Le quadrille de Conway

Année 2017-2018

Elèves :
Remigy Lucile TSI, Cemeno Clémence TS, Piquemal Manon TS,
Papy Lisa 1SI, Triestram Lena 1SI, Voisembert Capucine 1SI, Pellon Adrien 1S, Vanderbeken Clémence 1S
Morel Clara 2de, Dedieu Maurine 2de, Hadot Melvin 1S, Larroui Timothey 1S, Mengelle Arnaud 1S, Dziki Yannis 1S, Starck Paul 1S .

Enseignants:
Lionel Marquet, Laure Villacampa, Guilhem Mazet, Christine Potier, Virginie Valette, Cécile Dubois

Etablissements :
Lycées *Jean-Pierre Vernant* de Pins Justaret et *Henri Matisse* de Cugnaux

Chercheur :
Olivier Mazet, INSA Toulouse

Sujet

Deux brins de cordes parallèles sont tenus à leur extrémité par quatre personnes.

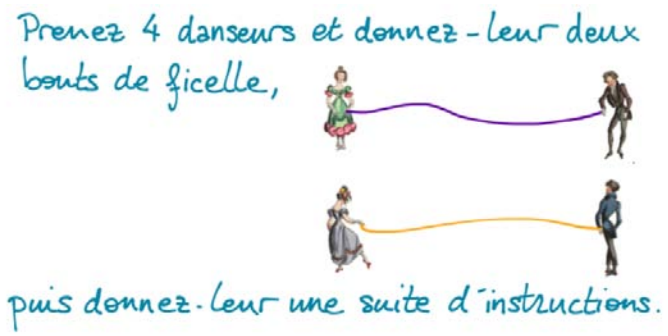
Etant donné une suite d'instructions (composée d'un certain nombre de « échanger » et « tourner »), qui aboutit à un nœud, peut-on déterminer une suite d'instructions (composée aussi d'un certain nombre de « échanger » et « tourner ») permettant de défaire le nœud et de revenir au point de départ ?

Résultats trouvés

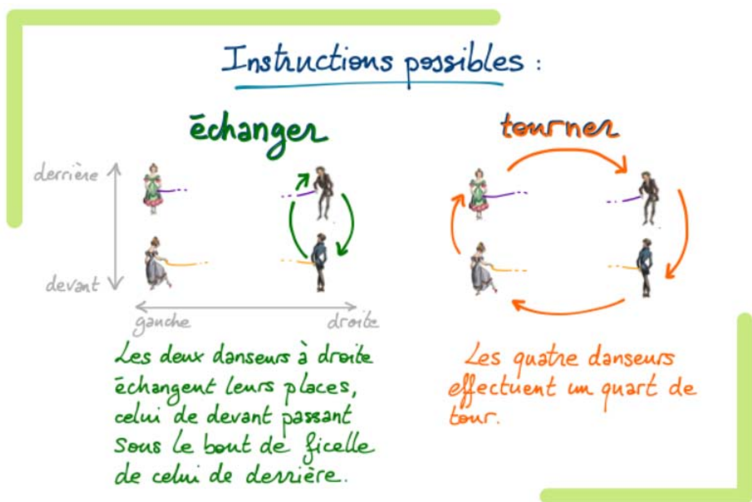
1- Etant donné un nœud, nous avons déterminé un protocole permettant de le défaire en utilisant les deux opérations « tourner » et « échanger »
2- Grâce à une représentation algébrique des opérations « tourner » et « échanger » nous avons démontré qu'il est possible de défaire n'importe quel nœud.
3- Nous avons créé un programme en Javaschool permettant de modéliser l'élaboration d'un nœud, puis sa résolution.

I- MISE EN SITUATION DU PROBLEME.

Au départ, 4 danseurs sont positionnés comme sur le dessin ci-dessous avec deux cordes. Le but est de créer un nœud avec deux instructions puis de le défaire avec ces deux mêmes instructions.



La première instruction est «Échanger» qui consiste à échanger la place des danseurs de telle sorte que celui de devant à droite passe sous le bout de la corde de celui de derrière à droite. Ce sont seulement les danseurs de droite qui échangent leur place entre eux.
«Echanger» sera notée «E» par la suite.
La deuxième instruction est «Tourner» qui consiste à effectuer un quart de tour dans le sens horaire.
«Tourner» sera notée «T» par la suite.



Etant donné un nœud fait, nous avons appelé « résoudre le nœud » l'action de défaire ce nœud pour revenir à la position initiale (deux cordes parallèles).

