`turing (v. 0.4)`
Une simulation des machines de Turing avec TEX

Jean-Côme Charpentier
email : `Jean-Come.Charpentier@wanadaoo.fr`

3 février 2003

1 Présentation de la machine de Turing

1.1 Un peu d'histoire

La machine de Turing a été conçu par ... Turing (Alan de son petit nom). Cette machine, à l'origine, était uniquement une machine théorique puisque les ordinateurs qui auraient pu la simuler n'existaient pas encore. En effet, la machine de Turing a été imaginée dans le cadre d'un travail sur une définition précise de la notion de preuve formelle au cours des années 1930 (en 1936 pour la machine de Turing). On peut qualifier d'apothéose de la machine de Turing la thèse de Turing-Church qui peut s'énoncer comme suit :

Toute fonction calculable l'est par une machine de Turing

Comme un programme informatique peut être considéré comme une fonction calculable, il s'en suit que tout programme informatique peut être effectuée sur une machine de Turing. En ce sens, la machine de Turing est universelle.

Avant d'en terminer avec les généralités, le lecteur pourrait se demander pourquoi on n'utilise pas de machines de Turing au lieu d'ordinateurs réels. Il y a deux raisons principales à cela :

- la machine de Turing est une machine idéale totalement impossible à obtenir (nous allons voir qu'elle fait appel à une bande d'écriture de longueur infinie ;
- même en prenant une machine de Turing finie, celle-ci se révèle désespéremment lente dans son fonctionnement.

1.2 Anatomie et fonctionnement

Une machine de Turing est assez simple à décrire. Il s'agit d'un mécanisme comportant une bande (mettons de papier) de longueur infinie dans les deux sens, sur laquelle une tête de lecture/écriture se déplace. La bande est découpée en cellules.

Les symboles qui peuvent se trouver sur la bande (un symbole par cellule) forme l'alphabet de la machine de Turing. Normalement, cet alphabet est quelconque mais on peut montrer qu'on peut toujours se ramener à un alphabet composé de deux symboles : ce sera le choix de l'extension `turing`.

A tout instant, une machine de Turing se trouve dans un état déterminé. On notera cet état avec une suite de caractères (par exemple, l'état `i` ou bien l'état `A12`). Parmi ces états, deux vont avoir un rôle particulier : l'état initial qui est l'état de la machine de Turing avant que ses calculs ne commencent et l'état terminal qui est l'état qui indique que les calculs sont terminés[1].

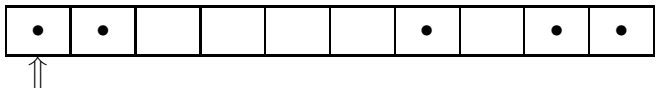
Enfin, une machine de Turing va évoluer selon des règles précises. Pour chaque état, et selon le symbole actuellement au niveau de la tête de lecture/écriture, la machine de Turing écrira un nouveau symbole à cet emplacement, changera d'état et déplacera la tête d'une cellule vers la gauche ou vers la droite. L'ensemble des règles consistera donc à préciser quoi faire lorsque la machine est dans un certain état en train de lire un certain symbole. Le « quoi faire » étant quel symbole écrire, dans quel état devra se trouver la machine après cette écriture et dans quel sens devra se déplacer la tête de lecture.

[1] Là encore, une machine de Turing peut avoir plusieurs états terminaux possibles. On ne perd aucune généralité à décréter qu'il n'y en a qu'un seul. L'extension `turing` ne proposera qu'un seul état terminal.

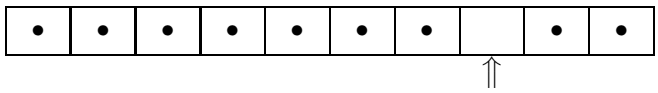
Un petit exemple pour fixer les idées. Supposons que l'état initial soit `i`, que l'état final soit `t`, qu'il n'y ait que les deux symboles • et ⟨vide⟩. Les règles sont décrites par le tableau ci-dessous :

État courant	Symbole lu	État de sortie	Symbole écrit	Déplacement
`i`	•	`a`	•	`r`
`i`	⟨vide⟩	`a`	•	`r`
`a`	•	`t`	•	`r`
`a`	⟨vide⟩	`a`	•	`r`

