# Expérience de programmation générique sur des structures non-séquentielles : les automates

*IGM - Université de Marne La Vallée*

Vincent Le Maout

# Plan

1. Motivations
2. La programmation générique
3. STL
4. ASTL
5. Les curseurs
6. Word Grep
7. Librairies actives

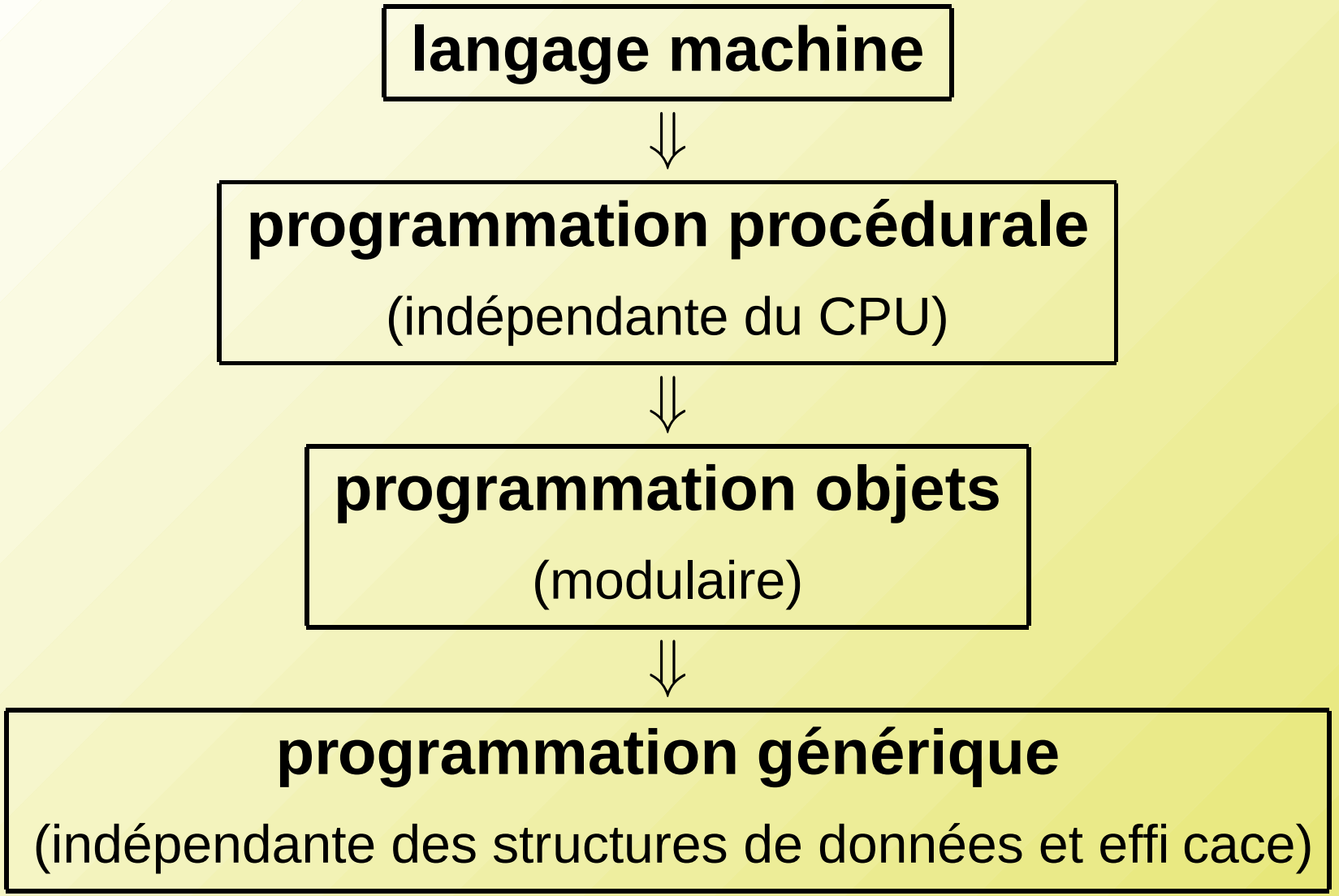
# Motivations

- Outil de manipulation d'automate
- Adaptable
- Efficace
- Réutilisable
- Evaluation de la programmation générique en C++

# Librairies

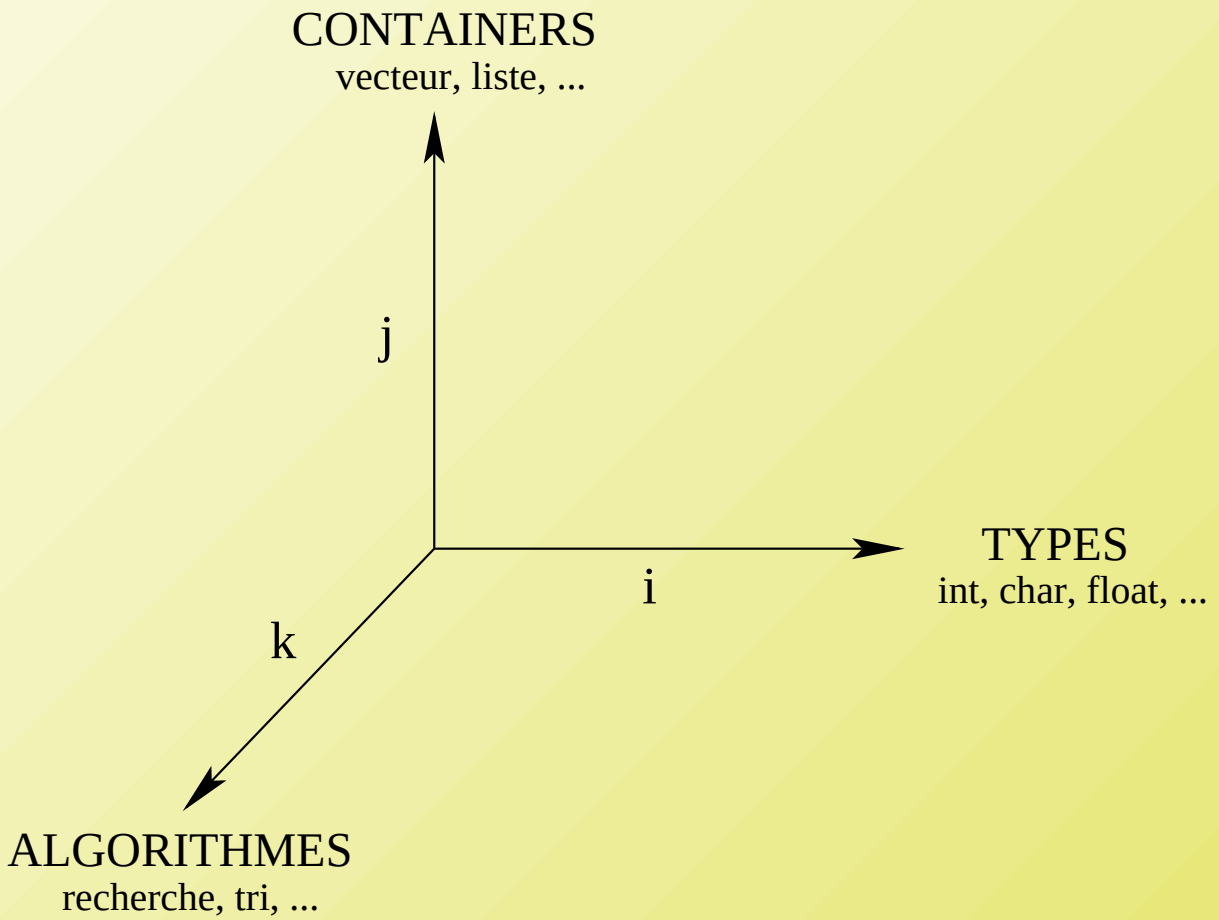
- **LEDA**, Library of Efficient Data types and Algorithms
- **BGL**, Boost Graph Library
- **GTL**, Graph Template Library
- **Grail+**
- **FIRE Lite**, FInite automata and Regular Expressions
- **AUTOMATA**
- **FSA**, Finite State Automata utilities