II- EXPERIMENTATION DU PROBLEME:

Nous avons commencé à expérimenter le problème en essayant de résoudre des actions basiques.
Ainsi pour résoudre un T, on applique 3T : « T T T »
En effet, de façon évidente, si l'on fait un quart de tour, puis ensuite trois quarts de tour, les danseurs reviennent à la position initiale.
Puis pour résoudre un E, on applique «T E T E T».

En effet, si l'on fait un « échanger », suivi de l'enchainement T E T E T, les danseurs reviennent à la position initiale.

Remarque :

Nous avons obtenu deux enchainements qui ne changent rien à une situation :

- T T T T
- E T E T E T

Si l'on procède à ces mouvements, quelle que soit la position des danseurs au départ, ils auront la même après avoir effectué ces enchainements.

Dans un deuxième temps, nous avons établi une méthode pour résoudre un nœud quelconque, c'est-à-dire résultant de plus de un seul mouvement.

Méthode de résolution d'un nœud :

Soit un nœud obtenu avec un enchainement de E et de T. Pour le résoudre il suffit de :

1) Inverser l'ordre des lettres E et T de l'enchainement
2) Remplacer chaque T par T T T et chaque E par T E T E T

L'enchainement de résolution était souvent très long. Nous avons alors ajouté une troisième instruction permettant de le raccourcir :

3) Simplifier l'enchainement de résolution en supprimant toutes les séquences égales à T T T T ou E T E T E T.

Exemple :

Soit le nœud obtenu avec l'enchainement E T E E

Pour le résoudre :

1) On inverse l'ordre : E E T E
2) On remplace : T E T E T T E T E T T T T T E T E T
3) On simplifie : T E T E T T E T E **T T T T** T E T E T

   T E T E T T E T E T E T E T

   T E T E T T E T **E T E T E T**

   T E T E T T E T

III-PASSAGE D'UN NŒUD A UN OBJET MATHEMATIQUE:

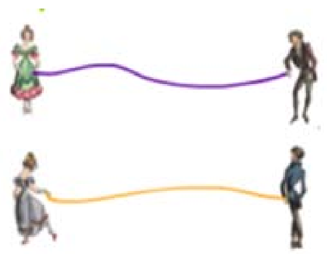
Quand nous avons eu terminé le protocole de résolution d'un nœud, le chercheur qui nous avait présenté le problème nous a proposé de passer d'une représentation du nœud par E et T, à une représentation algébrique.

Le modèle qu'il nous a donné est le suivant :

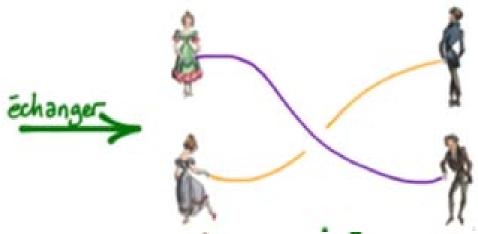
- un nœud est associé à un nombre
- la position de départ (cordes parallèles) est associée au nombre 0
- le mouvement T (tourner) correspond à la fonction $T : x \mapsto -\dfrac{1}{x}$
- le mouvement E (échanger) correspond à la fonction $E : x \mapsto x+1$ (1)

Exemple

Si l'on part de la position initiale (valeur du nœud : 0)



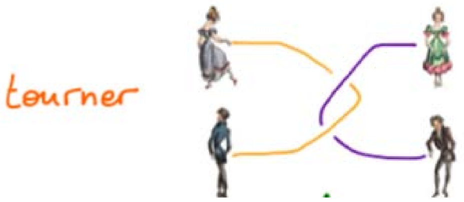
que l'on applique l'instruction E : $0 \mapsto 0+1=1$



puis encore l'instruction E : $1 \mapsto 1+1=2$

puis l'instruction T : $2 \mapsto -\frac{1}{2}$

On obtient alors un nœud qui est associé au nombre $-\frac{1}{2}$



Notre objectif est de passer du nœud correspondant à -1/2 à la situation initiale correspondant à 0 en utilisant les deux fonctions : E : $x \mapsto x+1$ et T : $x \mapsto -\frac{1}{x}$

