PC 9
Automates finis

1. Automates finis
2. Exemples
3. Mots et langages
4. Automates déterministes
5. Expressions rationnelles

Automates finis

Pour *analyser* des suites de symboles, on construit des diagrammes avec:

1. des sommets appelés *états*
2. des transitions étiquetées
3. des actions dans les états de reconnaissance.

On utilise les automates :

- Comme structure de données (généralisation des tries)
- Comme méthode de traitement de textes

Un exemple en traitement du signal

Considérons le problème suivant : on veut éliminer dans une suite binaire les 0 ou les 1 isolés (considérés comme un bruit) (*bounce filter*).

L'automate est représenté sur la figure **??**. La transformation est effectuée suivant la règle très simple :

a	b	c	d
0	0	1	1

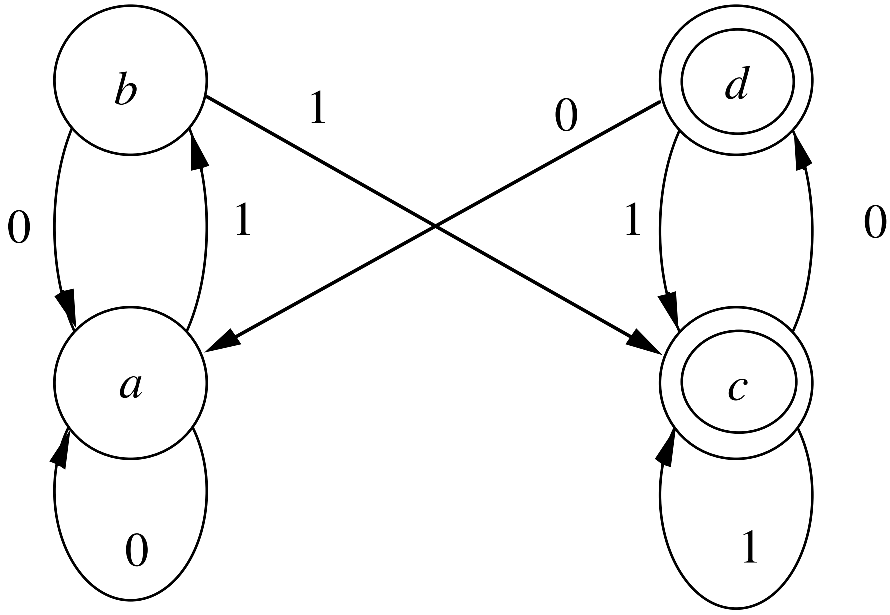

Figure 1: Le filtre binaire

Programmation

La programmation d'un automate est très simple : on suit le chemin en fonction des symboles de la donnée. Dans chaque état, on fait les actions correspondantes.

```
var c: char;
begin
1:   write('0');
     read(c);
     if c = '0' then goto 1;
     if c = '1' then goto 2;
     goto 99;

2:   write('1');
     read(c);
     if c = '0' then goto 1;
     if c = '1' then goto 3;
     goto 99;
 ...
99:
    end;
```

Un autre exemple

L'automate ci-dessous effectue la recherche du mot $p = abaab$ dans un texte. La programmation

