Gestion de fichier de configuration par une vue abstraite modifiable

Présentation de mémoire de maitrise
Sherbrooke, Québec, Canada, novembre 2010

Francis Giraldeau <francis.giraldeau@usherbrooke.ca>

Plan

- Introduction 5 min
- Préliminaires 15 min
- XSugar 10 min
- Augeas 10 min
- Conclusions 5 min
- Total 45 min
- Questions, au besoin

Introduction

Fichier de configuration

Fichier de configuration du fond d'écran

```
# absolute path only
[desktop]
background = "/path/to/img.png"
```

Commentaire

Nom de section

Paramètre clé-valeur

Chargement du fond d'écran

```
settings = Ini.load("/home/user/.desktop.conf")
img = settings.get("desktop","background","default.png")
png = Png.load(img)
drawBackgroundImage(png)
```

Fichier smb.conf (Samba)

```
#======================= Global Settings =======================

[global]

## Browsing/Identification ###

# Change this to the workgroup/NT-domain name your Samba server will part of
   workgroup = WORKGROUP

# server string is the equivalent of the NT Description field
   server string = %h server (Samba, Ubuntu)

# Allow users who've been granted usershare privileges to create
# public shares, not just authenticated ones
   usershare allow guests = yes

# Windows clients look for this share name as a source of downloadable
# printer drivers
[print$]
   comment = Printer Drivers
   path = /var/lib/samba/printers
   browseable = yes
   read only = yes
   guest ok = no
```

Fichier fstab

```
# /etc/fstab: static file system information.
#
# Use 'blkid -o value -s UUID' to print the universally unique identifier
# for a device; this may be used with UUID= as a more robust way to name
# devices that works even if disks are added and removed. See fstab(5).
#
# <file system> <mount point>   <type>  <options>             <dump>  <pass>
proc            /proc           proc    defaults              0       0
/dev/sda2       /               ext4    errors=remount-ro     0       1
# swap was on /dev/sda3 during installation
UUID=6c48f137-d79b-41e5-b500-33affa28c10c none   swap    sw   0       0
```