# Réutilisation



# Programmation générique

## Décomposition orthogonale de l'espace des composants



# Approche orientée algorithme

Position du plus petit élément d'une séquence

```
Iterateur min(Iterateur x, Iterateur y)
{
  Iterateur p = x;
  while (x != y)
  {
    if (*x < *p) p = x;
    ++x;
  }
  return p;
}
```

# La programmation générique

| Pour pouvoir écrire | `Iterateur` doit |
|---|---|
| `r = x` | Être copiable |
| `x != y` | Définir une relation d'équivalence |
| `++x` | Être incrémentable |
| `*x` | Être déréférençable |

$\Rightarrow$ définition d'abstractions (concepts)

# La programmation générique

### Valid expressions

In addition to the expressions defined in Trivial Iterator, the following expressions must be valid.

| Name | Expression | Type requirements | Return type |
|---|---|---|---|
| Preincrement | `++i` | | X& |
| Postincrement | `(void)i++` | | |
| Postincrement and dereference | `*i++` | | T |

### Expression semantics

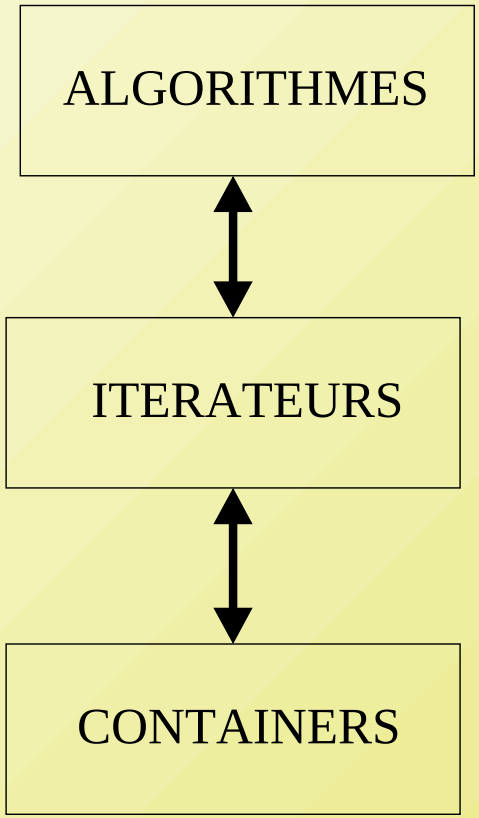
| Name | Expression | Precondition | Semantics | Postcondition |
|---|---|---|---|---|
| Dereference | `*t` | `i` is incrementable | | |
| Preincrement | `++i` | `i` is dereferenceable | | `i` is dereferenceable or past-the-end [3] [4] |
| Postincrement | `(void)i++` | `i` is dereferenceable | Equivalent to `(void)++i` | `i` is dereferenceable or past-the-end [3] [4] |
| Postincrement and dereference | `*i++` | `i` is dereferenceable | Equivalent to `{T t = *i; ++i; return t;}` | `i` is dereferenceable or past-the-end [3] [4] |

### Complexity guarantees

All operations are amortized constant time.