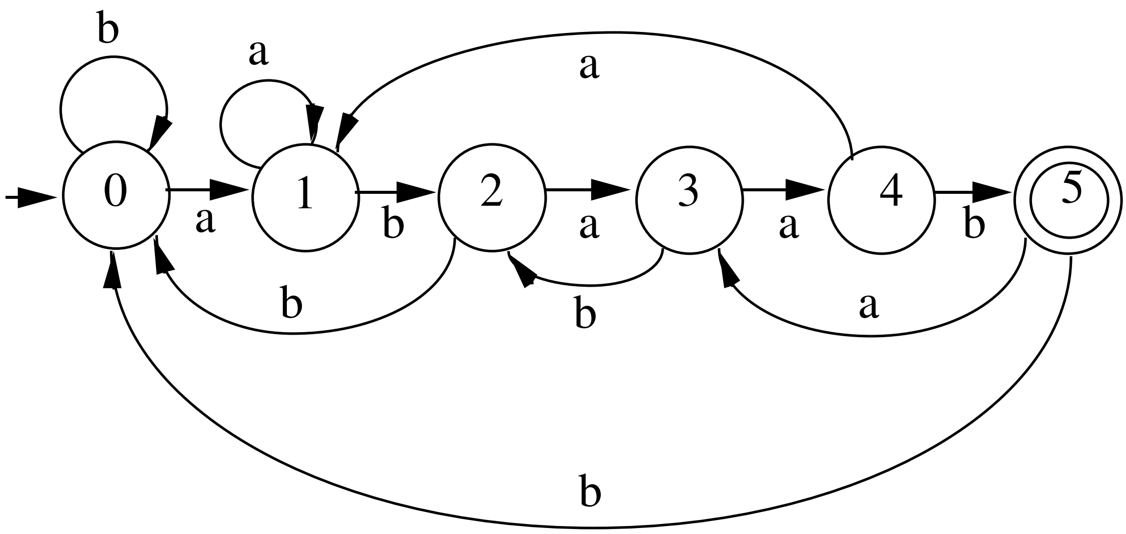

Figure 2: Un automate de recherche

utilise une fonction de remplacement:

n	1	2	3	4	5
$g(n)$	0	1	1	2	2

où $g(1) = 0$ et $g(n) = k$ si le plus long préfixe de $p[1..n-1]$ qui est aussi un suffixe est de longueur $k - 1$.

L'algorithme de Knuth, Morris et Pratt

On cherche le *motif* $p[1..M]$ dans le *texte* $t[1..N]$.

```
function kmp: integer;
 var i,j: integer;
 begin
  i:=1; j:=1; init;
  repeat
   if (j=0) or (a[i]=p[j])
    then begin i:=i+1; j:= j+1 end
    else j:=g[j];
  until (j>M) or (i>N);
  if j>M then kmp:= i-M else kmp:=i;
 end;
```

Complexité

- $O(N+M)$ (au lieu de $O(NM)$ pour l'algorithme naïf). En effet, $i+(i-j)$ augmente strictement à chaque étape.
- Surtout, il ne nécessite jamais de retour en arrière sur le texte.

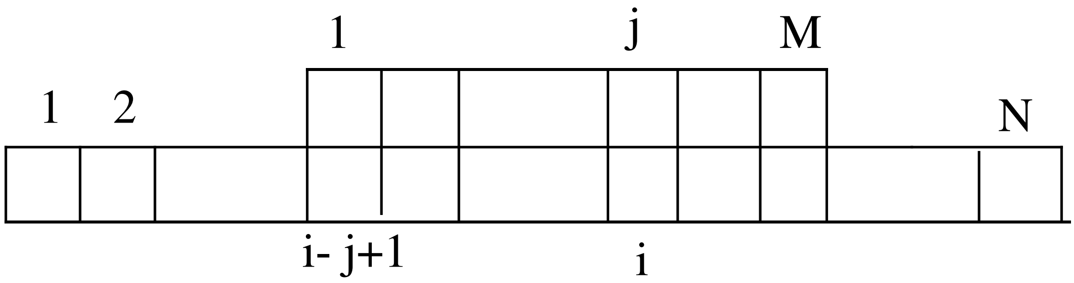

Figure 3: La recherche dans un texte

Applications: recherche dans un dictionnaire (algorithme de Aho Corasick).

Le prétraitement

Le calcul de g est fait par la procédure init:

```
procedure init;
 var i,j: integer;
 begin
  i:=1; j:=0; g[1]:= 0;
  repeat
   if (j=0) or (p[i]= p[j])
    then
     begin i:= i+1; j:= j+1; g[i]:= j end
    else j:= g[j];
  until i>= M;
 end;
```

Mots et Langages Formels

On définit en général un *mot* sur un *alphabet* A comme une suite finie de symboles pris dans A.

