  căutat este următorul:

$$[E_2] \quad y \leftarrow y$$

$$y \leftarrow y$$

$$y \leftarrow y + 1$$

Să mai facem următoarele observații:

- programul vid are numărul atașat egal cu  $1 - 1 = 0$ ;
- corespondențele de mai sus sunt cele care au făcut necesare admiterea etichetării unor instrucțiuni diferite din același program cu aceeași etichetă.

### Teza lui Church. Problema opririi

Teza lui Church (1936) se exprimă astfel: *Date fiind numerele naturale  $x_1, \dots, x_n$ , numărul  $y$  poate fi "calculat" pe baza lor dacă și numai dacă există un program în limbajul  $S$  având la intrare valorile  $x_1, \dots, x_n$  și la ieșire valoarea  $y$ .*

Altfel spus, înțelegem prin algoritm care calculează valoarea  $y$  plecând de la valorile  $x_1, \dots, x_n$  un program în limbajul  $S$  care realizează acest lucru.

Evident, se pune problema dacă definiția cuvântului "algoritm" dată mai sus nu este prea restrictivă. Legat de aceasta, se impun următoarele precizări:

- noțiunea de algoritm nu poate fi definită decât pe baza unui limbaj de programare particular sau a unei "mașini matematice ideale"; de aceea teza lui Church nu poate fi



demonstrată ca o teoremă din matematică;

- toate încercările de a defini noțiunea de algoritm, încercări dintre care unele vor fi prezentate în continuare, au condus la definiții care s-au dovedit echivalente cu cea din enunțul tezei lui Church.

Aceste considerații ne determină să acceptăm definiția algoritmului așa cum apare ea în teza lui Church.

Suntem acum în măsură să prezentăm o primă problemă nedecidabilă, adică o problemă pentru care nu există un program în limbajul  $S$  care să o rezolve.

Este vorba de problema opririi (terminării) programelor.

Fie  $HALT$  predicatul binar definit astfel:

$HALT(x, x_2) = 1$ , programul  $P$  cu  $\#(P) = x_2$  se termină pentru valoarea de intrare  $x_1$ .

### Teorema 1

Predicatul  $HALT$  nu este calculabil.

### Demonstrație

Să presupunem prin absurd că predicatul  $HALT$  ar fi calculabil. Atunci putem considera următorul program (notat prin  $P$ ) care constă din unica instrucțiune:

[A] **if**  $HALT(x, x)$  **goto** A.

Atunci funcția de un argument calculată de  $P$  este:

$$P = \begin{cases} 1, & \text{dacă } HALT(x, x) = 1 \\ 0, & \text{dacă } HALT(x, x) = 0 \end{cases}$$

Fie  $y_0 = \#(P)$ . Atunci:  $HALT(x, y_0) = 1 \Leftrightarrow \psi_P^{(1)}(x) \downarrow$  (este nedefinită)  $\Leftrightarrow HALT(x, x) = 0$  (programul cu numărul  $y_0$  se termină pentru valoarea  $x$ ).

Punând  $x = y_0$ , obținem  $HALT(y_0, y_0) = 1 \Leftrightarrow HALT(y_0, y_0) = 0$ . Se ajunge astfel la o contradicție.

Cum predicatul  $HALT$  nu este calculabil, conform tezei lui Church ajungem la următorul rezultat:

**Corolar.** Problema opririi programelor este nedecidabilă, în următorul sens: nu există un algoritm care, pentru orice program scris în limbajul  $S$  și orice valori de intrare, să decidă dacă programul se termină pentru acele date de intrare.

### Va urma...

În următorul articol, care este de altfel și ultimul din această serie, vom prezenta alte probleme nedecidabile și vom trata problema decizibilității funcțiilor folosind diferite metode de abordare.

### Bibliografie

1. Martin D. Davis, Elaine J. Weyuker, *Computability, Complexity and Languages*, Academic Press, 1983
2. Christos H. Papadimitriou, *Computational Complexity*, Addison-Wesley, 1994

*Dl. prof. dr. Horia Georgescu este cadru didactic la Universitatea București, fondator și director științific al GInfo și poate fi contactat prin e-mail la adresa [g\\_horia@hotmail.com](mailto:g_horia@hotmail.com).*

## WINAMP 5 (2+3)

Cea mai cunoscută și utilizată aplicație capabilă să ruleze muzică și clipuri video utilizând puține resurse sistem, Winamp, a ajuns la versiunea 5.

Acest lucru se datorează faptului că versiunea 3, chiar dacă s-a dorit să fie mai performantă decât versiunea 2, a fost considerată un eșec de cei de la NullSoft, mulți utilizatori ai acestui produs revenind la versiunea 2.

În primăvara acestui an a apărut prima versiune a produsului Winamp 5. Această versiune era *superprealpha* (m-am mirat foarte tare când am văzut numele versiunii) și nu avea nimic în plus față de produsul Winamp 2, din contră era plin de *bug*-uri.

De curând a apărut versiunea beta 10 care combină, într-un singur pachet, cele mai bune elemente prezente în cadrul celor două versiuni anterioare.

La facilitățile oferite de Winamp 2 și Winamp 3 s-au adăugat elemente noi, cum ar fi posibilitatea de a avea interfețe grafice cu altă formă decât cea cunoscută și ferestre translucide.

În momentul în care am pornit aplicația am rămas blocat. Nu am putut decât să-mi zic "Uau!".

Cu ajutorul acestui produs se pot viziona filme și videoclipuri și nu mai vorbesc de calitatea sunetului care este cunoscută chiar de la versiunea 1.

Fanii produselor Winamp au acum posibilitatea de a crea in-

terfețe grafice care să aibă cele mai ciudate forme, Winamp 5 având inclus un pachet de *scripting* pentru a facilita acest lucru.

Cei de la NullSoft au anunțat adăugarea a două formate proprietare NSA (NullSoft Audio) și NSV (NullSoft Video).

Chiar dacă este încă la versiunea beta, acest produs merită să fie instalat. Mie mi-a plăcut.

Claudiu Soroiu

