Vaucanson

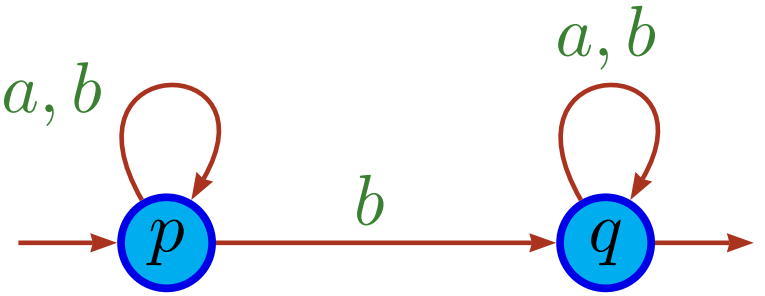
Sylvain Lombardy

Le projet Vaucanson

- Vaucanson-G Package LaTeX pour dessiner des automates
- Vaucanson-XML Format XML de description d'automates (xsd)
- Plateforme Vaucanson Logiciel de manipulation d'automates
  - bibliothèque C++
  - ensemble de binaires
  - interface graphique (en développement)

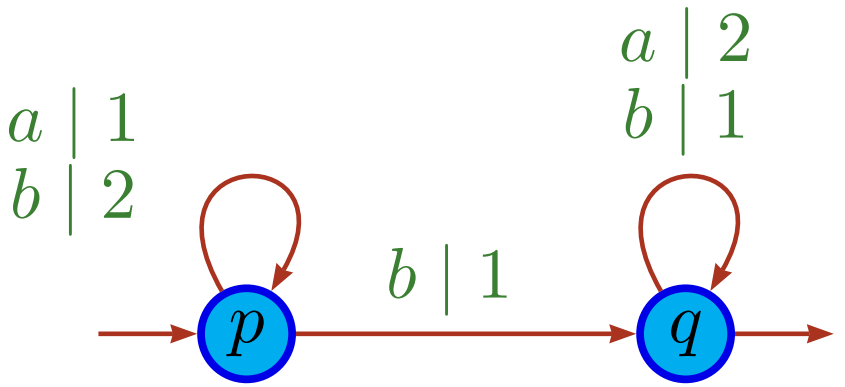
Automates ?

Automates "classiques", accepteurs, booléens



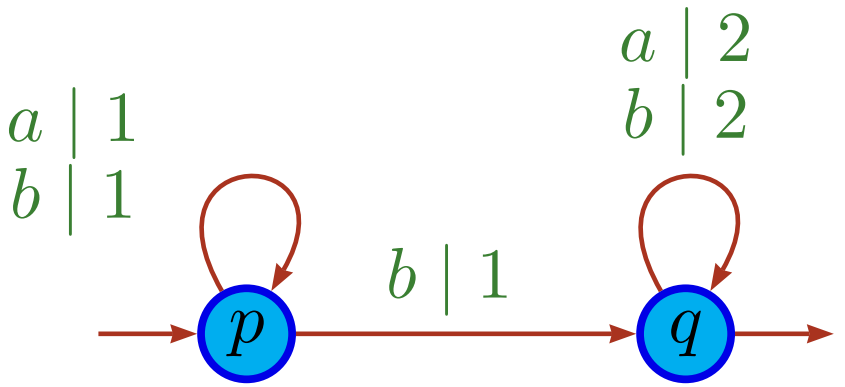
Automates ?

