

Rapport Projet INF402

Takuzu

Présentation :

Le Takuzu est un jeu de réflexion consistant à remplir une grille par des 0 et des 1 selon des règles particulières.

Le Takuzu est composé d'une grille allant de 6x6 à 14x14 en générale (nous utiliseront des grilles de 10x10 pour commencer).. Chaque case peut prendre les valeurs de 0 ou 1.

La complétion se fait selon les règles suivantes.

Règles du Takuzu :

- autant de 1 et de 0 sur chaque ligne et sur chaque colonne
- pas plus de 2 chiffres identiques côte à côte
- 2 lignes ou 2 colonnes ne peuvent être identiques

Exemple :

Grille de Takuzu 4x4 initiale, puis complétée :

	1		0
		0	
	0		
1	1		0

0	1	1	0
1	0	0	1
0	0	1	1
1	1	0	0

Formes logiques :

La variable x_{ij} avec $i, j \in \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}$ décrit la présence du chiffre 1 dans la case située dans la $j^{\text{ème}}$ ligne, $i^{\text{ème}}$ colonne si vrai, 0 si fausse. N désigne la taille du tableau.

Pour la suite, $\prod_a^b x$ désigne la conjonction des x allant de a à b . De même, $\sum_a^b x$ désigne la disjonction des x allant de a à b

Pas plus de 2 chiffres identiques côte à côte :

$$\bigwedge_{i=1}^n \bigwedge_{j=1}^{n-2} \left((X_{i,j} \vee X_{i,j+1} \vee X_{i,j+2}) \wedge (\neg X_{i,j} \vee \neg X_{i,j+1} \vee \neg X_{i,j+2}) \right) \wedge \left((X_{j,i} \vee X_{j+1,i} \vee X_{j+2,i}) \wedge (\neg X_{j,i} \vee \neg X_{j+1,i} \vee \neg X_{j+2,i}) \right)$$

Autant de 1 et de 0 sur chaque ligne et sur chaque colonne

$$\bigwedge_{i=1}^n \forall (k_1, \dots, k_{\frac{n}{2}+1}) \left(\bigvee_{i=1}^{\frac{n}{2}+1} \neg X_{ik,j} \wedge \bigvee_{j=1}^{\frac{n}{2}+1} X_{ik,j} \right)$$

2 lignes ou 2 colonnes ne peuvent être identiques

$$\forall i_1, i_2, \exists j \text{ tq } \neg (X_{i_1,j} \leftrightarrow X_{i_2,j}) \wedge \forall j_1, j_2, \exists i \text{ tq } \neg (X_{i,j_1} \leftrightarrow X_{i,j_2})$$

Les différents formats

La grille de Takuzu sera créée et lu sous un format texte de n lignes de n entiers compris entre 0 et 2, 0 et 1 indiquant un remplissage de la case correspondante par 0 ou 1, et 2 un non remplissage.

Le format Conjonctif sera formé de case contenant 1 (= vrai) écrite (i,j) ou 0 (= fausse) écrite -(i,j), séparé par des "ou" logiques "+" et des "et" logiques ".", sous forme de clauses.

Le format **Dimacs n-Sat** sera **formé de clauses**, comportant des variables/cases portant l'indice de la case (ligne * taille + colonne) dans le cas d'une case avec 1, et toujours avec – devant dans le cas d'une case avec 0. Les variables sont séparés par des espaces, et les clauses par des 0.