L'alphabet A peut être l'ensemble des caractères de base et les mots les lexèmes ou:

	alphabet	mots
code ASCII	0,1	caractères
analyse lexicale	caractères	lexemes
analyse syntaxique	lexemes	programmes

On utilise le *mot vide* noté ϵ.

Un *langage formel* est un ensemble de mots.

Automates déterministes

Un automate est déterministe si:

1. Il possède un seul état initial
2. De chaque état part au plus une flèche étiquetée par chaque symbole.

Par exemple, l'automate:

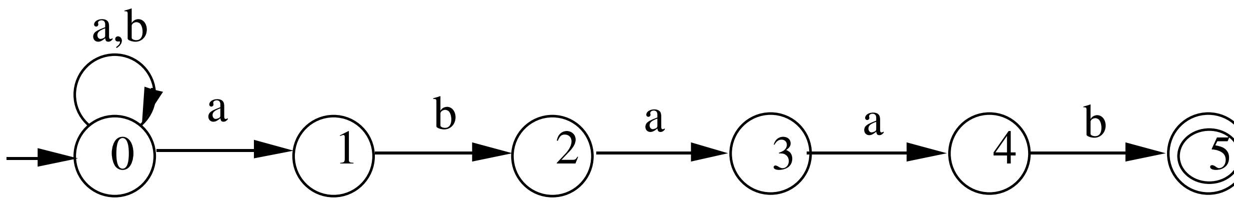

Figure 4:

n'est pas déterministe.

Déterminisation

Tout automate fini est équivalent à un automate fini déterministe. C'est une propriété fondamentale des automates.

Algorithme: on part d'un automate non-déterministe $\mathcal{A} = (Q, I, T)$. On construit l'automate déterministe $\mathcal{D}$ ayant

1. comme états les parties de Q
2. comme état initial I
3. comme transitions: (U, a, V) si V est l'ensemble des états atteignables depuis U par une transition a
4. comme états terminaux les $U \subset Q$ tels que $U \cap T \neq \emptyset$.

Complexité: si $\mathcal{A}$ a n états, $\mathcal{D}$ peut avoir 2^n états.