Ce « programme » va laisser tranquille la première séquence de puces en la parcourant de gauche à droite et, dès qu'une cellule vide est rencontrée, on y écrit des puces. Le programme s'arrête lorqu'une nouvelle série de puces débute. Voici l'illustration de la chose. Supposons qu'on ait la bande initiale :



où la flèche verticale représente la tête de lecture-écriture. Après l'exécution complète de ce programme on aura la situation suivante :



2 L'extension turing

2.1 Préparation du terrain

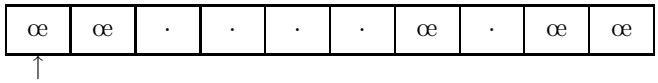
Dans sa version actuelle, l'extension `turing` ne propose que les fonctions les plus importantes pour gérer une machine de Turing. Le codage a été effectué pour pouvoir servir aussi bien sous LaTeX que sous TeX. En revanche, bien qu'un certain nombre de tests ait été réalisés, il est tout à fait possible que des effets de bords nuisibles aient lieu sous certaines conditions d'utilisation. L'utilisateur scrupuleux n'hésitera pas à en faire part à l'auteur !

Comme il a été dit précédemment, les machines de Turing créées avec cette extension ne comporteront que deux symboles possibles, un seul état initial et un seul état terminal (on dit également état accepteur). Évidemment, la bande sera finie mais elle sera capable de s'agrandir automatiquement lorsqu'il le faudra.

Commençons par les symboles. Le premier symbole sera toujours spécifié avec le caractères `*` par l'utilisateur mais sa présentation au niveau du document pourra être modifiée. Par défaut, il sagit du symbole •. Ce symbole est en fait stocké dans la macro `\Tmark` que l'on pourra modifier à volonté. Le second symbole disponible sera spécifié avec n'importe quel caractère (hormis `*`[2]). Au niveau du document, sa réprésentation est vide mais là encore, on peut modifier cette présentation par défaut en redéfinissant la macro `\Tempty`. Enfin, le symbole de la tête de lecture/écriture et lui aussi modifiable via la macro `\Thead`. Par exemple, avec la spécification suivante :

```
\def\Tmark{\oe}
\def\Tempty{.}
\def\Thead{$\uparrow$}
```

le premier exemple de bande deviendrait :

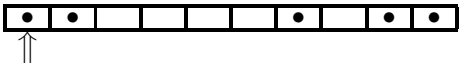


Pour terminer avec ces histoires de présentation, les dimensions des cellules ainsi que l'épaisseur des traits sont paramètrables avec les dimensions `\Twidth`, `\Theight` et `\Tthick`. Par exemple, avec la spécification suivante :

```
\Twidth=16pt % syntaxe TeX
\setlength{\Theight}{8pt} % syntaxe LaTeX
\Tthick=1pt
```

[2] Il faut également éviter les caractères actifs. Ainsi, avec les extensions de francisation, les caractères « : », « ; », « ! » et « ? » deviennent interdits

le premier exemple de bande deviendrait :



Lorsqu'on veut gérer une machine de Turing, les deux premières étapes consistent à indiquer les symboles écrits sur la bande et à donner la table de transitions (le tableau de la page précédente).

La liste des symboles sur la bande est spécifiée en utilisant la macro `\TuringInit` avec comme paramètre, une suite de caractères, les `*` de cette chaîne de caractères indiquant les emplacements des symboles •.

La table de transition est donnée par la macro `TuringTabState` avec, comme paramètre, une suite de quintuplets de la forme : `<E_entrée>,<S_entrée>,<E_sortie>,<S_sortie>,<dir>` où `<E_entrée>` indique l'état de départ et `<S_entrée>` le symbole actuellement au niveau de la tête de lecture/écriture. Les trois autres données indiquent ce que dois faire la machine de Turing, à savoir se mettre dans l'état `<E_sortie>` après avoir écrit le symbole `<S_sortie>` puis déplacer sa tête de lecture/écriture dans la direction `<dir>`. La direction sera soit `r` ou `R` pour un déplacement vers la droite ou bien `l` ou `L` pour un déplacement vers la gauche. Chaque quintuplet sera séparé du suivant par une virgule[3]. Il faudra faire attention à ne pas mettre d'espace parasite au niveau de ce paramètre[4]. Une présentation permettant d'éviter les oublis peut être quelque chose du style :

```
\TuringInit{**....*.**}%
\TuringTabState{i,*,i,*,r,%
                i,.,a,*,r,%
                a,*,t,*,r,%
                a,.,a,*,r}%
```

C'est ce qui a été tapé dans ce document pour obtenir le premier exemple de programme.