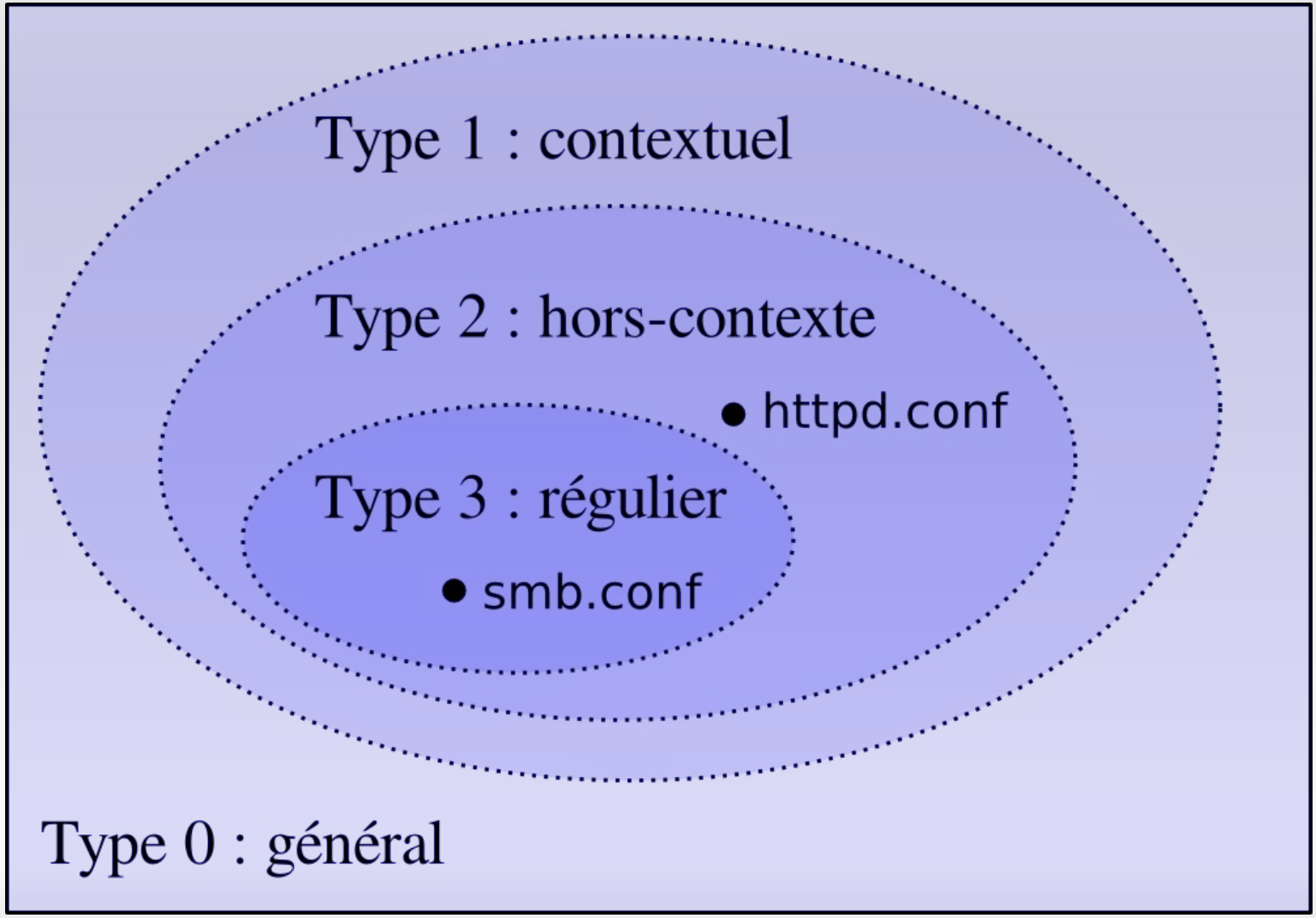
Fichier dhcpd3.conf

```
#
# Sample configuration file for ISC dhcpd for Debian
#

ddns-update-style none;

# option definitions common to all supported networks...
option domain-name "example.org";
option domain-name-servers ns1.example.org, ns2.example.org;

default-lease-time 600;
max-lease-time 7200;

shared-network 224-29 {
  subnet 10.17.224.0 netmask 255.255.255.0 {
    option routers rtr-224.example.org;
  }
  subnet 10.0.29.0 netmask 255.255.255.0 {
    option routers rtr-29.example.org;
  }
  pool {
    allow members of "foo";
    range 10.17.224.10 10.17.224.250;
  }
  pool {
    deny members of "foo";
    range 10.0.29.10 10.0.29.230;
  }
}
```

Fichier httpd.conf (Apache)

```
<VirtualHost *:80>
        ServerAdmin webmaster@localhost
        DocumentRoot /var/www
        <Directory />
                Options FollowSymLinks
                AllowOverride None
        </Directory>
        <Directory /var/www/>
                Options Indexes FollowSymLinks MultiViews
                AllowOverride None
                Order allow,deny
                allow from all
        </Directory>
</VirtualHost>
```

