

NOM: G LOESSPrénoms: Paul YNotes: I 4 II 6 III 4 IV 5

PG211

19

Durée: deux heures et dix-sept secondes.

Accès Internet limité à la page personnelle de Paul Y Gloess (<http://www.enseirb.fr/~gloess>) et toute sa descendance, y compris la documentation en ligne de CHIP ; accès au système CHIP autorisé ; communication ("e-mail", "chat", ou autre procédé) interdite.



Pure and Impure Prolog: which is which?

### I. [4 pts] Pure Prolog parallel member

On rappelle que **Prolog pur** ("pure Prolog") n'utilise dans ses programmes que des clauses de Horn définies, écrites en syntaxe Edinburg (adoptée ici) sous la forme "c." ou la forme "c :- h<sub>1</sub>, ..., h<sub>n</sub>." où la conclusion "c" et les hypothèses "h<sub>1</sub>", ..., "h<sub>n</sub>" sont des littéraux positifs. Prolog pur n'autorise ni coupure, ni négation, ni symbole prédéfini tel que "=", "+", "\*", "..."; on utilisera la notation Edinburg pour les listes: [] pour la liste vide; [E | L] pour le "cons" de E et L.

On définira récursivement (sans utiliser de prédicat auxiliaire), en deux clauses de Prolog pur, le prédicat "member" d'arité 4 muni de la sémantique:

$$\text{member}(E1, L1, E2, L2) \stackrel{\text{def}}{=} \begin{array}{l} E1 \text{ est membre de la liste } L1, \\ E2 \text{ est membre de la liste } L2, \\ \text{et } E1 \text{ et } E2 \text{ sont à la même position, à partir de la gauche,} \\ \text{dans les listes } L1 \text{ et } L2 \text{ respectivement.} \end{array}$$

On utilisera la variable anonyme notée "\_" à chaque fois que cela sera possible.

$$\begin{array}{l} \text{member}(E1, [E1|_], E2, [E2|_]), \\ \text{member}(E1, [_|L1], E2, [_|L2]) :- \\ \quad \text{member}(E1, L1, E2, L2). \end{array}$$

**II. [6 pts] LP search tree**

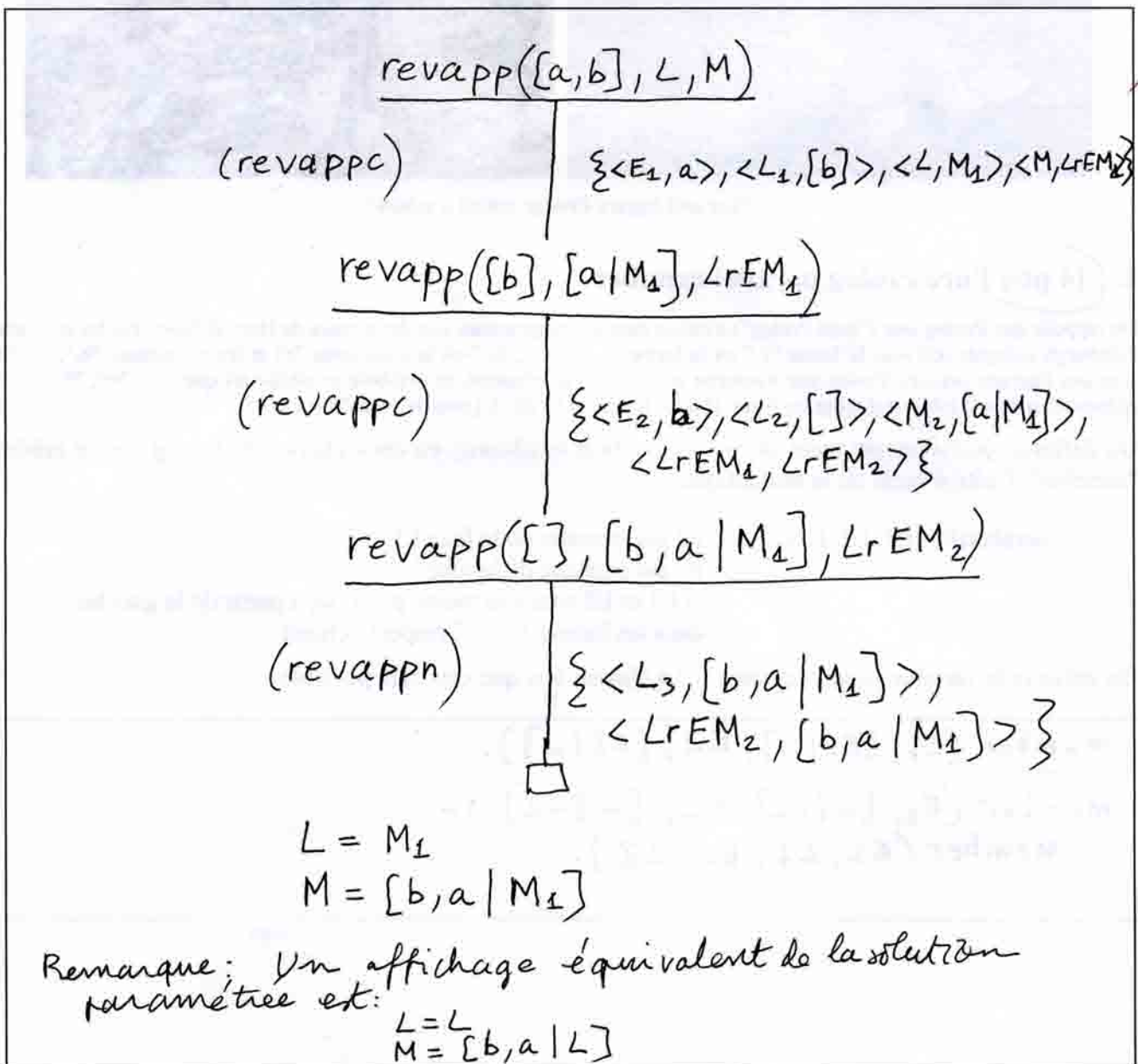
On rappelle la définition du prédicat “revapp” de miroir et concaténation:

$revapp([], L, L)$  . (revappn)  
 $revapp([E | L], M, LrEM) :- revapp(L, [E | M], LrEM).$  (revappc)

a. Rappeler la sémantique informelle de “revapp” :

$revapp(L, M, LrM)$   
 veut dire que  $LrM$  est la concaténation du miroir de la liste  $L$  et de la liste  $M$ .

b. Dessiner l’arbre de recherche du but “ $revapp([a, b], L, M)$ ” en faisant apparaître sur chaque arc le nom de la clause et la substitution unificatrice utilisés. On rappelle que “[ $a, b$ ]” est une abréviation pour “[ $a | [b | []]$ ]”.



4  
III. [5 pts] CLP search tree

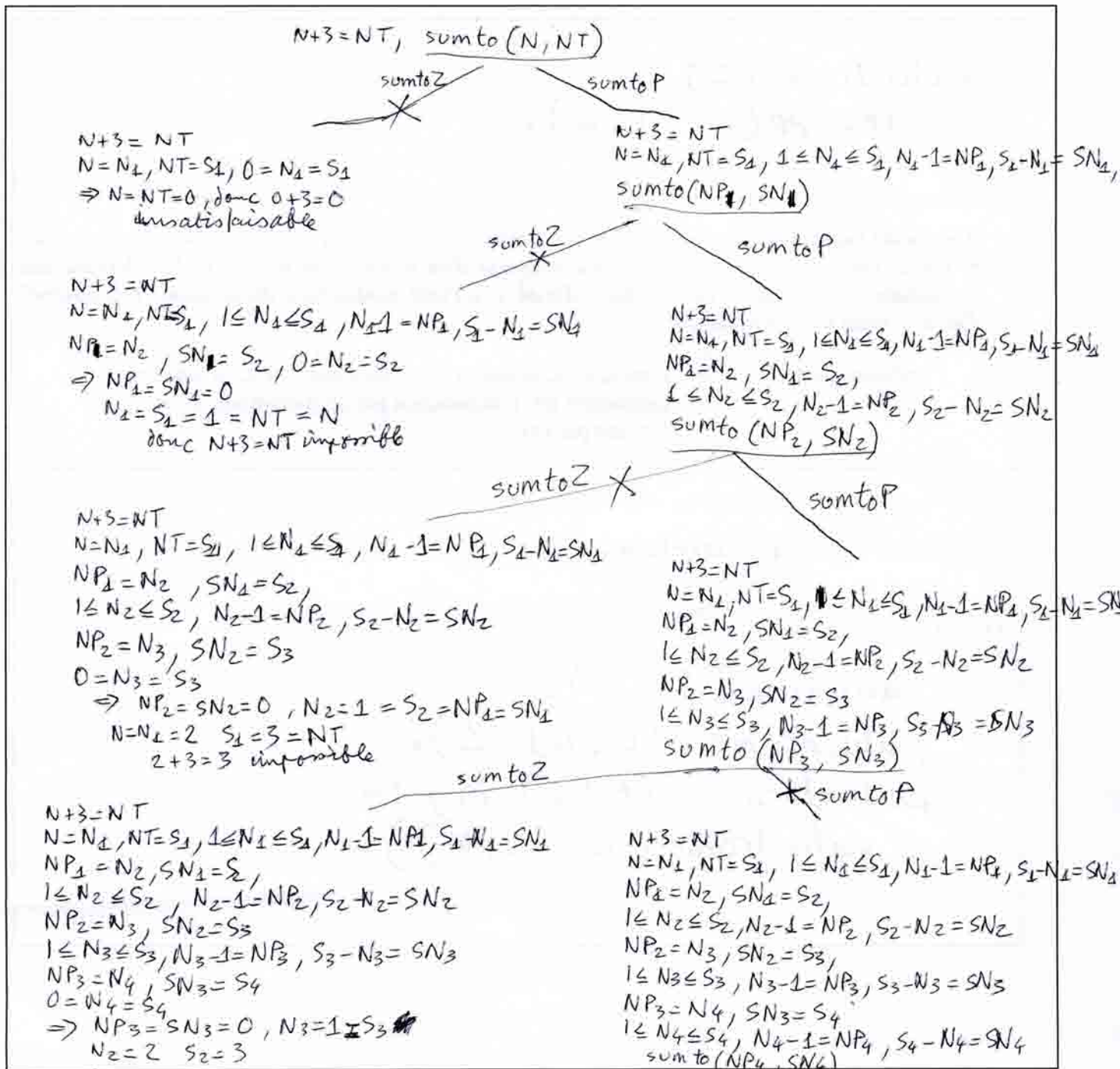
*Sort ou casse! (-1)*

On se place dans le cadre CLP(Q<sub>lin</sub>), de la programmation logique avec contraintes linéaires en nombres rationnels. On reprend l'exemple introductif du cours, sous forme normalisée, dans la syntaxe CHIP:

sumto(N, S) :- 0 ≤ N, 0 ≤ S. (sumtoZ)

sumto(N, S) :- 1 ≤ N, N ≤ S, N-1 ≤ NP, S-N ≤ SN, sumto(NP, SN). (sumtoP)

Dessiner l'arbre de recherche du but "N+3 ≤ NT, sumto(N, NT)" en faisant apparaître l'unique solution et les quatre ensembles de contraintes insatisfaisables et quatre branches coupées correspondantes. [On pourra utiliser la notation "=" pour "≤", et "≤" pour "≤"; ne pas oublier de numéroter les variables en fonction de la profondeur.]



N1=3 S1=6=NT  
N=3, NT=6  
OK solution

-3/4-  
N=N1=N2+1=N3+2=N4+3  
NT=S1=S2+N4+3=S3+N2+N4+3=  
NT=S4+N3+N2+N4+3=S4+N4+1+N4+2+N4+3  
=S4+3N4+6 ⇒ N4+6=S4+3N4+6 ⇒ S4+2N4=C  
impossible car S4 ≥ 1 et N4 ≥ 1