# La programmation générique

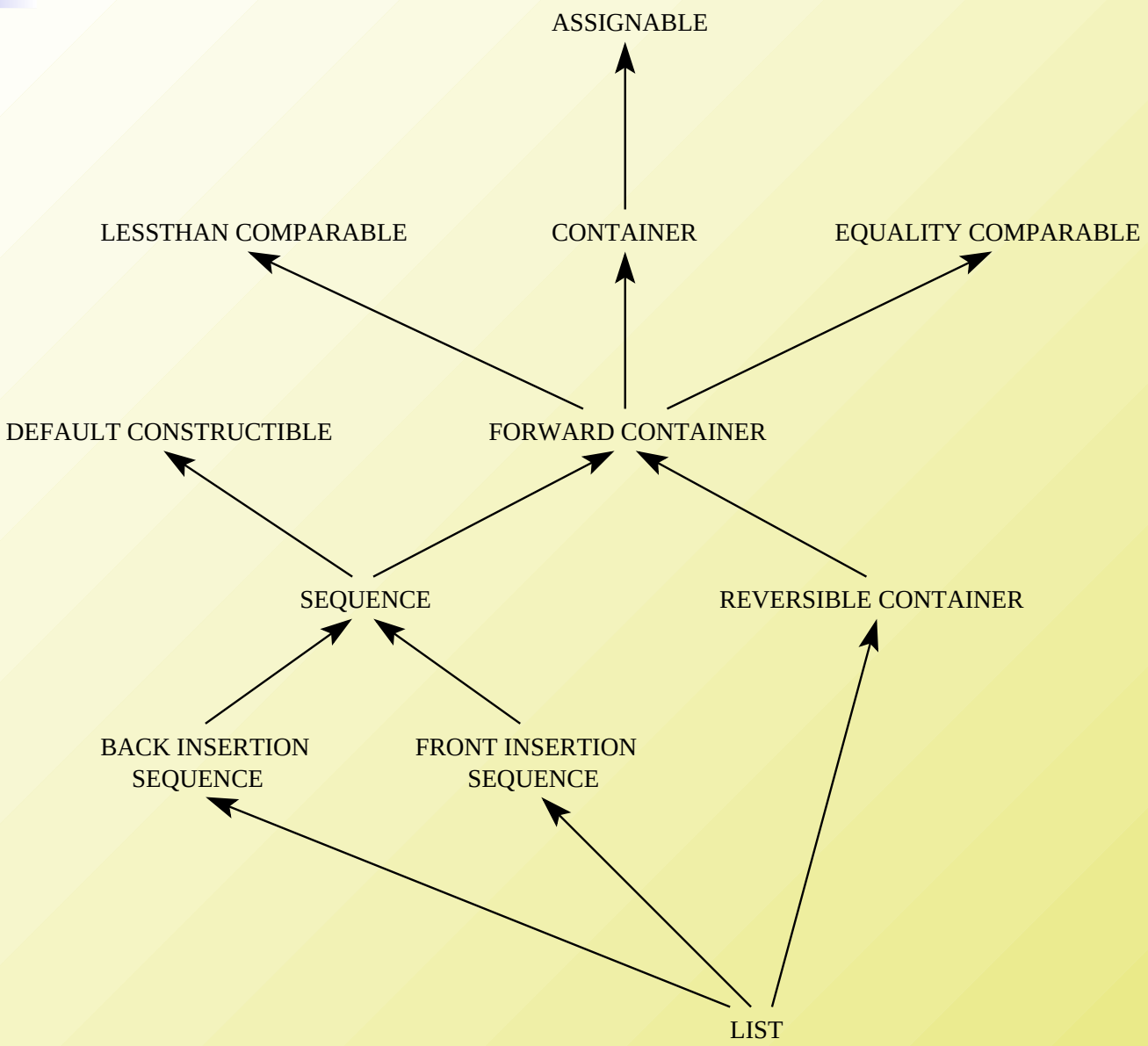
Découplage total entre algorithme et structure de données



# La Standard Template Library

- Composants logiciels standards fournis avec les compilateurs de C++
- 5 familles d'abstractions : algorithmes, itérateurs, containers, adaptateurs, allocateurs
- Les concepts se présentent sous la forme de tableaux de contraintes imposées aux interfaces des objets
- Garantit l'indépendance des algorithmes vis-à-vis des données

# La Standard Template Library




Héritage de concept

# ASTL

- Composants génériques fondamentaux de manipulation d'automates
  - Construction, comparaison, copie
  - Opérations ensemblistes
  - Déterminisation, minimisation
  - Expressions rationnelles
- Définit trois sortes d'abstractions : algorithme, container d'automate et curseur

# ASTL

Les containers $A(\Sigma, Q, I, F, \Delta, T, \tau)$

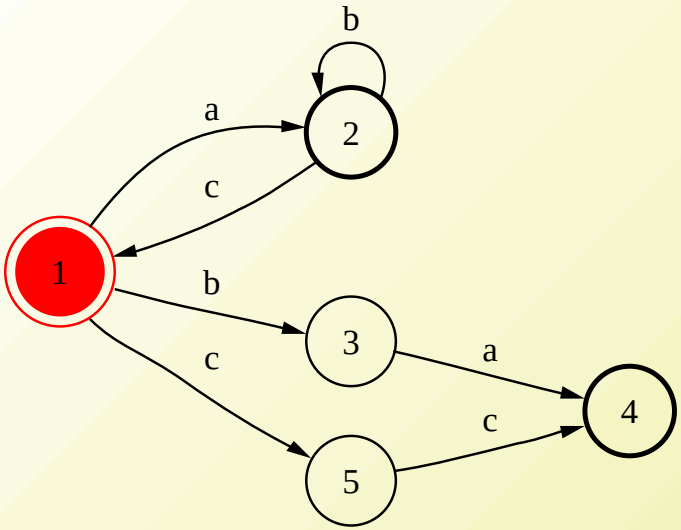
- Indépendants du type d'alphabet
- Diversité des caractéristiques :
  - DFA_matrix
  - DFA_map, DFA_hash
  - DFA_compact
- Interface simple et petite facilitant la spécialisation

# Les curseurs

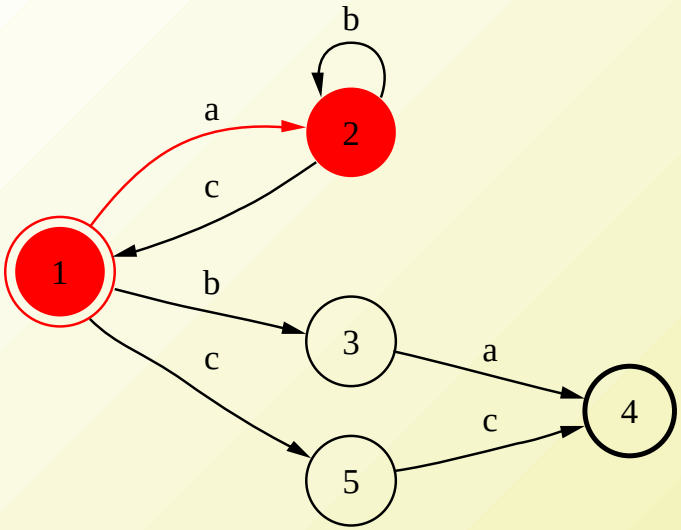
- Itérateur sur des containers d’automate
- Matérialise une position
- Évolue sur l’automate