Pour visualiser la bande de la machine de Turing, il suffit de taper `\TuringPrint`. Par défaut, cette macro ne crée pas de paragraphe et ne centre pas la bande. Si on veut obtenir une présentation hors-texte, on peut utiliser la macro `\TuringCenterPrint`. Celle-ci réalise une présentation centrée en plaçant les espaces verticaux définis par les dimensions `\Turingbeforeskip` et `\Turingafterskip` (par défaut, ces deux valeurs sont fixées à `\smallskip`).

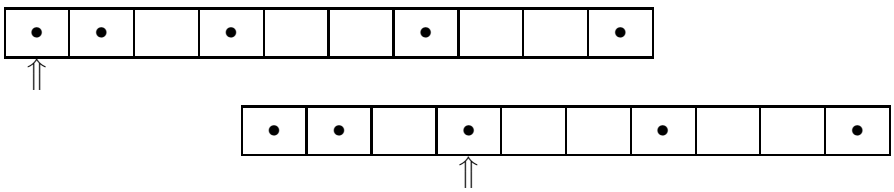
Enfin, la macro `\TuringMoveIO` permet de positionner la tête de lecture/écriture à n'importe quel endroit de la bande. Pour cela, il suffit de passer en paramètre l'indice de la cellule sur laquelle on veut que soit positionnée la tête de lecture/écriture. Lorsque la bande est crée avec `\TuringInit`, la cellule de gauche est la cellule numéro 0 et la tête de lecture/écriture est positionnée à ce niveau, mais au cours des calculs, en cas d'allongement de la bande, cet indice peut devenir négatif.

Voici un exemple complet pour montrer l'action des différentes macros :

```
\TuringInit{**.*..*mk*}

\TuringPrint
\TuringMoveIO{3}
\TuringCenterPrint
```

va donner :



[3]Tout ceci rend l'écriture de la teble de transitions un peu pénible. Une prochaine version de `turing` devrait améliorer ce point.

[4]Une prochaine version devrait permettre l'inclusion d'espace sans créer de problème.

2.2 Calculs

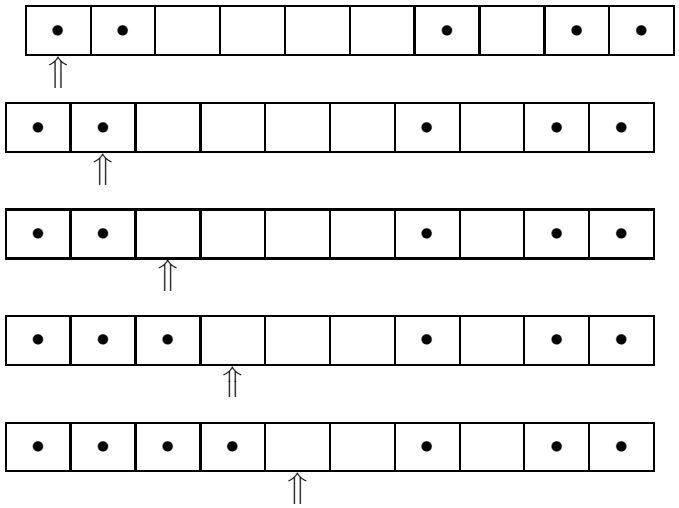
L'extension `turing` met deux macros à disposition pour simuler le fonctionnement d'une machine de Turing : `\TuringStep` qui effectue un seul pas de calcul et `TuringRun` qui lance la machine jusqu'à ce qu'elle ait atteint son état terminal.