Hiérarchie de Chomsky



Propriétés

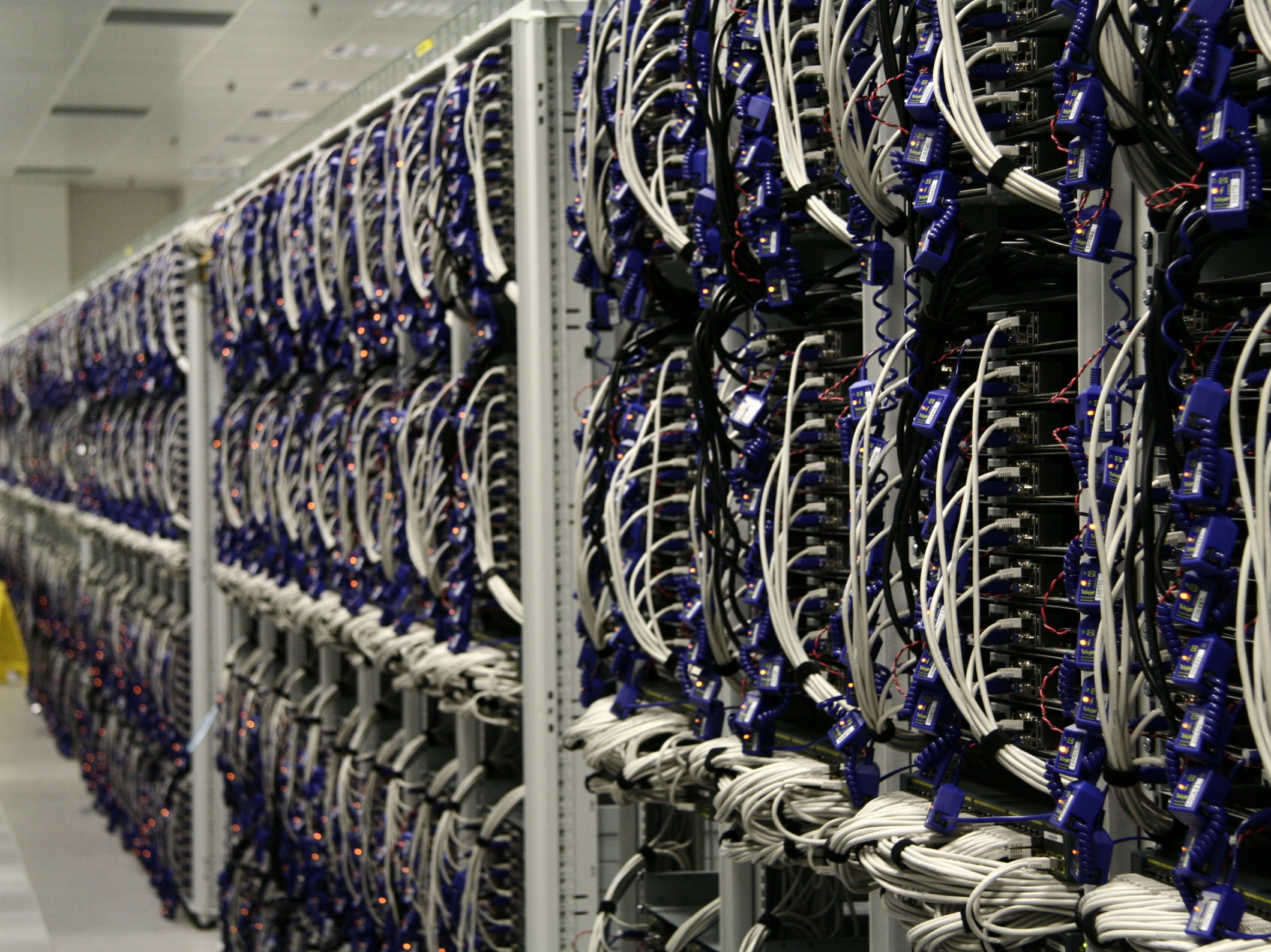
- Type de données: chaine de caractères
  - atoi("911") == 911
- Nécessite un analyseur (*parser*)
- Nombreux formats en usage

Gestion de configuration

- **C**reate : ajouter un paramètre
- **R**ead : lire une valeur existante
- **U**pdate : modifier le fichier
- **D**elete : supprimer un paramètre