Le header spécifique se trouve au début ("p cnf <nbvars> <nbclauses>")

Le format Dimacs 3-Sat est identique mais formé de clauses de 3 variables chacune uniquement

Exemples de grilles résolues par le programme

Grille Vierge 6x6

0	1	0	1	1	0
1	0	0	1	0	1
0	0	1	0	1	1
1	1	0	1	0	0
0	1	1	0	0	1
1	0	1	0	1	0

1					
		0		0	
1					
		1		1	

0	1	0	0	1	1
1	0	1	1	0	0
0	1	0	1	0	1
1	0	1	0	1	0
1	0	0	1	0	1
0	1	1	0	1	0

Grille prérempli aléatoirement 6x6

	0				
	0				
	0				

Ok

Validation par clause unitaire de -2

Validation par clause unitaire de -8

Validation par clause unitaire de -14

Clause vide, retour en arrière

Insatisfaisable. La grille de depart n'est peut etre pas valide

Grille erronée

Implémentation

Dans ce rapport sera décrite l'implémentation en **pseudo-code tiré du Visual Basic**, pour permettre une lecture plus lisible. Des éléments ont donc été volontairement retirés ou mis à un niveau plus haut : gestion des erreurs, affichage, threading, fonctions mathématiques etc.

Le **Sat Solver codé est une implémentation simple de l'algorithme DPLL**, avec une sauvegarde de l'ensemble des clauses et des variables à chaque attribution arbitraire.

Le code complet est disponible ici : <https://dev.azure.com/CochonCorp/ProjetLogique2019>.

Global

```
'Variables intégré en VB :  
' vbCrLf = Saut de ligne  
' Nothing = pas d'instance d'objet  
' False = 0 ; True = tout le reste (-1 par défaut)  
Public n As Integer = TailledeGrille()
```

Forme Conjonctive

```
Public Function CreateConjonctive()  
    New StreamWriter = CreateTextFile("Takuzu_Conjonctive.txt")  
    'Préremplissage  
    Dim j = 0  
    While Not EndOfStream  
        Dim s = ReadLine()  
        For i = 0 To n - 1  
            If s(i) = "0" Then  
                WriteLine("-({0},{1}) .", j, i)  
            If s(i) = "1" Then  
                WriteLine("({0},{1}) .", j, i)  
            End If  
        Next  
        j += 1  
    End While  
    'Règle 1  
    For i = 0 To n - 1  
        For j = 0 To n - 3  
            Write("({0},{1}) + ({0},{2}) + ({0},{3})) ." + vbCrLf + "(-({0},{1})  
+ -({0},{2}) + -({0},{3})) ." + vbCrLf + "(({1},{0}) + ({2},{0}) + ({3},{0})) ." +  
vbCrLf + "(-({1},{0}) + -({2},{0}) + -({3},{0})) ." + vbCrLf, i, j, j + 1, j + 2)  
        Next  
    Next  
    'Règle 2 - ligne (colonne : même code en inversant j et (i1 ou i2))  
    'Increment(tab) ajoute 1 à tab. ex:  
    'Increment(0010)=0011; Increment(0011)=0100; Increment(0100)=0101; etc.  
    Dim tab(n - 1) As Boolean  
    For i1 = 0 To n - 2  
        For i2 = i1 + 1 To n - 1  
            For k = 0 To 2n - 1  
                For j = 0 To n - 2  
                    If tab(j) Then  
                        Write("({0},{1}) + ({0},{2}) + ", j, i1, i2)  
                    Else
```

```

        Write("-({0},{1}) + -({0},{2}) + ", j, i1, i2)
    End If
Next
If tab(n - 1) Then
    Write("({0},{1}) + ({0},{2}) ." + vbCrLf, n - 1, i1, i2)
Else
    Write("-({0},{1}) + -({0},{2}) ." + vbCrLf, n - 1, i1, i2)
End If
Increment(tab)
Next
Next
Next
'Règle 3 - ligne (colonne : même code en inversant j et (i1 ou i2))
'Init3(tab)met les n/2+1 premières cases de tab sur True
'Increment3(tab) ajoute 1 à tab, sautant les cas où le nombre de case True
'n'est pas égal à n/2+1. ex: Increment(0111)=1011; Increment(1011)=1101
Dim tab(n - 1) As Boolean
For j = 0 To n - 1
    Init3(tab)
    While Increment3(tab)
        Dim compt = 0, k = 0
        While (compt < n / 2 + 1)
            If (tab(k)) Then
                If (compt <> n / 2) Then
                    Write("({0},{1}) + ", j, k)
                Else
                    Write("({0},{1}) . " + vbCrLf, j, k)
                End If
                compt += 1
            End If
            k += 1
        End While
        compt = 0
        k = 0
        While (compt < n / 2 + 1)
            If (tab(k)) Then
                If (compt <> n / 2) Then
                    Write("-({0},{1}) + ", j, k)
                Else
                    Write("-({0},{1}) . " + vbCrLf, j, k)
                End If
                compt += 1
            End If
            k += 1
        End While
    End While
Next