Automates à distance, min-plus



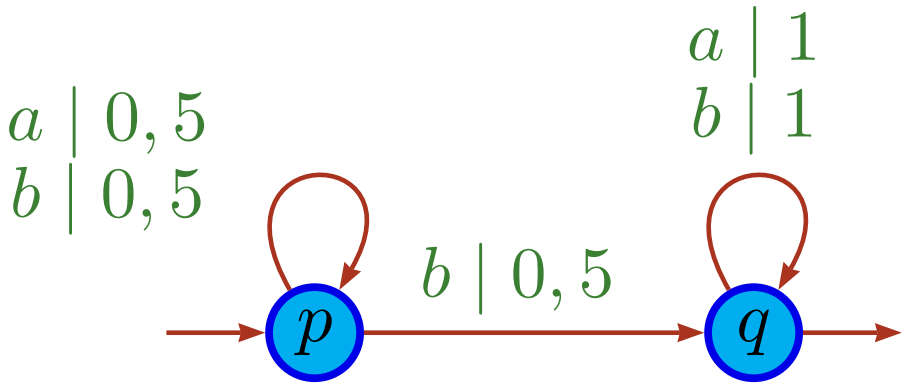
Automates ?

Automates à multiplicité



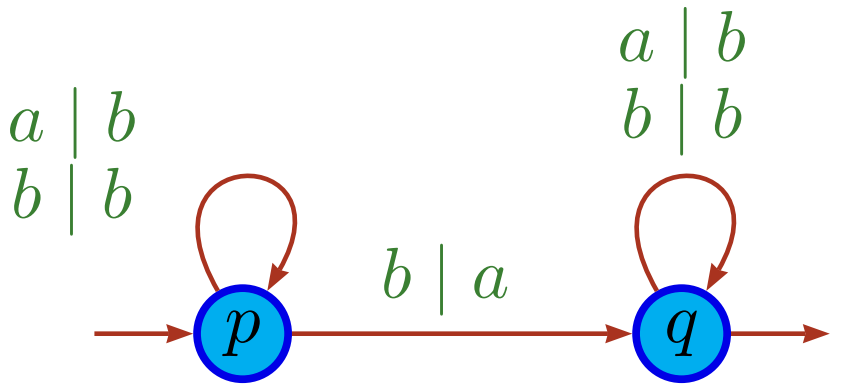
Automates ?

Automates probabilistes



Automates ?

Transducteurs, automates avec sorties



TAF-Kit

Pour les automates, les $\mathbb{Z}$-automates et les transducteurs, on dispose de binaires qu'on appelle Typed Automata Functions Kit.

Chaque binaire lit des automates décrits en Vaucanson-XML ou FSM

écrit des automates en Vaucanson-XML, FSM ou DOT

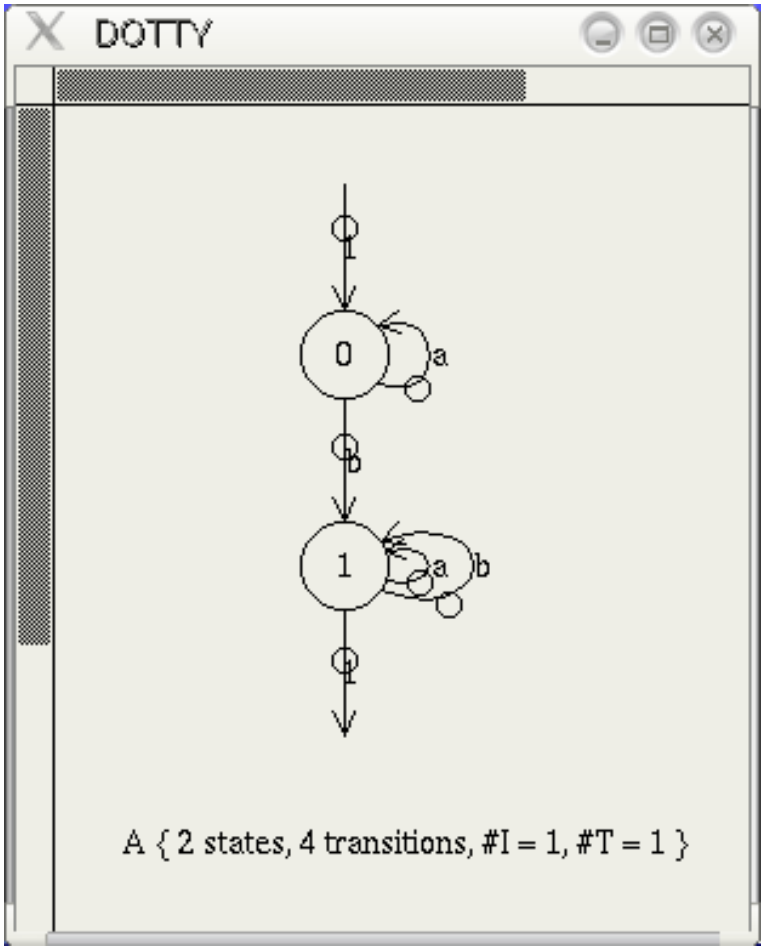
On peut donc les chaîner.

