Un petit document pour faire le lien et voir la différence fondamentale entre

1. la déterminisation des AFN (Automates Finis Non-déterministes) de TL1 et
2. la déterminisation des MTN (Machines de Turing Non-déterministes).

1 Pour les automates finis

Pour passer d'un AFN (sans ε-transition) à un AFD, c'est « facile » :

Si $\delta(q,a) = \{p_1, ..., p_k\} \quad (k \geq 0)$ alors une config $(q, a.w)$ a k configs suivantes possibles :
$(p_1, w), ..., (p_k, w)$

ce qu'on transforme en déterminisant en :
$(\{q\}, a.w) \vdash (\{p_1, ..., p_k\}, w)$: unique config. suivante

qui se généralise en :
$(\{q_1, ..., q_m\}, a.w) \vdash (\{p_{1_1}, ...p_{1_{k_1}}, ..., p_{m_1}, ..., p_{m_{k_m}}\}, w)$

ce qui nous amène à l'AFD, qui a potentiellement un nombre exponentiel d'états (cf. TL1, exo 9, feuille 3, car $\mathrm{Card}(\mathcal{P}(Q)) = 2^{\mathrm{Card}(Q)}$) mais qui ne change pas le « temps » d'acceptation d'un mot.

2 Pour les machines de Turing

Pour déterminiser une MTN c'est plus compliqué, car maintenant
$\delta(q,a) = \{(p_1, a_1, M_1), ..., (p_k, a_k, M_k)\} \quad (k \geq 0, a_i \in \Gamma, M_i \in \mathcal{M})$

et du coup, une config $c = \alpha q a \beta$ (notée ici $[q, \alpha|a|\beta]$) a k configs suivantes possibles :
$c_1 = [p_1, \alpha_1|X_1|\beta_1] \quad ... \quad c_k = [p_k, \alpha_k|X_k|\beta_k]$

avec α_i, X_i et β_i définis en fonction de a_i et M_i (cf. diapo 4).

On ne peut donc pas juste « factoriser » les états comme pour AFN→AFD, car pour chaque config. suivante, on a aussi

- le contenu du ruban $(...\alpha_i X_i \beta_i ...)$ et
- la position de la tête de lecture (sur X_i)

qui changent et donc, pour le « pas » suivant, c'est de chacune de ces nouvelles configs. qu'il faut partir !
Du coup, la seule option qu'« on » a trouvée c'est :

Pour chaque config. [état, ruban, tête de lecture sur ce ruban] depuis laquelle on veut exécuter un pas (donc pour chaque nœud d'un niveau donné de l'arbre de la diapo 42) et qui a k configs suivantes possibles :

1. créer k nouveaux rubans en y recopiant le ruban de la config courante, et avec pour chacun une tête de lecture à la même position ;
2. faire le « boulot » pour chacun des k nouveaux rubans : changer a, resp. en a_1, ... a_k ; bouger la tête, resp. selon M_1, ... M_k ; changer q, resp. en p_1, ... p_k
3. « jeter » le ruban de la config courante

Et puis c'est reparti...

1. prend un temps $\Omega(k)$
2. prend un temps $\Theta(k)$
3. prend un temps $\Theta(1)$

D'où le temps exponentiel $\Theta(k^p)$ pour simuler p pas d'exécution...
La « simulation » de la diapo 44 fait exactement ça, mais en se « ramenant » tout le temps à 1 ruban (les « bouts de nouveaux rubans, positions de tête de lecture et nouvel état » étant résumés dans les configs ajoutées à la fin)...