# Les curseurs

- `c = 1`

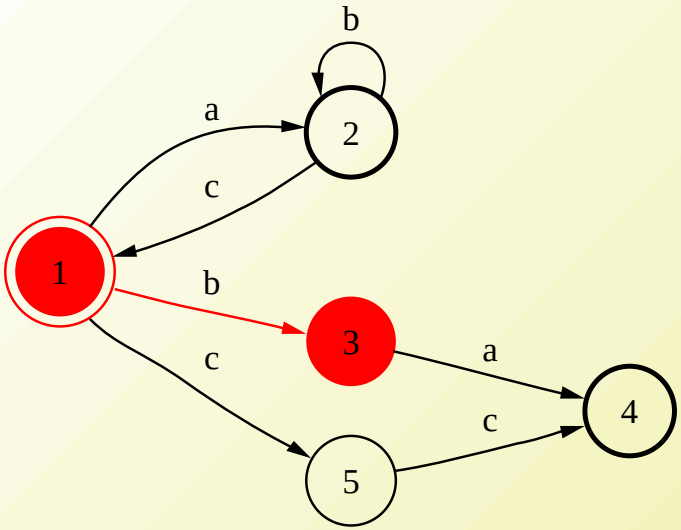


# Les curseurs



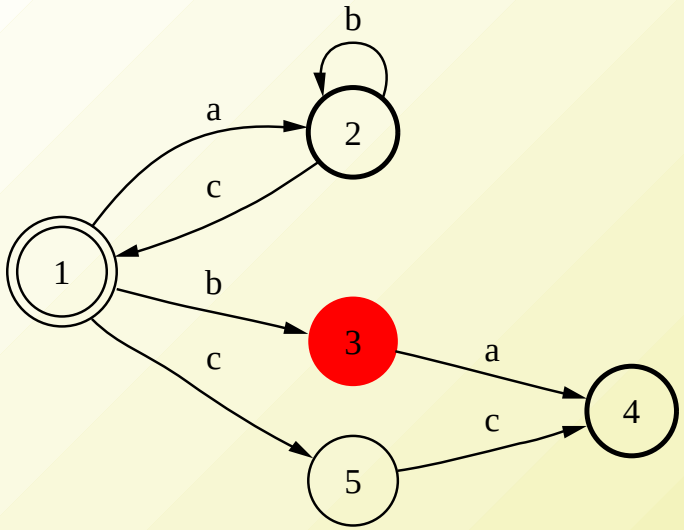

- `c = 1`
- `c.first_transition()`

# Les curseurs



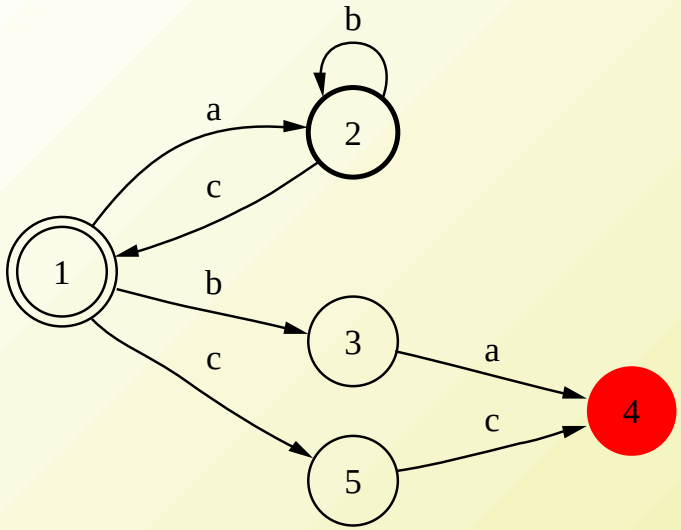

- `c = 1`
- `c.first_transition()`
- `c.next_transition()`

# Les curseurs



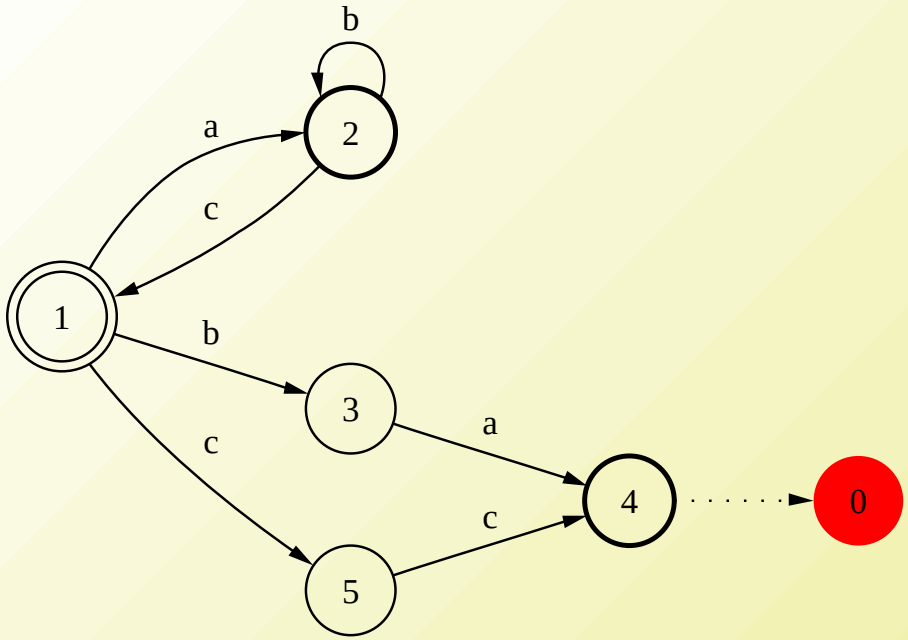

- `c = 1`
- `c.first_transition()`
- `c.next_transition()`
- `c.forward()`

# Les curseurs




- `c = 1`
- `c.first_transition()`
- `c.next_transition()`
- `c.forward()`
- `c.forward('a')`

# Les curseurs




- `c = 1`
- `c.first_transition()`
- `c.next_transition()`
- `c.forward()`
- `c.forward('a')`
- `c.forward('a')`