```

Format Dimacs n-sat

```
Public Function Createdimacs()
```

```

New StreamReader("Takuzu_Conjonctive.txt")
New StreamWriter = CreateTextFile("Takuzu_Dimacs.txt")

'Ecriture de l'en-tete
Dim m = 0
While Not EndOfStream
    c = Read()
    If (c = ".") Then
        m += 1
    End If
End While
WriteLine("p cnf {0} {1}", n * n, m)

'Ecriture du corps
'BaseN(a) convertit un entier a de la base 10 vers la base n
New StreamReader("Takuzu_Conjonctive.txt")
Dim a As String
While Not EndOfStream
    c = Read()
    If c >= "0" And c <= "9" Then
        a += c
    ElseIf c = "+" Then
        Write(BaseN(a) + " ")
        a = ""
    ElseIf c = "." Then
        Write(BaseN(a) + " 0" + vbCrLf)
        a = ""
    ElseIf c = "-" Then
        Write("-")
    End If
End While
Write(BaseN(a) + " 0")

```

Format 3-sat

```

Public Function Create3Sat()
    New StreamWriter = CreateTextFile("Takuzu_3Sat.txt")
    'Calcul du nouveau nombre de variable et de clause et entête
    New StreamReader("Takuzu_Dimacs.txt")
    Dim s = Split(ReadLine, " ")
    Dim nbvar = s(2)
    Dim nbclause = s(3)
    While Not EndOfStream
        s = Split(ReadLine, " ")
        Select Case (s.Length - 1)
            Case 0
                'Erreur
            Case 1
                nbvar += 2
                nbclause += 3

```

```

        Case 2
            nbvar += 1
            nbclause += 1
        Case 3
            'nothing
        Case Else
            nbvar += s.Length - 4
            nbclause += s.Length - 4
    End Select
End While
WriteLine("p cnf {0} {1}", nbvar, nbclause)

'Ecriture du corps
New StreamReader("Takuzu_Dimacs.txt")
Dim s = Split(ReadLine, " ")
Dim nbv = s(2)
While Not EndOfStream
    s = Split(ReadLine, " ")
    Select Case (s.Length - 1)
        Case 0
            'Erreur, ne pas recopier
        Case 1
            WriteLine("{0} {1} {2} 0", s(0), nbv + 1, nbv + 2)
            WriteLine("{0} -{1} {2} 0", s(0), nbv + 1, nbv + 2)
            WriteLine("{0} {1} -{2} 0", s(0), nbv + 1, nbv + 2)
            WriteLine("{0} -{1} -{2} 0", s(0), nbv + 1, nbv + 2)
            nbv += 2
        Case 2
            WriteLine("{0} {1} {2} 0", s(0), s(1), nbv + 1)
            WriteLine("{0} {1} -{2} 0", s(0), s(1), nbv + 1)
            nbv += 1
        Case 3
            WriteLine("{0} {1} {2} 0", s(0), s(1), s(2))
        Case Else
            WriteLine("{0} {1} {2} 0", s(0), s(1), nbv + 1)
            For i = 2 To (s.Length - 4)
                WriteLine("-{0} {1} {2} 0", nbv + 1, s(i), nbv + 2)
                nbv += 1
            Next
            WriteLine("-{0} {1} {2} 0", nbv+1, s(s.Length-3), s(s.Length-2))
            nbv += 1
    End Select
End While