Lorsqu'on crée une machine avec les macros `\TuringInit` et `\TuringTabState`, elle se met automatiquement dans l'état initial. Cet état est l'état `i` par défaut. Cette valeur par défaut peut être modifiée en redéfinissant la macro `\Tinit`. De même, il existe une valeur par défaut pour l'état final qui est l'état `t` et qu'on peut modifier en redéfinissant la macro `Tterm`. Si on décide de modifier l'état initial, il faut obligatoirement le faire avant d'appeller la macro `\TuringInit`.

La macro `\TuringStep` permet d'effectuer un seul pas de calcul. Si la machine est dans l'état terminal, rien ne se passera et si la machine se trouve dans un état e avec un symbole s au niveau de la tête de lecture/écriture mais que la table de transition n'a pas prévu ce cas de figure, un message d'avertissement sera affiché sur le terminal et aucune action ne sera effectuée[5]. En reprenant l'exemple donné au début de cette documentation, en tapant :

```
\TuringInit{**....*.**}
\TuringCenterPrint
\TuringTabState{i,*,i,*,r,%
                i,.,a,*,r,%
                a,*,t,*,r,%
                a,.,a,*,r}
\TuringStep
\TuringCenterPrint
\TuringStep
\TuringCenterPrint
\TuringStep
\TuringCenterPrint
\TuringStep
\TuringCenterPrint
```

on obtient l'affichage suivant :

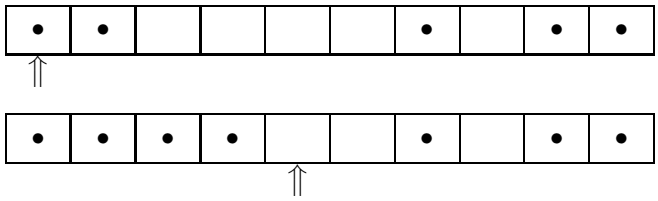


La macro `\TuringRun` permet d'aller directement à la fin du calcul. Évidemment, si cette fin n'existe pas, la compilation va boucler indéfiniment. C'est donc à l'utilisateur de faire un peu attention à ce qu'il fait ! La macro `\TuringRun` peut être spécifiée avec un argument optionnel indiquant le nombre de pas de calcul maximum qui seront fait avant d'effectuer une pause. Ainsi, on aurait pu obtenir le dernier état de l'exemple précédent de façon direct avec le code :

[5] Il est possible que dans une version ultérieure, on laisse le choix entre un message d'avertissement et un message d'erreur.

```
\TuringInit{**....*.**}
\TuringCenterPrint
\TuringTabState{i,*,i,*,r,%
                i,.,a,*,r,%
                a,*,t,*,r,%
                a,.,a,*,r}
\TuringRun[4]
\TuringCenterPrint
```

qui donnera :



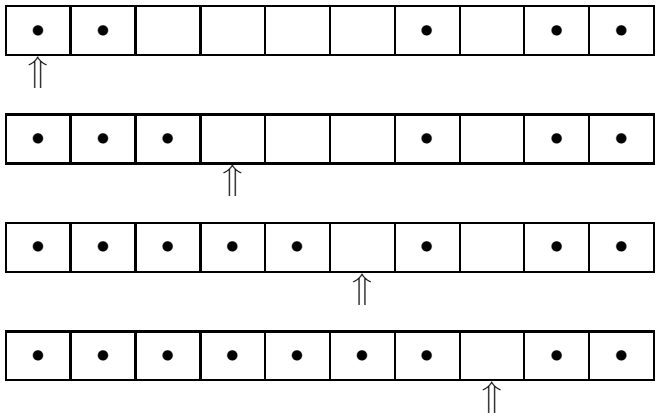
Il est possible de commencer un calcul avec la machine dans un état autre que l'état initial. Pour cela, il suffit de redéfinir la macro `\Tstate` après l'appel à la macro `\TuringInit`.

Attention : `\TuringRun` procède à des initialisations. Cette macro est sensée *commencer* un calcul. Par conséquent, si on utilise cette macro avec un argument optionnel indiquant un nombre d'étapes maximum (syntaxe `\TuringRun[<max>]`), la reprise du calcul ne devra pas se faire avec `\TuringRun`. Pour reprendre un calcul interrompu, la macro à utiliser est `\TuringResume`. Cette dernière fonctionne comme `\TuringRun`, en particulier, elle admet également un paramètre optionnel indiquant un nombre maximum d'itérations.