Pour résoudre cela, on récupère la fraction du nœud précédent : -1/2

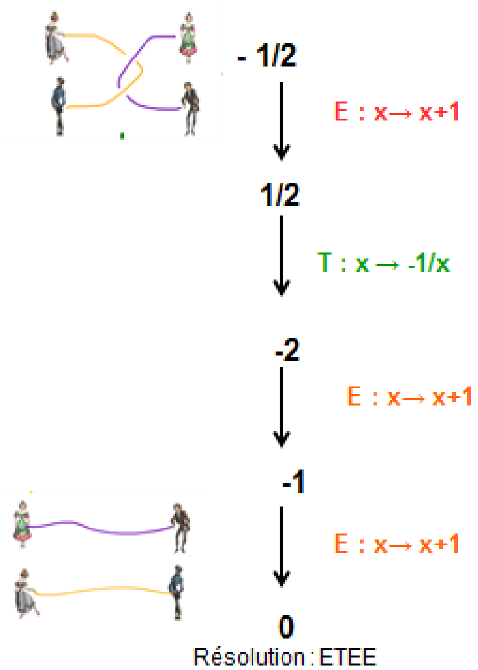
Ensuite, on applique la fonction E : x↦ x+1. On obtient 1/2

Puis on applique la fonction T : x↦ -1/x on obtient -2

(notre but est d'atteindre 0 donc cela ne sert à rien de continuer les x↦ -1/x).

Enfin, on applique E : $x \mapsto x+1$ deux fois, on obtient alors 0.

Ainsi, la succession de fonctions appliquées au nombre -1/2 correspond à la résolution : ETEE.



(2)

Nous nous sommes alors demandé si, pour chaque nombre correspondant à un nœud, il était possible, par un enchainement des deux fonctions, de revenir à zéro.

Tout d'abord, pour faire un nœud on part de 0 puis applique seulement les fonctions E : $x \mapsto x+1$ et T : $x \mapsto -\frac{1}{x}$, et ceci un nombre fini de fois. Donc, le nombre correspondant à un nœud est forcément une fraction.

Théorème

Étant donné une fraction, il est possible, en appliquant un nombre fini de fois les fonctions E : $x \mapsto x+1$ et T : $x \mapsto -\frac{1}{x}$ de revenir à 0.

Démonstration

Si la fraction n'est pas négative on peut se ramener à une fraction négative à l'aide de la fonction T : $x \mapsto -\frac{1}{x}$.

Soit donc une fraction négative : $-\frac{a}{b}$ avec $a \in \mathbb{N}$ et $b \in \mathbb{N}^*$.

→ avec la division euclidienne de a par b, on peut écrire : $-\frac{a}{b} = \frac{-nb+r}{b}$ avec $n, r \in \mathbb{N}$ et $0 \leq r < b$

→ on applique n fois la fonction E : $x \to x+1$ au nombre $-\frac{a}{b} = \frac{-nb+r}{b} = -n + \frac{r}{b}$

→ on obtient alors une nouvelle fraction : $\frac{r}{b} > 0$ (3)

→ on applique à $\frac{r}{b}$ la fonction T : $x \mapsto -\frac{1}{x}$

→ on obtient la fraction : $-\frac{b}{r}$ avec $0 \leq r < b$ (4)

On a donc une nouvelle fraction dont le dénominateur r, est un entier strictement plus petit que b le dénominateur de la fraction de départ.

Comme il y a un nombre fini de nombres entiers entre 1 et b, en répétant un nombre fini de fois ce processus, on obtiendra une fraction de dénominateur égal à 1 : $-\frac{c}{1} = -c$ où $c \in \mathbb{N}$. (5)

Il suffit alors d'appliquer c fois la fonction E : $x \mapsto x+1$ pour obtenir 0.