Modifier un fichier de configuration revient à modifier une chaine de caractère

Forme d'interaction homme-machine

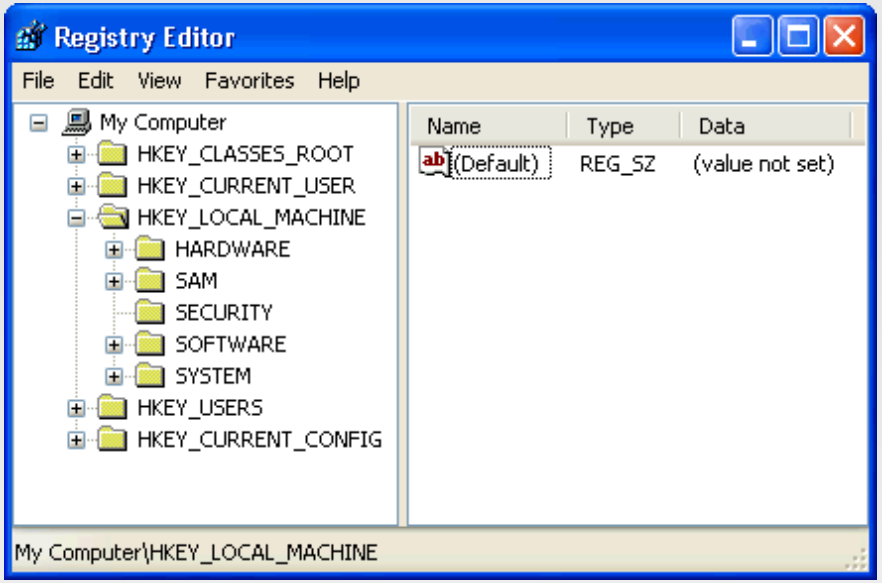


Automatisation

- Base de données
- Scripts
- Gabarits
- Bibliothèques

Base de données

- API uniforme
- Données typées
- Abstraction de la couche de persistance
- Registre Windows
- Elektra



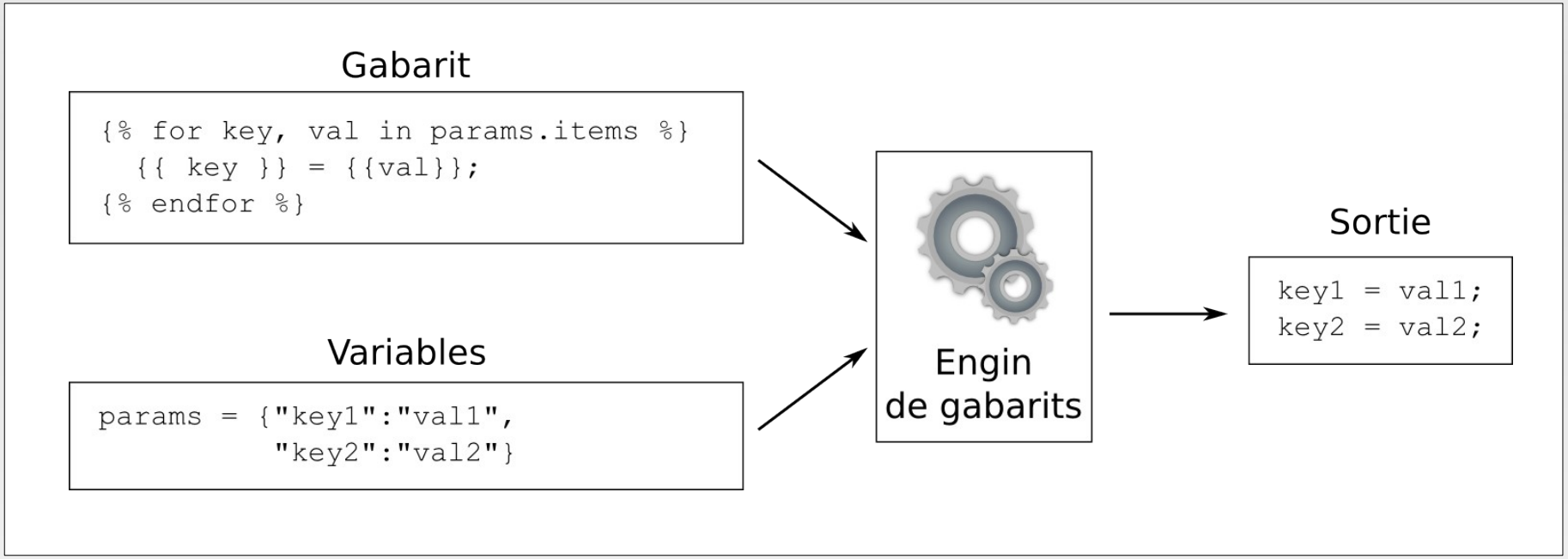
Approche top-down: nécessite la modification de toutes les applications!


Coins & Cards

Scripts

- Analyser et modifier la chaine *in situ*
- awk, sed, perl, grep, tail, head, cat, echo, ...
- Propice aux erreurs
- Indepotence hardue
- Difficile à tester et à maintenir
- Peu réutilisable

