DM3 - Sudoku

Léo SAMUEL

31 janvier 2021

1 Présentation du jeu de sudoku

1.1 Préliminaires

Question 1 :

```
let rec appartient x l =
  match l with
  | [] -> false
  | t::_ when t=x -> true
  | t::q -> appartient x q
;;
```

Question 2 :

```
let rec suppression x l=
  match l with
  | [] -> []
  | t::q when t=x -> suppression x q
  | t::q -> t::(suppression x q)
;;
```

Question 3 :

```
let ajoute x l=
  let rec aux x l flag=
    match l with
    | [] when not flag -> [x]
    | [] -> []
    | t::q when t=x -> t::(aux x q true)
    | t::q -> t::(aux x q flag)
  in aux x l false
;;
```

Question 4 :

```
let rec indice (b,r)=
  let i = (b mod 3)*3 + (r mod 3) in
  let j = (b/3)*3 + r/3 in
  (i,j)
;;
```

2 Codage de la formule initiale

2.1 Formule logique décrivant la règle du jeu

Question 1 :

(a) :

Soit $(i, j) \in [\![0, 8]\!]^2, \vee_{k=1}^{9} x_{(i,j)}^{k}$

(b) :

On a : $\forall (i, j) \in [\![0, 8]\!]^2, \exists k \in [\![1, 9]\!], x_{(i,j)}^{k}$

(c) :

On a : $(K_1) \equiv \wedge_{i=0}^{8}(\wedge_{j=0}^{8}(\vee_{k=1}^{9} x_{(i,j)}^{k}))$

(d) :

Il y a $8 \cdot 8 = 81$ clauses

(e) :

```
let case1 () =
  let res = ref [] in
  for i=0 to 8 do
    for j=0 to 8 do
      let temp = ref [] in
      for k=1 to 9 do
        temp:= (X(i,j,k)) :: !temp;
      done;
    res:= !temp :: !res;
    done;
  done;
  !res
;;
```

Question 2 :

On a : $\forall i \in [\![0, 8]\!], \forall k \in [\![1, 9]\!], \exists j \in [\![0, 8]\!], x_{(i,j)}^{k}$
On obtient alors la clause $L_1 \equiv \wedge_{i=0}^{8}(\wedge_{k=1}^{9}(\vee_{j=0}^{8} x_{(i,j)}^{k}))$

Question 3 :

(a) :

On a $(C_1) \equiv \wedge_{j=0}^{8}(\wedge_{k=1}^{9}(\vee_{i=0}^{8} x_{(i,j)}^{k}))$
Et $(B_1) \equiv \wedge_{b=0}^{8}(\wedge_{k=1}^{9}(\vee_{r=0}^{8} x_{\text{indice}(b,r)}^{k}))$

(b) :

```
let bloc1 ()=
  let res = ref [] in
  for b=0 to 8 do
    for k=1 to 9 do
      let temp = ref [] in
      for r=0 to 8 do
        let i,j = indice (b,r) in
        temp:= X(i,j,k) :: !temp;
      done;
      res:= !temp :: !res
    done;
  done;
  !res
;;
```

Question 4 :

(a) :

La ligne i ne contient pas deux fois la valeur k lorsque : $\forall j_1, j_2 \in [\![0,8]\!], j_1 < j_2 \Rightarrow (\neg x^k_{(i,j_1)} \vee \neg x^k_{(i,j_2)})$

On obtient alors $\wedge_{b=0}^{8} \wedge_{k=1}^{9} (\neg x^k_{(i,j_1)} \vee \neg x^k_{(i,j_2)})$

(b) :

On a $\forall i \in [\![0,8]\!], \forall k \in [\![1,9]\!], \forall j_1, j_2 \in [\![0,8]\!], j_1 \neq j_2 \Rightarrow (\neg x^k_{(i,j_1)} \vee \neg x^k_{(i,j_2)})$

D'où $(L_2) \equiv \wedge_{i=0}^{8} \wedge_{k=1}^{9} \wedge_{j_1=0}^{8} \wedge_{j_2=0}^{8} (\neg x^k_{(i,j_1)} \vee \neg x^k_{(i,j_2)})$

(c) :

Il y a $9 \cdot 9 \cdot \frac{9 \cdot 8}{2} = 2916$ clauses.

(d) :

```
let ligne2 ()=
  let res = ref [] in
  for i=0 to 8 do
    for k=1 to 9 do
      for j1=0 to 7 do
        for j2=j1 to 8 do
          res:= [NonX(i,j1,k);NonX(i,j2,k)] :: !res;
        done;
      done;
    done;
  done;
  !res
;;
```

Question 5 :

On a

$$(C_2) \equiv \wedge_{j=0}^{8} \wedge_{k=1}^{9} \wedge_{i_1=0}^{7} \wedge_{i_2=i_1+1}^{8} \left(\neg x^{k}_{(i_1,j)} \vee \neg x^{k}_{(i_2,j)}\right)$$
$$(K_2) \equiv \wedge_{i=0}^{8} \wedge_{j=0}^{8} \wedge_{k_1=1}^{8} \wedge_{k_2=k_1+1}^{9} \left(\neg x^{k}_{(i,j)1} \vee \neg x^{k}_{(i,j)2}\right)$$
$$(B_2) \equiv \wedge_{b=0}^{8} \wedge_{k=1}^{9} \wedge_{r_1=0}^{7} \wedge_{r_2=r_1+1}^{8} \left(\neg x^{k}_{\text{indice}(b,r_1)} \vee \neg x^{k}_{\text{indice}(b,r_2)}\right)$$