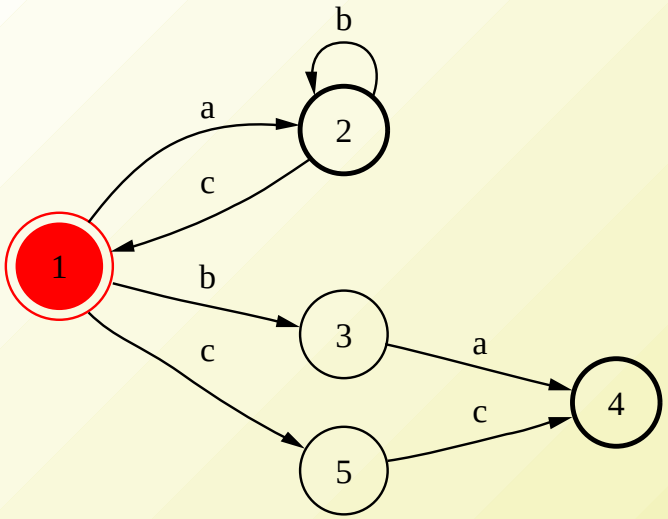
# Les curseurs

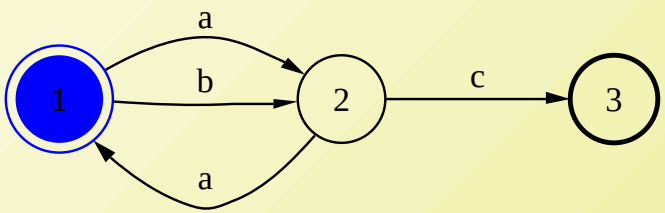
```
template <typename Iterateur, typename Curseur>
bool appartient(Iterateur x, Iterateur y,
                Curseur c)
{
  if (c.sink()) return false;

  while (x != y && c.forward(*x))
    ++x;

  return x == y && c.src_final();
}
```

# Les adaptateurs

- curseurs modifiant le comportement d'autres curseurs
  - voisins d'un mot
- permettent d'ajouter des comportements en ne travaillant qu'au niveau de l'adaptateur
  - filtres
  - déterminisation
- permettent une infinité de combinaisons
  - combinaison d'opérations ensemblistes