Exemple

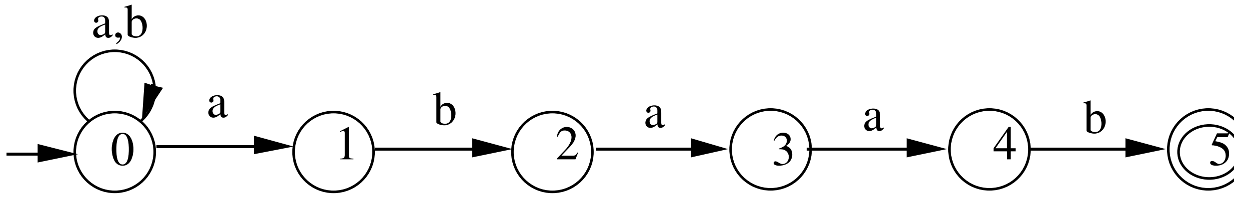

Figure 5: Un automate non-déterministe

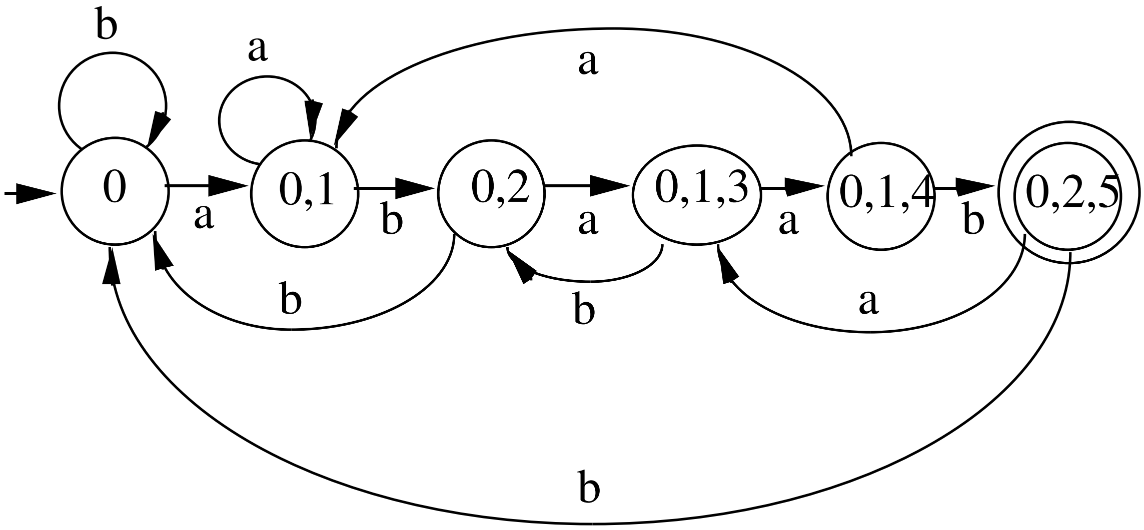

Figure 6: Le résultat de l'algorithme

Généralisation

On peut admettre des transitions d'étiquette ϵ (transitions *spontanées*).

Pour déterminiser, il faut cette fois

1. prendre pour l'étape 2 les états atteignables depuis I par un ϵ-chemin.
2. et pour 3 l'ensemble des états atteignables par un *chemin* d'étiquette a.

Expressions Rationnelles

Pour *spécifier* des suites de symboles, on utilise

1. l'union notée | (ou +)
2. La concaténation (pas de symbole)
3. L'itération *

Utilisation

- Editeurs de texte (vi ou ed)
- Spécifications lexicales (Lex)
- commande grep sous Unix:

  grep <expression> <fichier>

  imprime les lignes du fichier contenant un bloc satisfaisant l'expression.

Exemples

Identificateurs en Pascal:

$$\begin{aligned}\textbf{lettre} &\rightarrow A\,|B\,|\ldots|Z\,|a\,|b\,|\ldots|z\\ \textbf{chiffre} &\rightarrow 0\,|\,1\,|\,2\,|\ldots|\,9\\ \textbf{id} &\rightarrow \textbf{lettre (lettre}\mid\textbf{chiffre)*}\end{aligned}$$

Nombres sans signe en Pascal:

$$\begin{aligned}\textbf{chiffre} &\rightarrow 0\,|\,1\,|\,2\,|\ldots|\,9\\ \textbf{chiffres} &\rightarrow \textbf{chiffre chiffre*}\\ \textbf{frac} &\rightarrow \textbf{.chiffres}\mid\epsilon\\ \textbf{exp} &\rightarrow (E(+\mid\ -\ |\epsilon)\textbf{chiffres}\mid\epsilon\\ \textbf{num} &\rightarrow \textbf{chiffres frac exp}\end{aligned}$$

Avec les abréviations:

$$\begin{aligned}\textbf{chiffre} &\rightarrow 0\,|\,1\,|\,2\,|\ldots|\,9\\ \textbf{chiffres} &\rightarrow \textbf{chiffre}^{+}\\ \textbf{frac} &\rightarrow (\textbf{.chiffres})?\\ \textbf{exp} &\rightarrow (E(\,+\mid-)?\textbf{chiffres})?\\ \textbf{num} &\rightarrow \text{chiffres frac exp}\end{aligned}$$