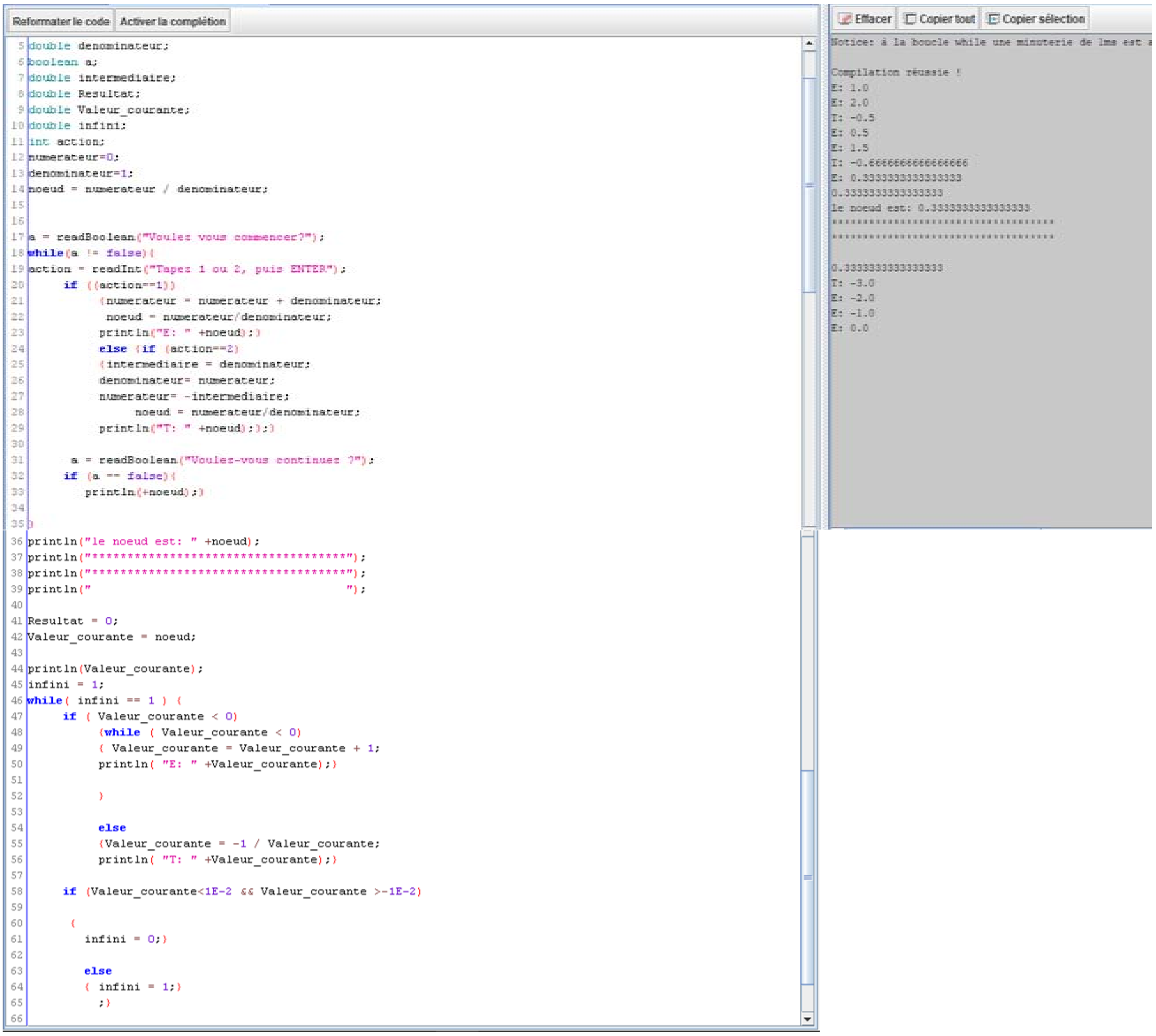
On a bien réussi, en un nombre fini d'étapes, à ramener une fraction quelconque à 0, le théorème est démontré.

III-PROGRAMME DE RESOLUTION D'UN NOEUD

En suivant le protocole utilisé dans la démonstration, nous avons écrit un programme en JavaSchool qui permet :

- en entrant une suite d'instructions de type T et E de déterminer la fraction associée au nœud obtenu ;
- étant donné un nœud (une fraction), de déterminer la suite d'instructions permettant de le résoudre.

La principale difficulté rencontrée dans l'écriture du programme est que JavaSchool ne prend pas les fractions : ce langage de programmation travaille uniquement avec des écritures décimales de nombres, et pour les fractions qui n'ont pas d'écriture décimale, c'est une valeur approchée qui est utilisée.
Donc, lorsqu'il y avait une suite d'instructions, les arrondis étaient de moins en moins précis, et pour la résolution on n'arrivait jamais à zéro : le programme tournait sans jamais s'arrêter.
Nous avons donc modifié l'instruction d'arrêt : au lieu de s'arrêter à 0, la dernière boucle s'arrête lorsque le nombre est compris entre -10^{-2} et 10^{-2}. (6)




```
double denominateur;
boolean a;
double intermediaire;
double Resultat;
double Valeur_courante;
double infini;
int action;
numerateur=0;
denominateur=1;
noeud = numerateur / denominateur;


a = readBoolean("Voulez vous commencer?");
while(a != false){
action = readInt("Tapez 1 ou 2, puis ENTER");
      if ((action==1))
           {numerateur = numerateur + denominateur;
            noeud = numerateur/denominateur;
           println("E: " +noeud);}
           else {if (action==2)
           {intermediaire = denominateur;
           denominateur= numerateur;
           numerateur= -intermediaire;
                noeud = numerateur/denominateur;
           println("T: " +noeud);};}

       a = readBoolean("Voulez-vous continuez ?");
      if (a == false){
         println(+noeud);}

}
println("le noeud est: " +noeud);
println("************************************");
println("************************************");
println("                                    ");

Resultat = 0;
Valeur_courante = noeud;

println(Valeur_courante);
infini = 1;
while( infini == 1 ) {
      if ( Valeur_courante < 0)
           (while ( Valeur_courante < 0)
           { Valeur_courante = Valeur_courante + 1;
           println( "E: " +Valeur_courante);}

           }

           else
           (Valeur_courante = -1 / Valeur_courante;
           println( "T: " +Valeur_courante);}

      if (Valeur_courante<1E-2 && Valeur_courante >-1E-2)

       {
         infini = 0;}

         else
         { infini = 1;}
           ;}