Gabarits



- Écriture seulement
- Sous-ensemble du langage

Bibliothèque spécialisée

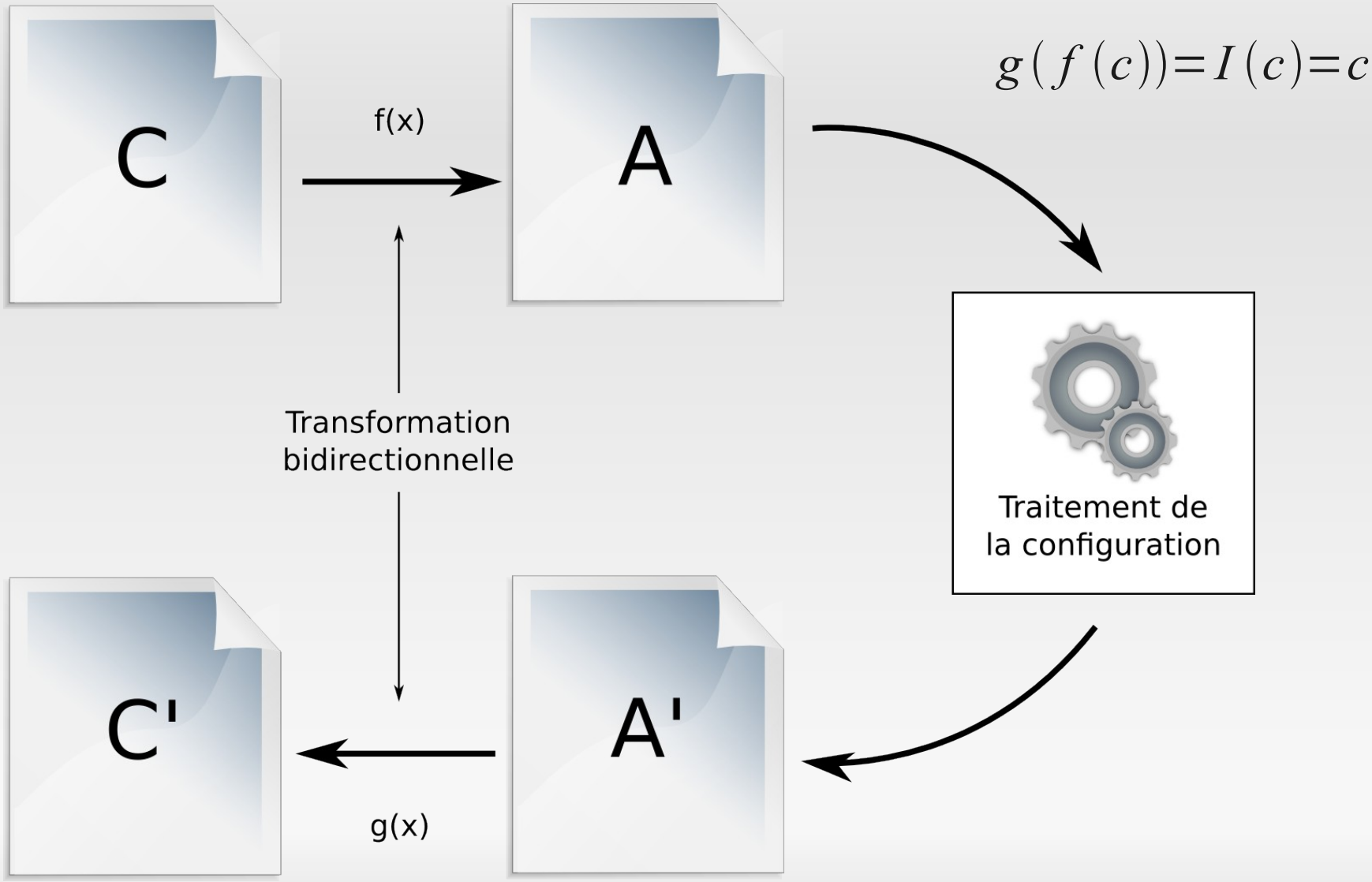
```
public class TestINI {
    @Test
    public void testIniLoad() throws BackingStoreException,
                    InvalidIniFormatException,
                    FileNotFoundException,
                    IOException {
        Ini ini = new Ini();
        ini.load(new FileReader("src/ca/udes/input.ini"));
        FileWriter w = new FileWriter("src/ca/udes/output.ini");
        ini.store(w);
    }
}
```

```
$ diff -ru input.ini output.ini
--- input.ini   2010-01-25 13:49:03.631307148 -0500
+++ output.ini  2010-01-25 13:50:03.844053328 -0500
@@ -1,4 +1,3 @@
-; commentaire
 [section1]
+key1 = value1

-    key1 = value1
```