TAF-Kit

```
[vaucanson ~]$ vcsn-b dump-automaton b1

<automaton xmlns="http://vaucanson.lrde.epita.fr" name="b1">

  <labelType>
    <monoid generatorType="letters" type="free">
      <generator value="a"/>
      <generator value="b"/>
    </monoid>
    <semiring operations="classical" set="B"/>
  </labelType>

  <content>
    <states>
      <state name="s0"/>
      <state name="s1"/>
    </states>
    <transitions>
      <transition dst="s0" label="a" src="s0"/>
      <transition dst="s0" label="b" src="s0"/>
      <transition dst="s1" label="b" src="s0"/>
      <transition dst="s1" label="a" src="s1"/>
      <transition dst="s1" label="b" src="s1"/>
      <initial state="s0"/>
      <final state="s1"/>
    </transitions>
  </content>

</automaton>
```

TAF-Kit

```
[vaucanson ~]$ vcsn-b dump-automaton b1 | vcsn-b determinize - \
>  | vcsn-b dot-dump - | dotty -
```



TAF-Kit

```
* Input/output work:
  - define-automaton file: Define an automaton from scratch.
  - display aut: Display ‘aut’.
  - dot-dump aut: Dump dot output of ‘aut’.
  - dump-automaton file: Dump a predefined automaton.
  - edit-automaton file: Edit an existing automaton.
  - identity aut: Return ‘aut’.
  - info aut: Print useful infos about ‘aut’.
  - list-automata: List predefined automata.
* Tests and evaluation on automata:
  - eval aut word: Evaluate ‘word’ on ‘aut’.
  - is-ambiguous aut: Return whether ‘aut’ is ambiguous.
  - is-complete aut: Return whether ‘aut’ is complete.
  - is-deterministic aut: Return whether ‘aut’ is deterministic.
  - is-empty aut: Return whether trimed ‘aut’ is empty.
  - is-realtime aut: Return whether ‘aut’ is realtime.
  - is-standard aut: Return whether ‘aut’ is standard.
```

TAF-Kit

```
* Generic algorithms for automata:
  - accessible aut: Give the maximal accessible subautomaton of ‘aut’.
  - eps-removal aut: Give ‘aut’ closed over epsilon transitions.
  - eps-removal-sp aut: Give ‘aut’ closed over epsilon transitions.
  - co-accessible aut: Give the maximal coaccessible subautomaton of ‘aut’.
  - complete aut: Give the complete version of ‘aut’.
  - concatenate aut1 aut2: Concatenate ‘aut1’ and ‘aut2’.
  - power aut n: Give the power of ‘aut’ by ‘n’.
  - product aut1 aut2: Give the product of ‘aut1’ by ‘aut2’.
  - quotient aut: Give the quotient of ‘aut’.
  - realtime aut: Give the realtime version of ‘aut’.
  - standardize aut: Give the standard automaton of ‘aut’.
  - union-of-standard aut1 aut2: Give the union of standard automata.
  - concat-of-standard aut1 aut2: Give the concatenation of standard automa
  - star-of-standard aut: Give the star of automaton ‘aut’.
  - sum aut1 aut2: Give the sum of ‘aut1’ and ‘aut2’.
  - transpose aut: Transpose the automaton ‘aut’.
  - trim aut: Trim the automaton ‘aut’.
```