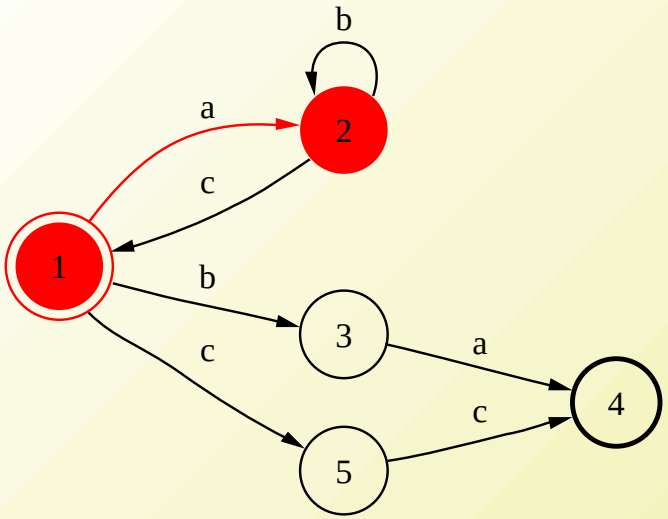
# Curseur d’intersection

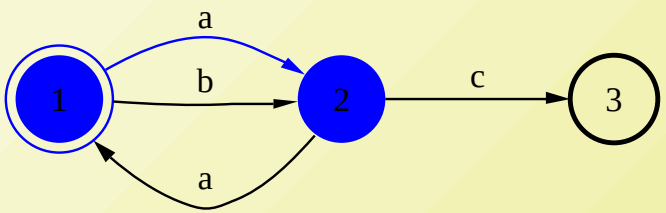



- `c = (1, 1’)`

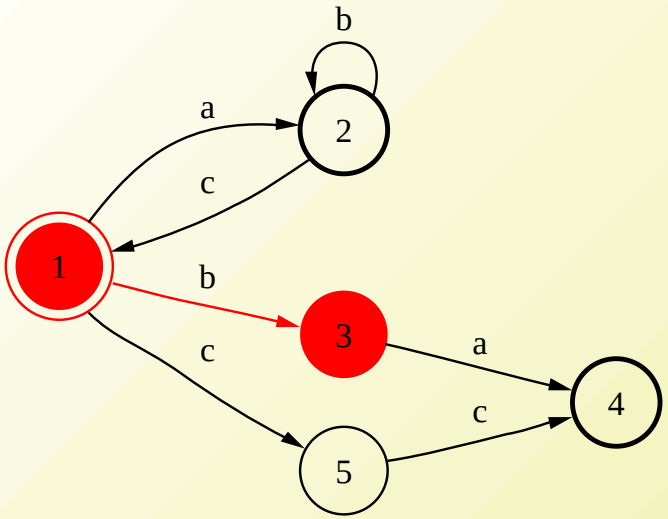
# Curseur d’intersection

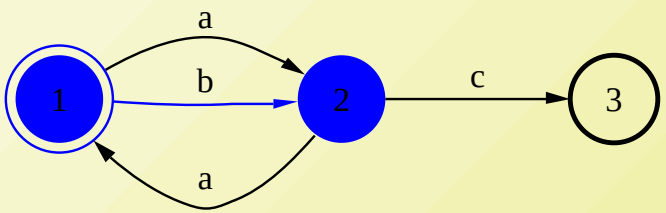



- `c = (1, 1’)`
- `c.first_transition()`

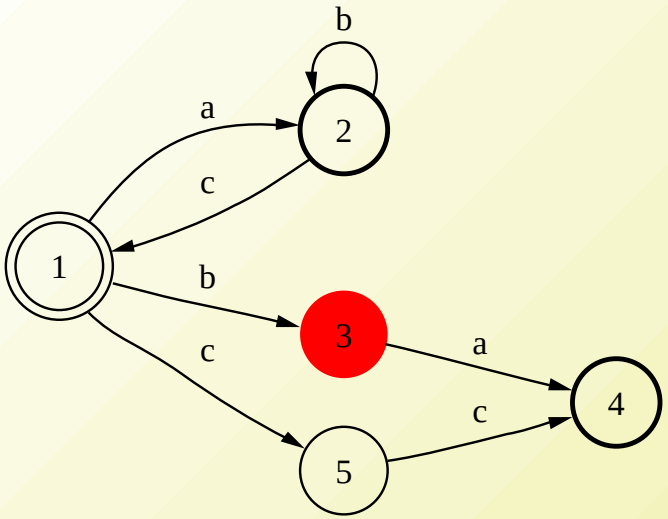
# Curseur d’intersection

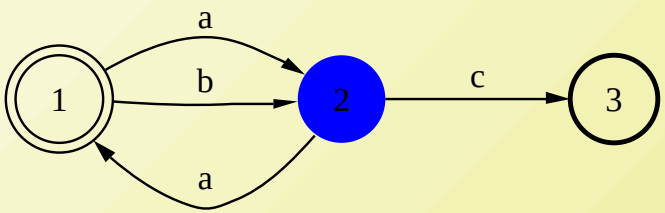



- `c = (1, 1’)`
- `c.first_transition()`
- `c.next_transition()`

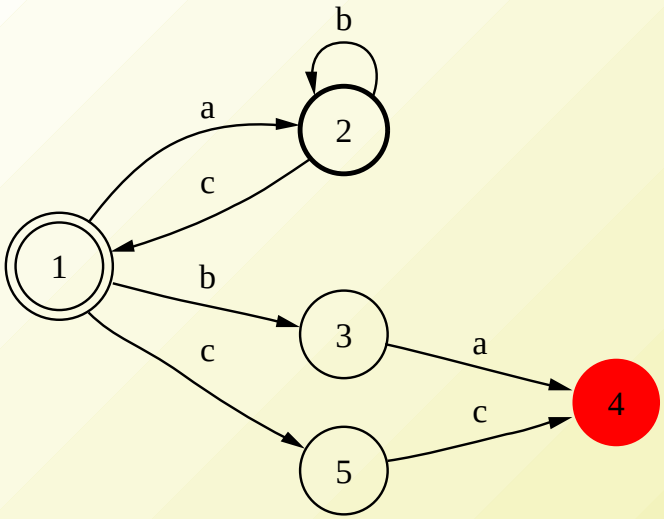
# Curseur d’intersection

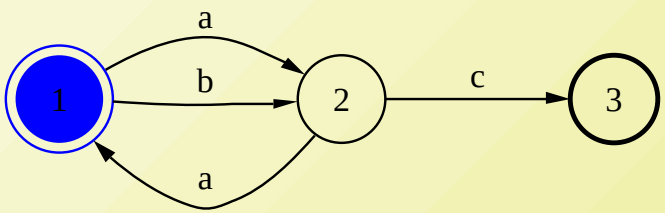



- `c = (1, 1’)`
- `c.first_transition()`
- `c.next_transition()`
- `c.forward()`

# Curseur d’intersection

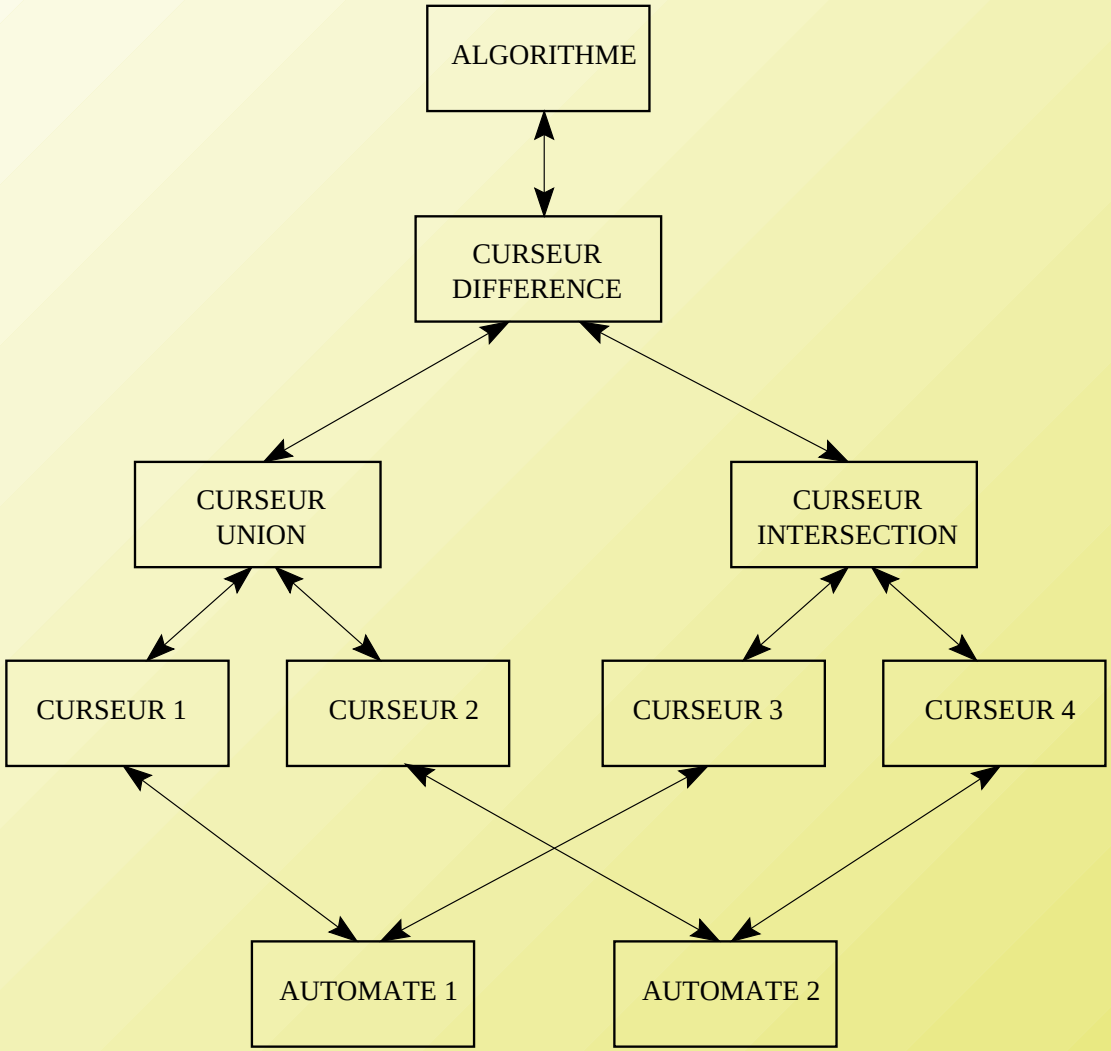



- `c = (1, 1’)`
- `c.first_transition()`
- `c.next_transition()`
- `c.forward()`
- `c.forward(’a’)`

# Différence symétrique

$$A \Delta B = (A \cup B) \setminus (A \cap B)$$



# Les curseurs sans automate

- **curseur de chaîne**, **générateur aléatoire**, $\Sigma^*$
  - Simple
  - Économique
- **curseur des permutations**
  - Évite l'explosion combinatoire
- **curseur d'expression rationnelle**
  - simule l'automate
- **curseur de lecture sur flux**
  - Lit sans reconstruire
  - Évite les problèmes d'espace mémoire