Propriétés désirées

- Modifier la chaine sur place
- Modification minimale
- API uniforme pour tous les formats
- Garantie de fonctionnement: validation statique

Principe d'opération



Vue abstraite

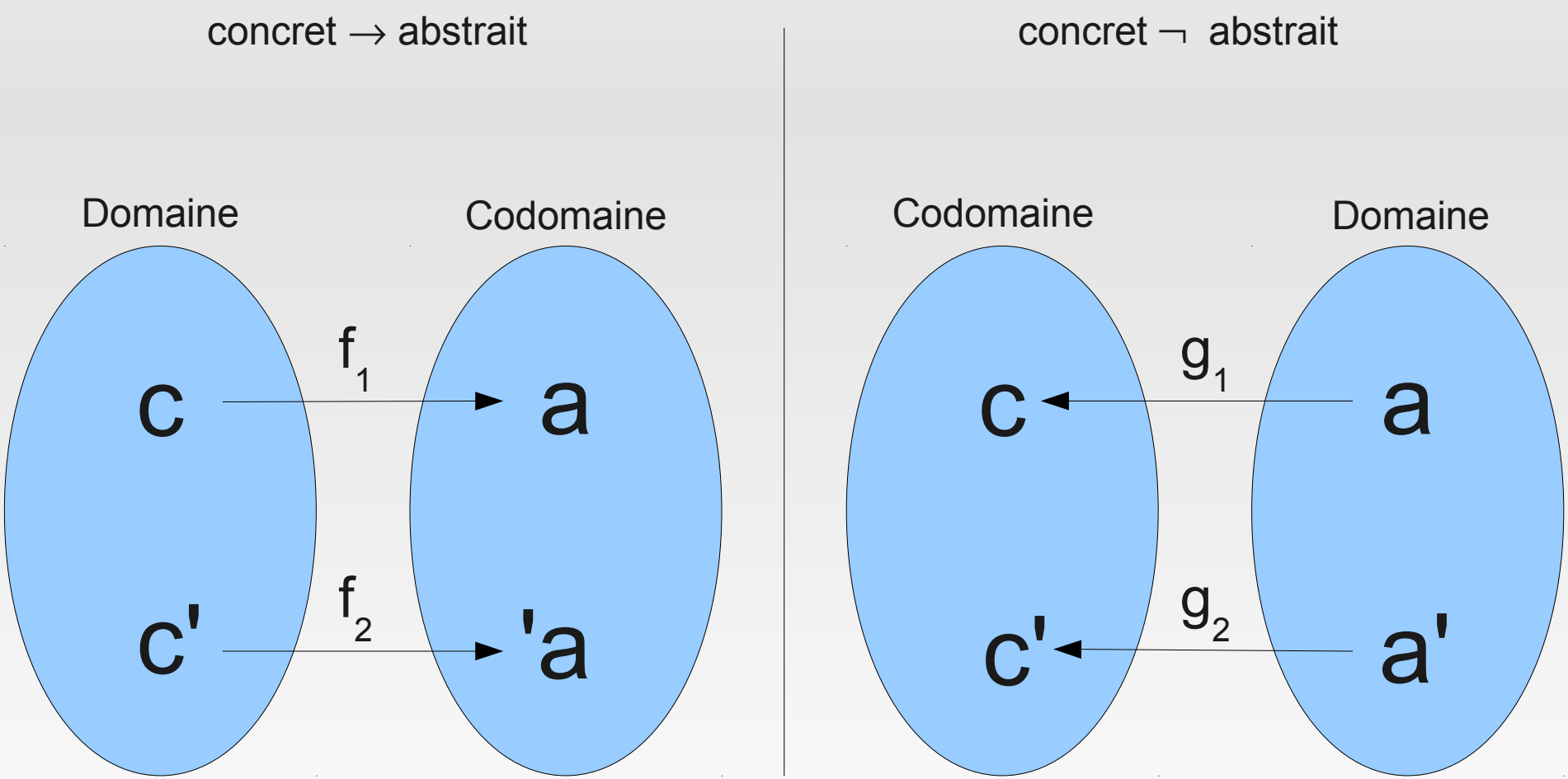
- Masquer des caractères
  - Espace, formatage, information redondante..
- Capturer et structurer la chaine
- Structure en arbre

