Rapport Machine de Turing

## IUT de Vannes

A l’attention de M.Fleurquin



Table des matières

[IUT de Vannes 1](#_Toc412971728)

[Architecture de l’application 3](#_Toc412971729)

[Choix d’implantation 4](#_Toc412971730)

[Le ruban 4](#_Toc412971731)

[La fonction de transition 4](#_Toc412971732)

[Le format des fichiers 4](#_Toc412971733)

[La gestion du pas à pas 4](#_Toc412971734)

[La gestion état par état 4](#_Toc412971735)

[La gestion de l’arrêt en cours d’éxecution 4](#_Toc412971736)

[Codes sources 5](#_Toc412971737)

[Tests appliqués 6](#_Toc412971738)

[Copies d’écran 7](#_Toc412971739)

# Architecture de l’application

/src

|-- **ctrl**

| `-- TMCtrl

| `-- TMListener

|-- **data**

| `-- Machine

| `-- Transition

| `-- TuringIO

| `-- TuringSyntaxe

|-- **view**

| `-- TMView

| `-- Tape

| `-- TransitionTableModel

|-- **test**

| `-- TestConfigurations

Pour effectuer cette application, nous avons choisi d’utiliser le modèle MVC.

## La partie « data »

Le package ‘data’ représente le fonctionnement interne d’une machine de Turing. Les ‘*Transitions’* étant le cœur de la machine, une classe entière y fut dédiée.

La ‘*Machine’* contient toutes les données entrées dans le fichier de configuration.

Les fichiers ‘*TuringIO’* et ‘*TuringSyntaxe’* permettent la lecture de ce fichier et d’en vérifier la conformité, selon les critères syntaxiques prédéfinis.

## La partie « view »

Ce package constitue l’interface visuelle de l’application. ‘*TMView’* représente l’interface générale de la fenêtre : boutons, listes, menus, etc.

‘*Tape’* est une classe dédiée à la représentation du ruban de la machine. Ce ruban étant dynamique (affichage de la tête de lecture, modification des caractères), il nécessitait un fichier à part entière.

La classe *‘TransitionTableModel’* est simplement utilisée pour personnaliser l’affichage du tableau représentant la liste des transitions de la machine.

## La partie « ctrl »

Cette partie est divisée en deux classes :

* ‘*TMListener*’, implémentant toutes les actions liées à l’interface graphique.
* *‘TMCtrl’*, permettant les différents lancements de la machine.

Le contrôle de la machine récupère les données depuis *‘Machine’* et effectue le lancement du programme tout en mettant à jour l’interface. Il initialise, met à jour, et remet à zéro la machine.

Il constitue donc le lien entre les données et l’interface.

# Choix d’implantation

## Le ruban

Le ruban est affiché sur un panel dédié. C’est une liste de JLabel, dessinée sous forme de cases (grâce aux bordures). Le ruban étant normalement infinie sur la droite, nous avons décidé de limiter les cases vides sur la droite, une implémentation illimitée n’étant pas optimisée. Le nombre de cases vides sur la droite est par défaut de 50, mais il est possible de le modifier dans le menu paramètres.

Chaque case est numérotée à partir de 1 afin de mieux se repérer sur le ruban, surtout lors du scroll automatique du ruban. En effet, l’interface reste fixée sur la tête de lecture, afin de ne pas la perdre de vue elle dépasse les cases initialement affichées.

La tête de lecture est colorée en jaune, par défaut, afin de la repérer (La couleur est modifiable dans les paramètres).

## La fonction de transition

Les transitions sont effectuées par la classe *‘TMCtrl’*. Une simple méthode ‘doTransition()’ lance la prochaine transition, selon l’état courant et le symbole lu sur la tête de lecture. Elle modifie ainsi les données courantes du programme, et l’interface graphique.

## Le format des fichiers

### Configurations

### Résultat

Les résultats sont affichés dans un fichier *Configurations.txt*, créé par défaut dans le répertoire parent de l’application (Le répertoire de sauvegarde est configurable dans les paramètres). Il inscrit, ligne par ligne, l’ensemble de toutes les configurations par laquelle la machine est passée.

Configurations du programme FoncTrans.txt :

q0 0 1 \_

b q2 1 \_

b 1 q2 \_

b 1 # q5 \_

b 1 q6 # 0 \_

b q6 1 # 0 \_

q6 b 1 # 0 \_

0 q7 1 # 0 \_

0 a q3 # 0 \_

0 a # q3 0 \_

0 a # 0 q3 \_

0 a # q4 0 1 \_

0 a q4 # 0 1 \_

0 q4 a # 0 1 \_

0 1 q7 # 0 1 \_

Une ArrayList<String> de configuration est formée pendant l’exécution du programme. Celle-ci est alors écrite dans le fichier à la fin de l’exécution complète (arrivée dans un état acceptant ou rejetant).

Elles sont écrites de la forme suivante :

[Mot à gauche de la tête] [État courant] [Symbole courant + mot à droite de la tête]

## La gestion état par état

Lors du clic sur le bouton « état par état », le programme effectue alors une seule transition (exécution de la méthode ‘doTransition()’).

## La gestion étape par étape

Dans le fichier de configuration, des états peuvent être marqués comme étant des états « stopant ». Ces états, lors d’un lancement de la machine en mode « Etape par étape » arrêtent l’exécution du programme.

## La gestion de l’arrêt en cours d’exécution

# Codes sources

# Tests appliqués

# Copies d’écran