2.2 Formule logique décrivant la grille initiale

Question 1 :

```
let donnees t=
  let res = ref [] in
  for i=0 to 8 do
    for j=0 to 8 do
      match t.(i).(j) with
      | 0 -> ()
      | k ->  for q=1 to 9 do
                 if q=k then
                   res:= [X(i,j,q)] :: !res
                 else
                   res:= [NonX(i,j,q)]::!res
               done;
    done;
  done;
;;
```

Question 2 :

(a) :

On a $b = 3\left\lfloor \frac{i}{3} \right\rfloor + \left\lfloor \frac{j}{3} \right\rfloor$

(b) :

```
let interdites_ij t i j=
  let res = ref [] in
  let b = 3*(i/3)+(j/3) in
  for l=0 to 8 do
    if t.(l).(j) <> 0 then
      res := ajoute (NonX(i,j,t.(l).(j))) !res;
    if t.(i).(l) <> 0 then
      res := ajoute (NonX(i,j,t.(i).(l))) !res;
    let i1,j1 = indice(b,l) in
    if t.(i1).(j1) <> 0 then
      res := ajoute (NonX(i,j,t.(i1).(j1))) !res;
  done;
!res ;;
```

(c) :

```
let interdites t =
  let res = ref [] in
  for i = 0 to 8 do
    for j = 0 to 8 do
      if t.(i).(j)=0 then
        res := (interdites_ij t i j) :: !res;
    done;
  done;
!res;;
```

Question 3 :

Chaque case remplie est associé à 9 clauses. De plus, chaque case non remplie est associé à 8 clauses. On notant r le nomvre de cases remplis dans la grille initiale.

Alors on a $9r + 8(81 - r)$ clauses. Dans le pire des cas, la grille est remplis et $r = 81$. Dans le pire des cas, il y a $9^3 = 729$ clauses.

3 Résolution

3.1 Propagation unitaire

Question 1 :

Depuis une grille initiale, on obtient une unique grille finale. Alors il existe une unique solution. D'où, il y a une seule valuation satisfaisant F_{initiale}.

Question 2 :

Il y a 729 clauses alors la table de vérité contient $2^{729} \approx 10^{219}$ lignes.

Question 4 :

On a $x^1_{(0,0)} \wedge (x^4_{(2,2)} \vee x^6_{(3,6)} \vee x^7_{(7,7)}) \wedge (\neg x^1_{(0,0)} \vee \neg x^6_{(3,6)})$
Puis $(x^4_{(2,2)} \vee x^6_{(3,6)} \vee x^7_{(7,7)}) \wedge (\neg x^6_{(3,6)})$
Finalement $(x^4_{(2,2)} \vee x^7_{(7,7)})$

Question 5 :

(a) :

Les possibilités sont 1,2,4 et 7. Or il y a un 7 présent dans le bloc 0 à la ligne 2 et dans le bloc 1 à la ligne 1 alors la seule possibilitée pour placer un 7 dans le bloc 2 est sur la ligne 0 dans la case libre.

Question 6 :

```
let rec nouveau_lit_isole f=
  match f with
  | [] -> X(-1,-1,-1)
  | [c]::q -> c
  | _::q -> nouveau_lit_isole q
;;
```

Question 7 :

```
let non x=
  match x with
  | X(i,j,k) -> NonX(i,j,k)
  | NonX(i,j,k) -> X(i,j,k)
;;

let rec simplification l f=
  match f with
  | [] -> []
  | t::q when appartient l t -> simplification l q
  | t::q -> suppression (non l) t :: simplification l q
;;
```

Question 8 :

```
let rec propagation t f=
  match nouveau_lit_isole f with
  | X(-1,-1,-1) -> f
  | X(i,j,k) -> t.(i).(j) <-k;
                propagation t (simplification (X(i,j,k)) f)
  | l -> propagation t (simplification l f)
;;
```

3.2 Règle du littéral infructueux

Question 2 :

```
let variables f =
  let res = ref [] in
  let rec ajouter_clause c =
    match c with
    | [] -> ()
    | X(i,j,k)::q ->  res := ajoute (X(i,j,k)) !res;
                      ajouter_clause q
    | NonX(i,j,k)::q -> res := ajoute (X(i,j,k)) !res;
                        ajouter_clause q in
  let rec ajouter_formule f =
    match f with
    | [] -> ()
    | t::q -> ajouter_clause t;
                ajouter_formule q
  in ajouter_formule f;
  !res
;;
```

Question 3 :

```
let copie_matrice m=
  let n,p=Array.length m,Array.length m.(0) in
  let nvmat = Array.make_matrix n p 0 in
  for i=0 to n do
    for j=0 to p do
      nvmat.(i).(j) <- m.(i).(j);
    done;
  done;
  nvmat
;;

let deduction t x f =
  let t1 = copie_matrice t in
  match x with
  | X(i,j,k) when propagation t1 ([NonX(i,j,k)] :: f) = [[]] -> 1
  | _ -> let t2 = copie_matrice t
  in
  if propagation t2 ([x] :: f) = [[]] then
    -1
  else
    0
;;
```