# Copie

- Utilise un curseur comme source et un container comme cible
- Indépendant des types de curseur et de container
- Copie par parcours en profondeur

copie + curseur d'intersection = algorithme de construction de l'intersection

copie + curseur de lecture sur flux = algorithme de chargement d'automate

# Construction paresseuse

- Système de cache
- Construction de l'automate "à la demande" : seules les parties utilisées sont construites
- La représentation matricielle de l'automate de cache est optimale en temps (très peu d'opérations par caractère)
- Automatique :

```
regexp_cursor e("(a|b)*abb");
lazy_cursor<regexp_cursor> c = e;
```

# Construction paresseuse

L'algorithme doit vérifier trois propriétés :

1. Accès non équiprobables aux transitions
2. L'algorithme n'implique pas l'automate en entier
3. Le calcul concernant un état ne dépend que de ses propriétés locales

- Recherche de motifs en langue naturelle
- Evite de recalculer le curseur de déterminisation
- Accélère les curseurs composés

# Word Grep

Extraction d'information par recherche de motifs définis sur les mots :

```
(Doris|Del|Jack|Joe|Jerry|Bernie|James|...) \
@{0,3} \
(senior|executive|managing|marketing|...)? \
(vice|former|ex|co)? \
(chairman|Chmn|Pres|president|director|...) \
(of|for) the? @{0,10} \
(Corp|Inc|Ltd|software|Group|centre|...)
```

# Résultats

**James** Gosling, **vice president of** Sun Microsystems **Inc**
**John** Greaves, **chairman of** One.Tel **Ltd**
**Ken** Sooley, Solect **director for** Asia Pacifi c, **the company**
**Jack** Myers, **chairman of the** Myers **Group**
**David** Thompson, **president of** Orbital Sciences **Corp**
**John** Sarazen, **director of** SynComm **Group**
**Michael** T. Devlin, **president of** Rational **Software**
**Peter** Nighswander, **vice president of the** research **firm**
**Dominic** Leung, **managing director of** Intisoft Asia **Ltd**
**Lee**, Rx.com's **director of** communications. **The company**
**Jim** Bidzos, **president of** RSA Data Security, **Inc**

# Word Grep

Expressions multiples et combinaisons :

```
logiciel[5]|soft|programme|application[5]
apprentissage|apprendre|enseigner|enseignement
langues étrangères?[5]|langue étrangère?[5]
enfant[5]|enfants[5]|gamin|gamins
```

(A) ’1 2 3 4’
(B) "1 2 3 4"
(C) ’1 2 3’
(D) "1 2 3"
(E) "2 3 4"
(F) "2 3"

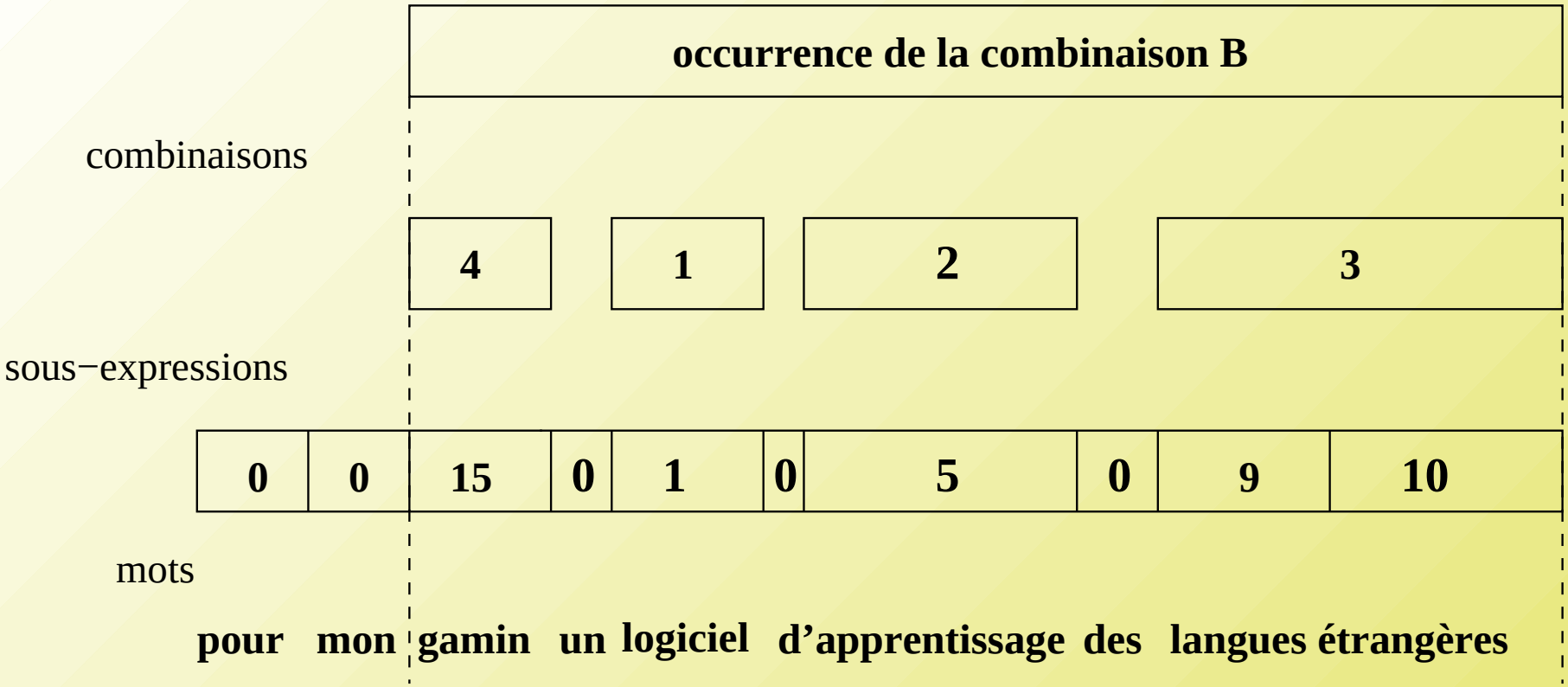
# Résultats

Nous proposons aux internautes de nouvelles méthodes d'enseignement des langues on-line.

Je recherche pour mon gamin un logiciel d'apprentissage des langues étrangères.