**IV. [5 pts] Pure Prolog palindrome**

Un palindrome est une liste de la forme  $[e_1, \dots, e_k, \dots, e_n]$ , avec  $0 \leq n$ , telle que pour tout entier  $k$  avec  $1 \leq k \leq n$ , on ait:

$$e_k = e_{n+1-k}.$$

Exemples de palindromes: [], [i, c, i], [a, n, n, a], [m, a, d, a, m].

On définira dans la suite en Prolog pur le prédicat "palindrome" de deux façons différentes et indépendantes, avec la sémantique:

palindrome(L) <sup>def</sup>  $\equiv$  L est un palindrome.

- a. (Première façon) Définir le prédicat "palindrome" d'arité 1 en une ou deux clauses, en n'utilisant pas d'autre prédicat que "revapp", défini à la question II.

```
palindrome(L) :-
    revapp(L, [], L).
```

- b. (Deuxième façon) On remarquera qu'un palindrome non vide est la concaténation d'une liste et de la même liste inversée, avec éventuelle insertion d'un élément au milieu. On en déduira une implantation du prédicat "palindrome", d'arité 1, à l'aide uniquement du prédicat "palindrome" d'arité 2, muni de la sémantique:

palindrome(L, Lr) <sup>def</sup>  $\equiv$  L est la concaténation d'un palindrome et de la liste Lr, (autrement dit, L commence par un palindrome et se termine par Lr).

```
%%%
%%% Définition de palindrome d'arité 1 en une clause:
%%%
palindrome(L) :- palindrome(L, []).

%%%
%%% Définition récursive de palindrome d'arité 2 en trois clauses,
%%% n'utilisant pas d'autre prédicat que palindrome d'arité 2:
%%%
palindrome(L, L).
palindrome([E|L], L).
palindrome([E|L], M) :-
    palindrome(L, [E|M]).
```