La bibliothèque C++

```
* Boolean automaton specific algorithms:
  - complement aut: Complement ‘aut’.
  - determinize aut: Give the determinized automaton of ‘aut’.
  - minimize aut: Give the minimized of ‘aut’ (Hopcroft algorithm).
  - minimize-moore aut: Give the minimized of ‘aut’ (Moore algorithm).
* Conversion between automata and expressions:
  - aut-to-exp aut: Give the automaton associated to ‘aut’.
  - derived-term exp: Use derivative to compute the automaton of ‘exp’.
  - exp-to-aut exp: Alias of ‘standard’.
  - expand exp: Expand ‘exp’.
  - standard exp: Give the standard automaton of ‘exp’.
  - thompson exp: Give the Thompson automaton of ‘exp’.
```

La bibliothèque C++

Pour utiliser Vaucanson, on inclut le(s) header(s) correspondant au(x) type(s) d'automates qu'on veut manipuler, ainsi que les algorithmes qu'on veut utiliser.

```
#include <vaucanson/boolean_automaton.hh>
#include <vaucanson/algorithms/determinize.hh>
#include <vaucanson/algorithms/transpose.hh>
#include <vaucanson/algorithms/complete.hh>
#include <vaucanson/algorithms/minimization_hopcroft.hh>

using namespace vcsn::boolean_automaton;
```

On dispose alors

- de types `automate_t`, `hstate_t`, `alphabet_t`,...

  répondant à des interfaces...
- de macros `for_all_state`, `for_all`,...
- d'algorithmes.

La bibliothèque C++ : algorithmes

```
accessible.hh
aci_canonical.hh
aut_to_exp.hh
backward_realtime.hh
berry_sethi.hh
brzozowski.hh
complement.hh
complete.hh
composition_cover.hh
concatenate.hh
cut_up.hh
derived_term_automaton.hh
determinize.hh
domain.hh
eps_removal.hh
eps_removal_sp.hh
eval.hh
evaluation_fmp.hh
extension.hh
finite_support_conversion.hh
fmp_to_realtime.hh
forward_realtime.hh
image.hh
initial_derivation.hh
invert.hh
is_ambiguous.hh
is_deterministic.hh
is_letterized.hh
is_normalized.hh
isomorph.hh
is_realtime.hh
krat_exp_cderivation.hh
krat_exp_constant_term.hh
krat_exp_derivation.hh
krat_exp_expand.hh
krat_exp_flatten.hh
krat_exp_linearize.hh
krat_exp_partial_derivation.hh
krat_exp_realtime.hh
letter_to_letter_composition.hh
minimization_hopcroft.hh
minimization_moore.hh
normalized_composition.hh
normalized.hh
product.hh
projection.hh
realtime_composition.hh
realtime_decl.hh
realtime.hh
realtime_to_fmp.hh
search.hh
standard.hh
standard_of.hh
sub_automaton.hh
sub_normalize.hh
sum.hh
thompson.hh
transpose.hh
trim.hh
```

La bibliothèque C++ : algorithmes

Chaque algorithme est écrit de manière *générique*. Il sera compilé à la demande pour chaque type d'automate pour lequel il est appelé.