```

Sat Solver

'La classe pile permet de sauvegarder. La fonction empile ajoute une sauvegarde de variables et f à la pile, dépile restaure variables et f de la sauvegarde
 'La fonction pos(a) renvoie True si a>0
 'La fonction RemoveAt(f,i) supprime la i^{ème} ligne du tableau f

```
Public Class Pile
```

```

Public val As Int16 = 0
Public savenbClause As Integer
Public b As Boolean = True
Public saveVar() As Int16
Public saveClause()() As Integer
Public suiv As Pile = Nothing

```

```

Public Function noeud(ByVal i As Int16, ByVal snbClause As Integer, ByVal
sVar() As Int16, ByVal sClause()() As Integer) As Pile
    Dim p = New Pile
    With p
        val = i
        saveVar = sVar
        saveClause = sClause
        savenbClause = snbClause
    End With
    Return p
End Function

```

```
End Class
```

```

Public variables() As Int16
Public f()() As Integer

```

```

Public Function ResolvePerso()
    'Lecture du fichier
    New StreamReader("Takuzu_3Sat.txt")
    Dim s = ReadLine().Split(" ")
    Dim nbVar = s(2)
    Dim nbClause = s(3)
    Dim f0 = ReadToEnd().Split(vbCrLf)
    ReDim variables(nbVar)
    For i = 0 To nbVar
        variables(i) = -1
    Next
    f = ConvertToIntArray(f0)
    'variables() contient donc mes variables en indices et -1 ou 0 ou 1 en valeur
    'f0 est un tableau de clause. J'ai converti chaque clause en tableau d'entier
    'dans ConvertToIntArray(f0).

```

```

    Dim resolution = True
    Dim save As Pile = New Pile

```

```
resolutionDPLL:
```

```

    While (nbClause > 0)
        Dim i, j As Integer
        While (resolution)
            'Clause unitaire -> Validation de la variable et retrait de la clause
            i = 0
            While (i < nbClause)

```



```

    If f(i).Length = 1 Then
        Dim a = Abs(f(i)(0))
        If variables(a) = -1 Then
            If f(i)(0) >= 0 Then
                variables(a) = 1
            Else
                variables(a) = 0
            End If
            RemoveAt(f, i)
            nbClause -= 1
            i -= 1
        ElseIf variables(a) = pos(f(i)(0)) Then
            RemoveAt(f, i)
            nbClause -= 1
            i -= 1
        Else
            GoTo unsatisfiable
        End If
    End If
    i += 1
End While
resolution = False

'Verification des clauses post validation unitaire
i = 0
While (i < nbClause)
    j = 0
    While (j < f(i).Length)
        If variables(Abs(f(i)(j))) <> -1 Then
            resolution = True
            If variables(Abs(f(i)(j))) = pos(f(i)(j)) Then
                RemoveAt(f, i)
                nbClause -= 1
                i -= 1
                Exit While
            Else
                RemoveAt(f(i), j)
                If f(i) Is Nothing Then
                    GoTo unsatisfiable
                End If
                j -= 1
            End If
        End If
        j += 1
    End While
    i += 1
End While
End While

```

'Plus de résolution possible, attribution arbitraire à Vrai

```

    For i = 1 To nbVar
        If variables(i) = -1 Then
            Empiler(i, save)
            variables(i) = 1
            resolution = True
            Exit For
        End If
    Next

End While

'Fin satisfaisable du DPLL
New StreamWriter = CreateTextFile("Takuzu_Solved_Dimacs.txt")
For j = 0 To nbVar
    If variables(j) = 1 Then
        Write(j + " ")
    ElseIf variables(j) = 0 Then
        Write("-" + j + " ")
    End If
Next
Write("0")
Exit Function

```

unsatisfiable:

```

    If (save.suiv Is Nothing) Then
        'Retour insatisfaisable
    Else
        'Dépiler comprend une attribution arbit. à Faux si la dernière était à vrai
        Dim a = Depiler(save)
        variables(a) = 0
        GoTo resolutionDPLL
    End If
End Function

```

Notre Groupe:

- François-Xavier Gros
- Aissame Boudaoudi
-