```

```
Notice: à la boucle while une minuterie de 1ms est a

Compilation réussie !
E: 1.0
E: 2.0
T: -0.5
E: 0.5
E: 1.5
T: -0.6666666666666666
E: 0.3333333333333333
0.3333333333333333
le noeud est: 0.3333333333333333
************************************
************************************

0.3333333333333333
T: -3.0
E: -2.0
E: -1.0
E: 0.0
```

Texte du Programme

```
void main(){
double noeud;
double numerateur;
double denominateur;
boolean a;
double intermediaire;
double Resultat;
double Valeur_courante;
double infini;
int action;
numerateur=0;
denominateur=1;
noeud = numerateur / denominateur;


a = readBoolean("Voulez vous commencer?");
while(a != false){
action = readInt("Tapez 1 ou 2, puis ENTER");
          if ((action==1))
              {numerateur = numerateur + denominateur;
               noeud = numerateur/denominateur;
              println("E: " +noeud);}
                    else {if (action==2)
                    {intermediaire = denominateur;
                    denominateur= numerateur;
              numerateur= -intermediaire;
                                noeud =
numerateur/denominateur;
                    println("T: " +noeud);};}

          a = readBoolean("Voulez-vous continuer ?");
          if (a == false){
            println(+noeud);} }
```

```
println("le noeud est: " +noeud);
println("***********************************");
println("***********************************");
println("                              ");


Resultat = 0;
Valeur_courante = noeud;

println(Valeur_courante);
infini = 1;
while( infini == 1 ) {
          if ( Valeur_courante < 0)
        {while ( Valeur_courante < 0)
                    { Valeur_courante =
Valeur_courante + 1;
                    println( "E: " +Valeur_courante);}}
                    else
                    {Valeur_courante = -1 /
Valeur_courante;
                    println( "T: " +Valeur_courante);}

          if (Valeur_courante<1E-2 &&
Valeur_courante >-1E-2)

           {
             infini = 0;}

             else
             { infini = 1;}
                    ;}

}
```

Notes d'édition

(1) On peut vérifier que les fonctions proposées dans ce modèle vérifient bien TTTT=Identité (c'est-à-dire que T(T(T(T(x)))) = x pour tout réel x non nul) et ETETET=Identité.
Cependant on a également TT=Identité. Si l'on revient au problème des danseurs, cela veut dire qu'on considère qu'après avoir effectué deux fois T, le nœud est résolu. Or après les opérations TT, les deux cordes sont bien parallèles horizontales, mais les danseurs ne sont pas au même endroit : les filles sont à droite et les garçons à gauche, et la corde du haut est passée en bas. Ce n'est donc pas exactement le même problème qu'on se pose avec cette modélisation.
D'autre part, la fonction correspondant à T n'est pas définie pour x=0. Donc cette modélisation ne peut rigoureusement décrire que des transformations qui ne commencent pas par effectuer T. On pourrait surmonter ce problème, mais cela compliquerait la rédaction qui suit.

(2) Lorsqu'on effectue la suite d'opérations décrite E puis E puis T puis E puis T puis E puis E, on n'obtient pas la configuration initiale redessinée en bas de la page 4, mais une configuration avec une femme et un homme à gauche, la femme est en bas mais c'est celle qui à l'origine était en haut. Par contre, les cordes sont bien horizontales et complètement dénouées.
Ici on trouve donc que la résolution de EET est ETEE, alors que dans la première partie on avait trouvé que la résolution de ETEE était TETETTET ; cela ne veut pas dire que EET = TETETTET car ici les danseurs ne reviennent pas exactement au même endroit.

(3) Si r/b=0, on a fini. On peut donc regarder ce qui se passe si r/b>0.

(4) En fait ici r>0 puisque si r=0 on avait terminé (pas de division par 0 à faire!)

(5) Ou alors on aura obtenu r=0 avant, et on aura également terminé.

(6) Il est donc possible qu'avec des suites d'opérations longues, le programme Javaschool ne donne pas le bon résultat. Mais il existe des langages permettant de travailler avec des rationnels.