Certains algorithmes sont écrits

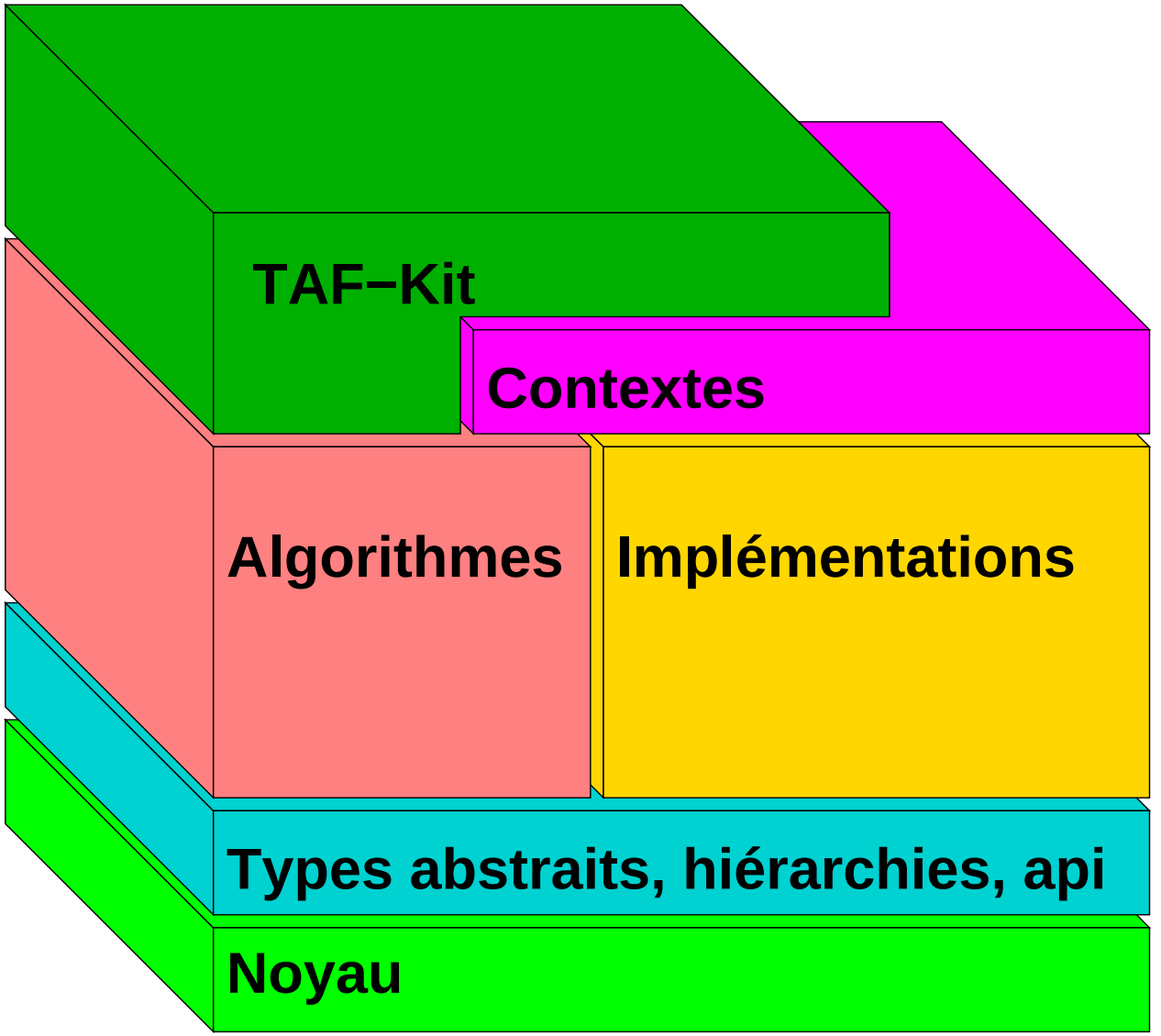
- pour des types théoriques d'automates particuliers;
- pour des implémentations particulières.

La bibliothèque C++ : les contextes

Les types d'automates suivants sont accessibles par défaut:

- automates booléens
- $\mathbb{Z}$-automates
- $\mathbb{R}$-automates
- $\mathbb{Z}$-min-plus-automates
- $\mathbb{Z}$-max-plus-automates
- transducteurs (automates avec sortie)
- transducteurs (automates sur des paires de mots)
- $\mathbb{Z}$-transducteurs

Vaucanson : organisation générale



Générique

Initiation du projet : Jacques Sakarovitch (ENST)

Encadrement: JS, SL, Akim Demaille (LRDE-EPITA), Alexandre Duret-Lutz (LRDE-EPITA)

Réalisation:

15 étudiants au LRDE depuis 2002

Thésards à l'ENST (Rodrigo de Souza, Pierre-Yves Angrand)