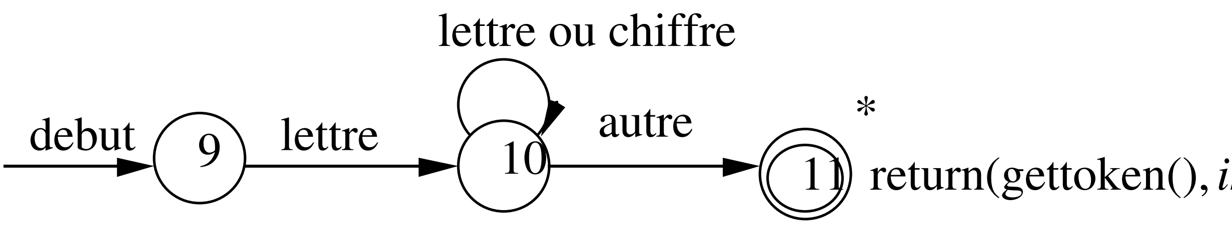

Figure 7: Identificateurs en Pascal

La construction récursive

On peut construire récursivement un automate fini reconnaissant une expression rationnelle. L'automate construit est tel que:

1. Il a un seul état initial i
2. Il a un seul état final $t \neq i$
3. Aucune flèche ne rentre dans i
4. Aucune flèche ne sort de t

Pour ϵ et a, on prend:

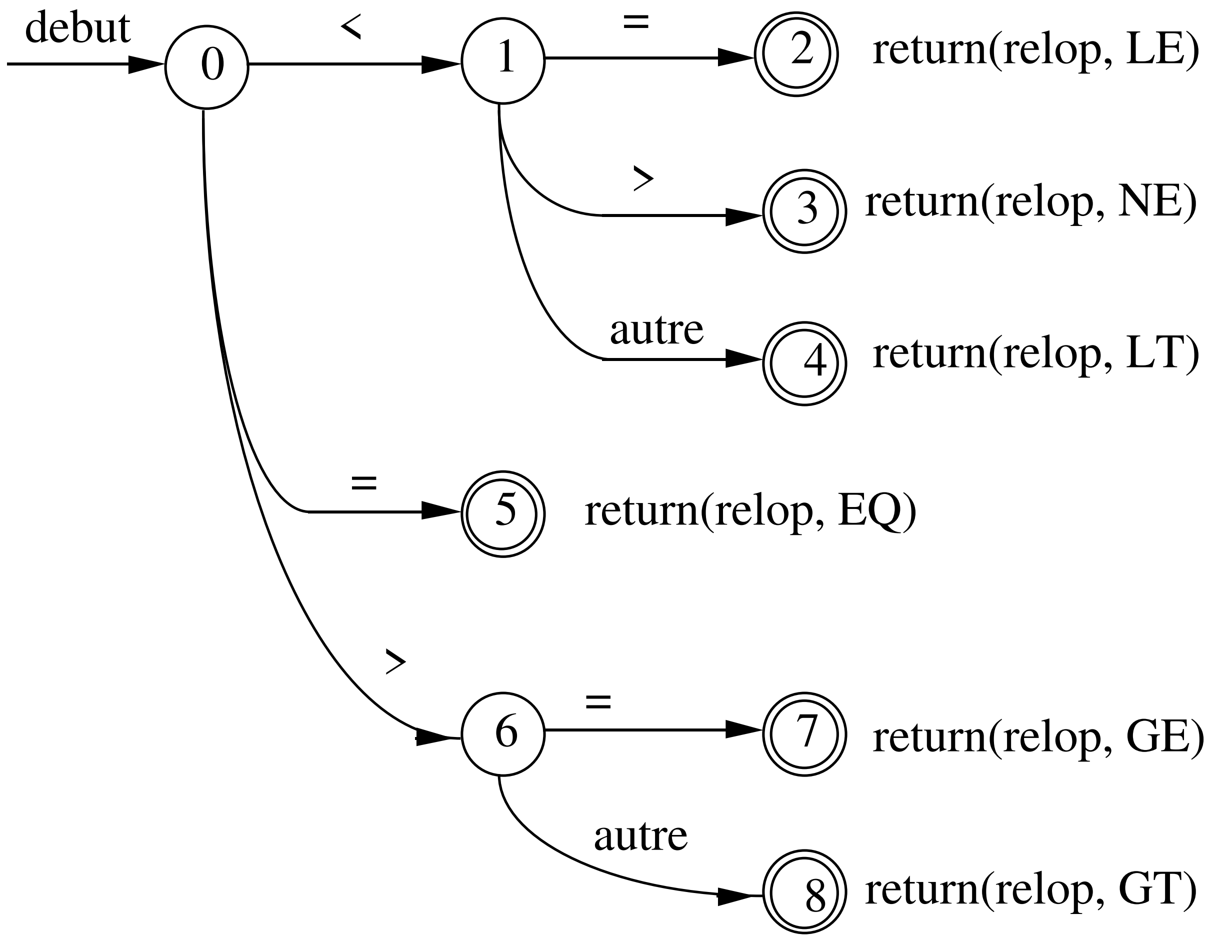

Figure 8: Opérateurs relationels

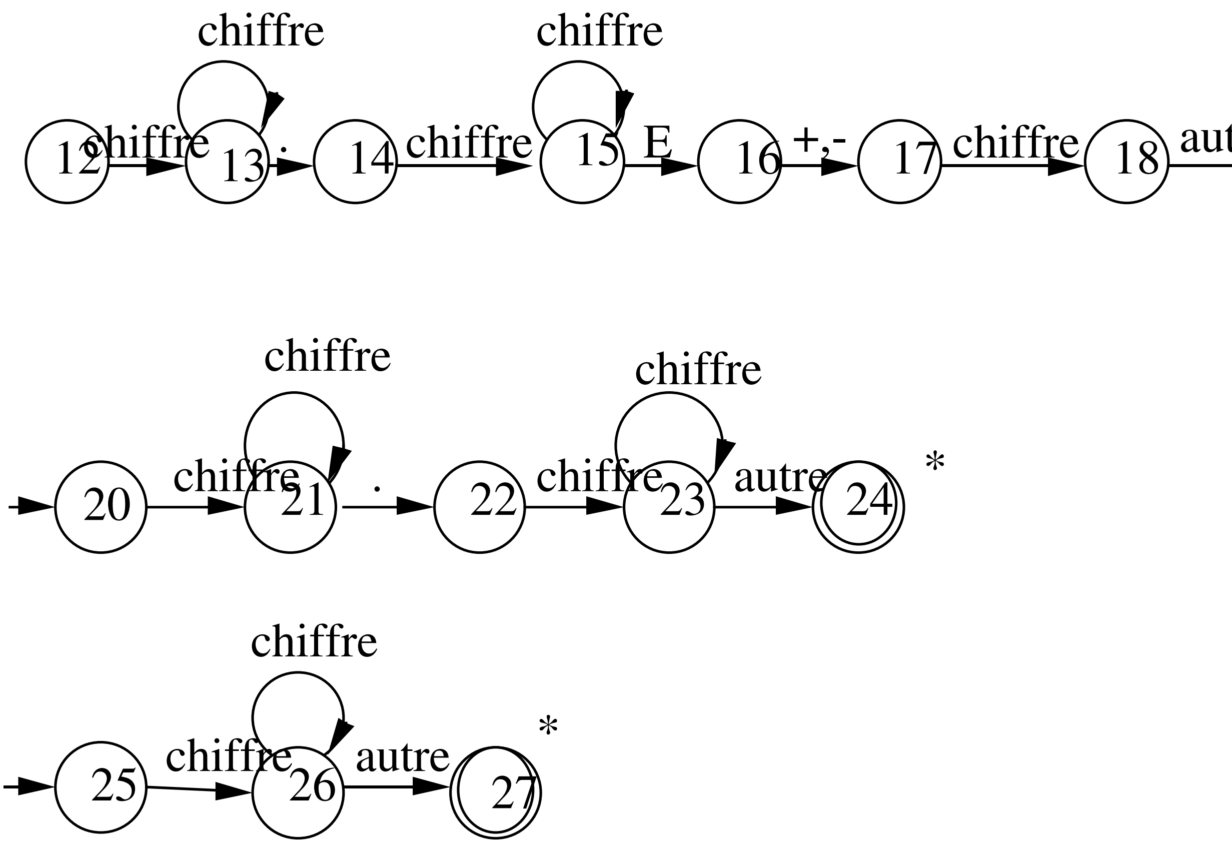

Figure 9: Nombres sans signe