(A) '1 2 3 4'
(B) "1 2 3 4"
(C) '1 2 3'
(D) "1 2 3"
(E) "2 3 4"
(F) "2 3"

# Word Grep


occurrence de la combinaison B
combinaisons
4
1
2
3
sous−expressions
0
0
15
0
1
0
5
0
9
10
mots
pour
mon
gamin
un
logiciel
d’apprentissage
des
langues
étrangères

# Word Grep

- Compilé sous Linux, exécutable 68 Ko
- Consommation mémoire $\simeq 1$ Mo
- Entre 1 et 3 millions de mots par seconde
- Bonne alternative aux moteurs de recherche lorsque le volume des données $\simeq 100$ Mo
- Simplification du déploiement et de la maintenance
- Indépendance vis-à-vis des technologies externes

# Un exemple de type concret

```
dfirst_cursor<stack_cursor<
  forward_cursor<DFA_compact<
      DFA_matrix<range<int, 0, 255>,
                 pair<int,string> > > >,
  vector<forward_cursor<DFA_compact<
      DFA_matrix<range<int, 0, 255>,
                 pair<int,string> > > > > >
                          mon_curseur;
```

# Librairies actives

Les librairies activent orientent le compilateur sur le choix de l'implémentation qui répond le mieux aux exigences de l'utilisateur.

- Présente une interface plus abstraite permettant de spécifier un type formel d'automate
- Automatisation de la construction du type concret

# Domain Specific Language

- Langage adapté et réduit à un domaine précis
- Abstraction sans lien direct avec l'informatique et la mise en œuvre
- Couvre tous les besoins potentiels

Le DSL "machine à états" permet :

- Déterminisme/non déterminisme
- Compromis espace/temps
- Type des étiquettes
- Aide au déboguage (*bound checking*)

# Métaprogrammation en C++

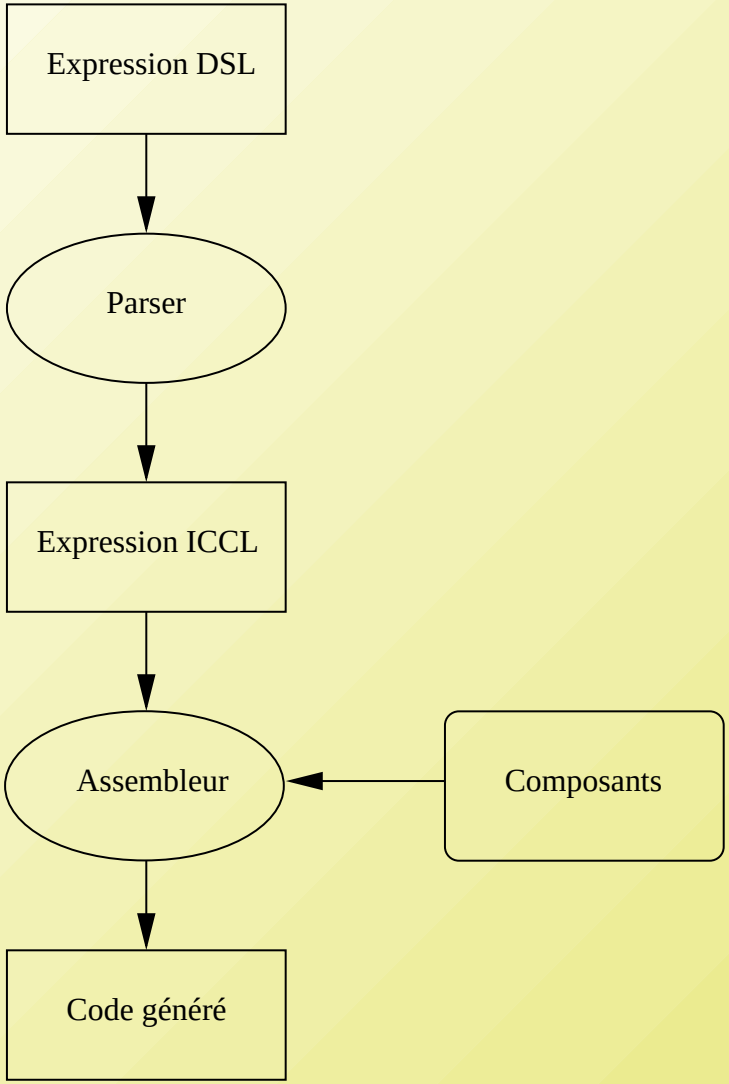
- Spécifications utilisateurs sous formes de patrons d'expressions

```
typedef key<char, deterministic> key_t;
typedef access<constant_time<> > access_t;
typedef fsm<key_t, access_t,
            speed, directed> fsm_type;
```

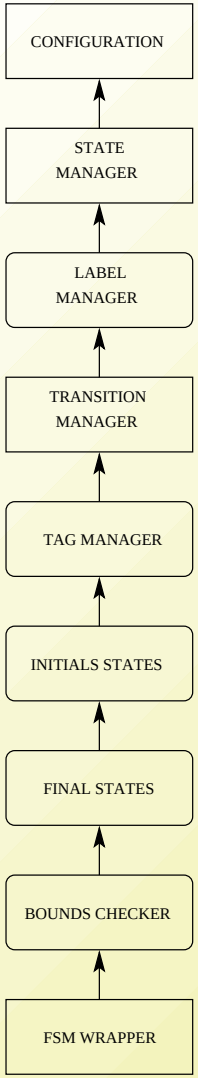
- Construction automatique d'un type

```
typedef fsm_generator<fsm_type>::R
                      concrete_fsm_type;
```

# Double interface



# Programmation par *mixin*




- Types composites
- Empilement des fonctionnalités
- Séparation des aspects

# Métaprogrammation en C++

- Permet de générer $\simeq 3000$ structures de données
- Le DSL est réutilisable pour générer des curseurs

```
typedef cursor_generator<fsm_type>::R
                                    cursor_type;
```

- Redondance minimale
- Effi cacité conservée

# Conclusion

ASTL

- apporte plus de souplesse grâce aux curseurs
- constitue la première implémentation réellement générique
- valide l'architecture à trois couches

La programmation générique en C++

- combine haut-niveau et efficacité
- difficile d'accès

# Conclusion

Les librairies actives et la métaprogrammation

- facilitent l'utilisation
- améliorent la réutilisation
- séparent les aspects

# Prochainement disponible...

## ASTL version 2.0

Lesser GNU Public License

Accès CVS

`https://sourceforge.net/projects/astl/`

Page officielle

`http://astl.sourceforge.net/`