En, reprenant l'exemple précédant, cela pourrait donner :

```
\TuringInit{**....*.**}
\TuringCenterPrint
\TuringTabState{i,*,i,*,r,%
                i,.,a,*,r,%
                a,*,t,*,r,%
                a,.,a,*,r}
\TuringRun[3]
\TuringCenterPrint
\TuringResume[2]
\TuringCenterPrint
\TuringResume
\TuringCenterPrint
```

qui donnera :

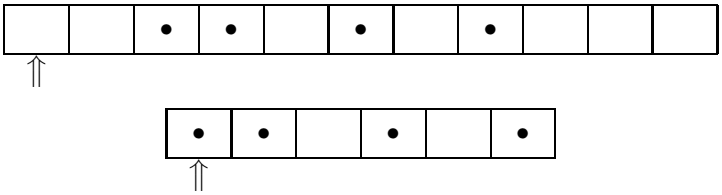


2.3 Améliorer l'affichage

La macro `\TuringFormat` supprime les séquences de cellules vides aux extrémités de la bande. Si la tête de lecture/écriture se trouve dans une zone supprimée, elle est déplacée sur la première cellule de la bande restante, sinon, elle reste sur la même cellule[6]. Voici un petit exemple illustratif :

```
\TuringInit{..**.*.*...}%
\TuringCenterPrint
\TuringFormat
\TuringCenterPrint
```

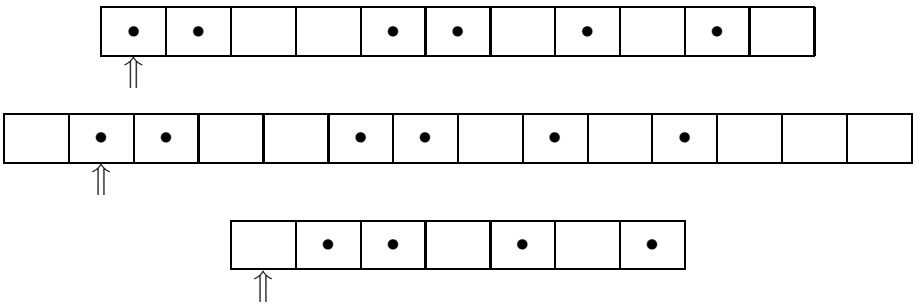
donne :



La macro `\TuringEnlarge` permet d'agrandir la bande avec des cellules vides à droite et à gauche de la bande. Cette macro demande deux paramètres qui seront respectivement le nombre de cellules à ajouter à gauche et le nombre de cellules à ajouter à droite. Ces paramètres peuvent être des nombres négatifs auquel cas, la bande sera diminuée au lieu d'être augmentée. Voici un exemple :

```
\TuringInit{**..**.*.*.}%
\TuringCenterPrint
\TuringEnlarge{1}{2}%
\TuringCenterPrint
\TuringEnlarge{-4}{-3}%
\TuringCenterPrint
```

donne :



3 Liste des choses à faire

La spécification de la table de transitions sans avoir le droit de mettre des espaces est un peu pénible. Il faudrait donc les permettre sans que cela interfère sur la lecture des arguments.

Lorsqu'une bande est fabriquée, elle l'est dans une boîte horizontale. Cela signifie que si la bande est plus large qu'une ligne, elle sera composée dans la marge du document (pas terrible). Il faudrait donc un mécanisme automatique de découpage de la bande en cas de dépassement. Le mieux serait de distinguer si la compilation se fait sous TEX ou sous LATEX pour tester la longueur du corps de texte respectivement avec `\hsize` ou `\columnwidth` (en cas de composition sur plusieurs colonnes).

La spécification de la bande d'origine et de la table de transition est assez lourde, il faudrait prévoir la possibilité de lecture de ces données sur un fichier annexe par un `\TuringRead{nom\_fichier}`.

Il y a sans doute d'autres points qui peuvent être améliorés. Les critiques, suggestions, améliorations de toutes sortes seront hautement appréciées.

[6] Ce n'est peut être pas une bonne idée. Une autre possibilité serait d'interdire la suppression au-delà de la tête de lecture/écriture.