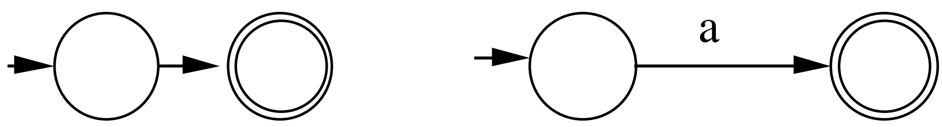

Figure 10:

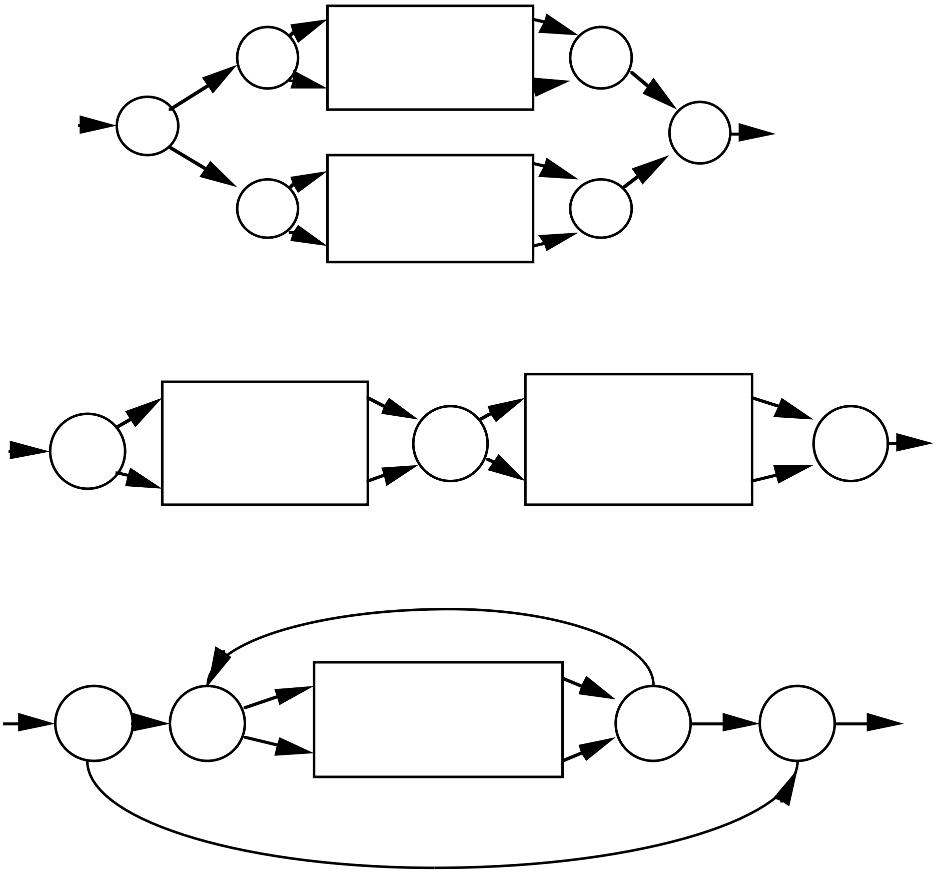

Figure 11: Union, produit et étoile