3.2.3 M_1 : mots jumeaux (détail)

On a déjà vu une ébauche de l'algorithme plus haut. Regardons en détail comment on peut réaliser cet algorithme.

Cœur de l'algorithme L'idée est que l'on va "zigzaguer" de part et d'autre du # en comparant un à un les symboles. En s'inspirant de l'exemple précédent, on voit comment on pourrait procéder : on peut barrer d'un x les caractères déjà vérifiés, marquer le début du mot par un blanc, mais il reste une question : comment réaliser le "test" ? Comment exprimer dans notre "langage de programmation" l'idée que l'on compare deux caractères ?

La réponse vient de la notion d'état, et plus généralement de chemin dans le graphe : il suffit que l'on soit sur des chemins différents lorsqu'on veut trouver un 1 et lorsqu'on veut trouver un 0. C'est la raison pour laquelle le diagramme final, proposé à la figure 3.2, présente une structure en fourche (il n'y a que deux branches car il n'y a que 2 symboles différents possibles).

Ainsi le cœur de l'algorithme correspond aux états de 8 à 13 dans le diagramme. Mais, tel qu'il est conçu, cet algorithme fait un certain nombre d'hypothèses sur le mot en entrée : on suppose qu'il contient bien un #, et une suite de 0 et de 1 de part et d'autre du dièze. Si cette hypothèse n'est pas vérifiée, on peut conclure tout de suite au rejet, sans poursuivre le calcul.

C'est la raison pour laquelle l'algorithme est construit avec une partie (états 3,5,7 et états 2,4,6) chargée de vérifier que le mot d'entrée est bien de la forme $u\#v$, avec u et $v \in \{0,1\}^*$.

Ces explications devraient suffire pour interpréter le diagramme de la MT proposée, en notant que ce diagramme n'est pas complet, au sens où n'y figurent pas les transitions vers l'état de rejet. Ces transitions correspondent à tous les cas non explicitement envisagés dans ce diagramme.

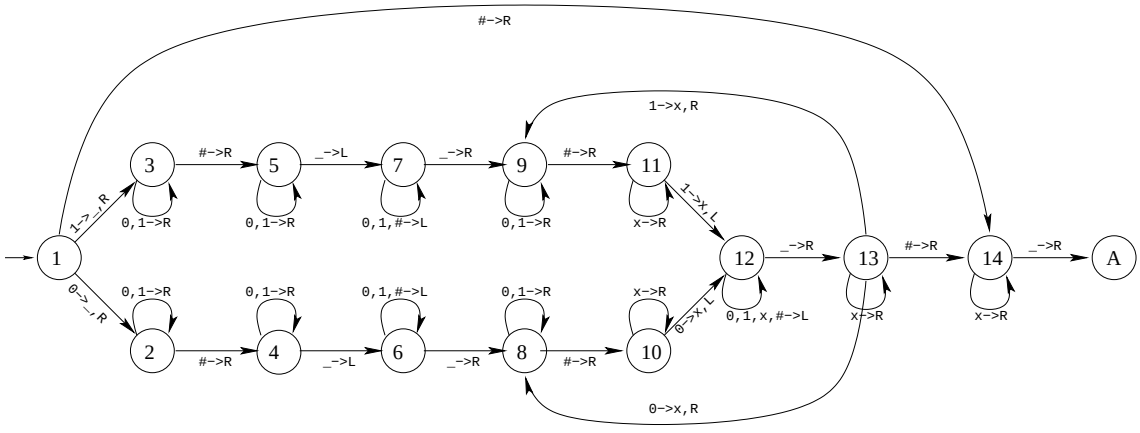

FIG. 3.2 – Diagramme d'état pour la MT des "mots-jumeaux"