

Master DAC – LODAS – TD « Apprentissage Symbolique »

Unification

Trouver, s'il existe, l'unificateur le plus général des termes t_1 et t_2 , t_1 et t_3 , et t_2 et t_3

$$t_1 = f(U, f(V, W)),$$

$$t_2 = f(T, f(R, R)),$$

$$t_3 = f(f(h(X), X), f(f(X, f(Y, Y)), f(X, f(h(Y), h(Z))))))$$

Principe de résolution

exercice 1

Dans cet exercice, on supposera que nous avons un ensemble de variables $V = \{x, y, z\}$, un ensemble de fonctions $F = \{a, b, f, g, \dots\}$ et un ensemble de prédicats $P = \{P, Q, R, S, \dots\}$. Soit quatre ensembles de clause, $S = \{C1, C2,$

$C3, C4, C5, C6\}$ avec

$$C1: \neg P(x) \vee Q(f(x), x)$$

$$C2: \neg P(x) \vee Q(y, x) \vee R(y)$$

$$C3: \neg R(y) \vee \neg S(y) \vee \neg P(x) \vee \neg Q(y, x)$$

$$C4: \neg S(y) \vee R(y)$$

$$C5: P(a)$$

$$C6: \neg R(y) \vee S(y)$$

Démontrer l'insatisfiabilité de l'ensemble de clauses en faisant appel au principe de résolution.

exercice 2

Soit les ensembles de clauses $S1 = \{C1, C2, C3, C4\}$ et $S2 = \{C5, C6, C7\}$ avec

$$C1: \neg R(f(x)) \vee R(x)$$

$$C2: \neg R(g(x)) \vee R(x)$$

$$C3: R(g(f(g(a))))$$

$$C4: \neg R(a)$$

$$C5: \neg P(g(a, g(b, g(c, h))), g(d, g(e, h)), z)$$

$$C6: P(h, x, x)$$

$$C7: P(g(x, y), z, g(x, u)) \vee \neg P(y, z, u)$$

Démontrer l'insatisfiabilité de $S1$ et de $S2$ en faisant appel au principe de résolution

exercice 3

$$C1 : \neg \text{tableau}(x) \vee \neg a \text{ peint}(y, x) \vee \neg a \text{ peint}(z, x) \vee \text{eq}(y, z)$$

$$C2 : \text{tableau}(\ll \text{LesMenines} \gg)$$

$$C3 : \neg \text{tableau}(x) \vee a_peint(b, x)$$

$$C4 : \text{inspiré_par}(\text{Picasso}, \text{Vélazquez})$$

$$C5 : a_peint(\text{Velazquez}, \ll \text{LesMenines} \gg)$$

$$C6 : \neg \text{dans_série}(x, \ll \text{LesMenines} \gg) \vee \text{tableau}(x)$$

$$C7 : \neg \text{peintre}(x) \vee a_peint(x, a)$$

$$C8 : \neg \text{dans_série}(x, \ll \text{LesMenines} \gg) \vee a_peint(\text{Picasso}, x)$$

$$C9 : \neg \text{eq}(\text{Picasso}, \text{Velazquez})$$

$$C10 : \neg \text{peintre}(x) \vee \text{tableau}(a)$$

$$C11 : \neg \text{tableau}(x) \vee \text{peintre}(b)$$

$$C12 : \neg \text{tableau}(x) \vee \neg \text{dans_série}(x, \ll \text{LesMenines} \gg) \vee \neg a_peint(\text{Velazquez}, x)$$

Montrer en utilisant la règle de résolution que $C1, C2, C3, C4, C5, C6, C7, C8, C9, C10, C11 \vdash C12$.

Exercices de PROLOG

Le prédicat **diviseur(M,N)** qui comporte deux arguments, M et N, est évalué à Vrai quand le premier argument M est un diviseur du second N et à faux sinon. Ainsi, `diviseur(2, 8)` et `diviseur(3, 6)` sont évalués à Vrai, tandis que `diviseur(3, 8)` et `diviseur(5, 6)` ne le sont pas.

Ecrire l'ensemble des clauses qui définissent ce prédicat en remarquant que `diviseur(N, P) = diviseur(N, P-N)` et en faisant appel uniquement à la soustraction.

Programmation en PROLOG: le crible d'Ératosthène

On désire construire un programme PROLOG qui engendre tous les nombres premiers inférieurs à N par la méthode dite du "crible d'Ératosthène". Pour cela on engendre les entiers de 2 à N, on les met dans une liste L où ils sont rangés en *ordre croissant*, puis on appelle récursivement la fonction crible qui élimine les entiers multiples de la tête de liste.

1. Le prédicat **entiers(N,L)** engendre les entiers compris entre deux et N pour les ranger dans une liste L en *ordre croissant*. En faisant appel à un prédicat auxiliaire **entiers_aux(N,L,A)** qui comprend un accumulateur A, déterminer l'ensemble des clauses qui définissent le prédicat **entiers(N,L)**.

2. Le prédicat **diviseur(M,N)** qui comporte deux arguments, M et N, est évalué à Vrai quand le premier argument M est un diviseur du second N et à faux sinon. Ainsi, `diviseur(2, 8)` et `diviseur(3, 6)` sont évalués à Vrai, tandis que `diviseur(3, 8)` et `diviseur(5, 6)` ne le sont pas. Ecrire l'ensemble des clauses qui définissent ce prédicat en remarquant que `diviseur(N, P) = diviseur(N, P-N)` et en faisant appel uniquement à la soustraction.

3. Le prédicat **filtre(L,P)** comporte, lui aussi deux arguments, L et P. Lorsque L est instancié par une liste d'entiers rangés en ordre croissant, l'évaluation de `filtre(L,P)` instancie P avec la liste L privée des multiples de la tête (c'est-à-dire du premier élément) de L. Ecrire l'ensemble des clauses qui définissent ce prédicat.

4. Déterminer l'ensemble des clauses qui définissent le prédicat à deux arguments **premiers(N,L)** dont l'évaluation renvoie la liste des nombres premiers inférieurs à un entier N quelconque.

Remarque : on pourra faire appel à un prédicat auxiliaire qui, partant des entiers de 2 à N, filtre successivement les multiples de la tête de liste.

Programmation en PROLOG: permutation des entiers

On désire construire un programme PROLOG qui engendre successivement toutes les permutations des éléments d'une liste. Pour cela, on fait appel au prédicat `appartient(E, L, NL)` qui teste l'appartenance de E à L et qui instancie NL avec la liste L à laquelle on soustrait une occurrence de l'élément E

Exemple 1 : l'appel à `appartient(E, [a, b, c, d, a], L)`, est satisfait par les solutions suivantes :

1) E=a, L=[b, c, d, a], 2) E=b, L=[a, c, d, a], 3) E=c, L=[a, b, d, a], 4) E=d, L=[a, b, c, a], E=a, L=[a, b, c, d].

Exemple 2 : l'appel à `permutation([a, b, c], L)` renvoie les solutions suivantes :

1) L=[a, b, c], 2) L=[a, c, b], 3) L=[b, a, c], 4) L=[b, c, a], 5) L=[c, a, b], 6) L=[c, b, a]

1- construire le prédicat `appartient(E, L, NL)`.

2- construire le prédicat `permutation(L, NL)` qui instancie NL avec toutes les permutations successives de L.