Un programme peut être vu comme un grand nombre

Tout programme C est une chaîne de caractères.
P est ici une fonction C, paramétrée par un entier x (on a utilisé le type int mais on aurait plutôt dû utiliser des entiers de taille arbitrairement grande, comme ceux de la bibliothèque GMP).

```
> P := "int P (int x) { return 2*x+3; }";
```

$$P := \text{"int P (int x) \{ return 2*x+3; \}"} \tag{1}$$

La suite des codes ASCII des caractères de P

```
> L := convert (P, bytes);
```

$$L := [105, 110, 116, 32, 80, 32, 40, 105, 110, 116, 32, 120, 41, 32, 123, 32, 114, 101, 116, 117, 114, 110, 32, 50, 42, 120, 43, 51, 59, 32, 125] \tag{2}$$

Le programme P peut être vu comme un grand entier, noté p (on voit la suite de nombres ci-dessus comme une suite de chiffres en base 128).

```
> p := add (L[i] * 128**(i-1), i = 1 .. nops (L));
```

$$p := 206105411674394777170124753571416671765150487566453001793038\backslash 006121 \tag{3}$$

La transformation inverse

```
> L := convert (p, 'base', 128);
```

$$L := [105, 110, 116, 32, 80, 32, 40, 105, 110, 116, 32, 120, 41, 32, 123, 32, 114, 101, 116, 117, 114, 110, 32, 50, 42, 120, 43, 51, 59, 32, 125] \tag{4}$$

```
> P := convert (L, bytes);
```

$$P := \text{"int P (int x) \{ return 2*x+3; \}"} \tag{5}$$

En particulier, la fonction M, qu'on retrouvera dans le raisonnement de Turing, peut être codée par un nombre m.

```
> M := "void M (int q) { if (A (q,q)) { while (1) ; } }";
```

$$M := \text{"void M (int q) \{ if (A (q,q)) \{ while (1) ; \} \}"} \tag{6}$$

```
> L := convert (M, bytes);
```

$$L := [118, 111, 105, 100, 32, 77, 32, 40, 105, 110, 116, 32, 113, 41, 32, 123, 32, 105, 102, 32, 40, 65, 32, 40, 113, 44, 113, 41, 41, 32, 123, 32, 119, 104, 105, 108, 101, 32, 40, 49, 41, 32, 59, 32, 125, 32, 125] \tag{7}$$

```
> m := add (L[i] * 128**(i-1), i = 1 .. nops (L));
```

$$m := 107019482240385125058342733387435809206084558149949655167926\backslash 751365252356559827044264600534109876837 4 \tag{8}$$