Albert.....1905\n
Roger.....1936\n
Francis...1982\n

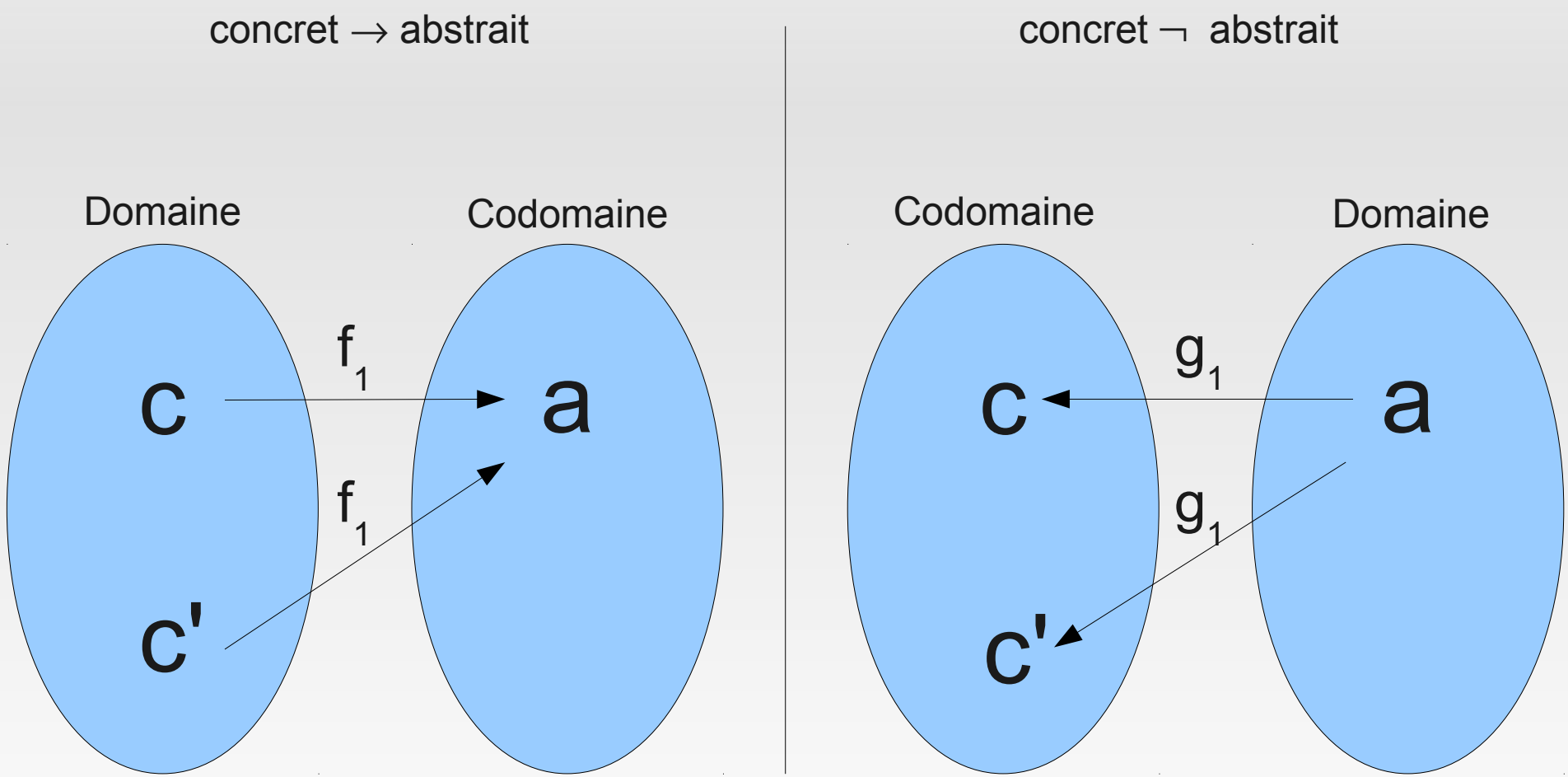
```
<liste>
  <personne>
    <nom>Albert</nom>
    <annee>1905</annee>
  </personne>
  ...
</liste>
```

set /liste/personne[nom="Albert"]/annee = "1879"

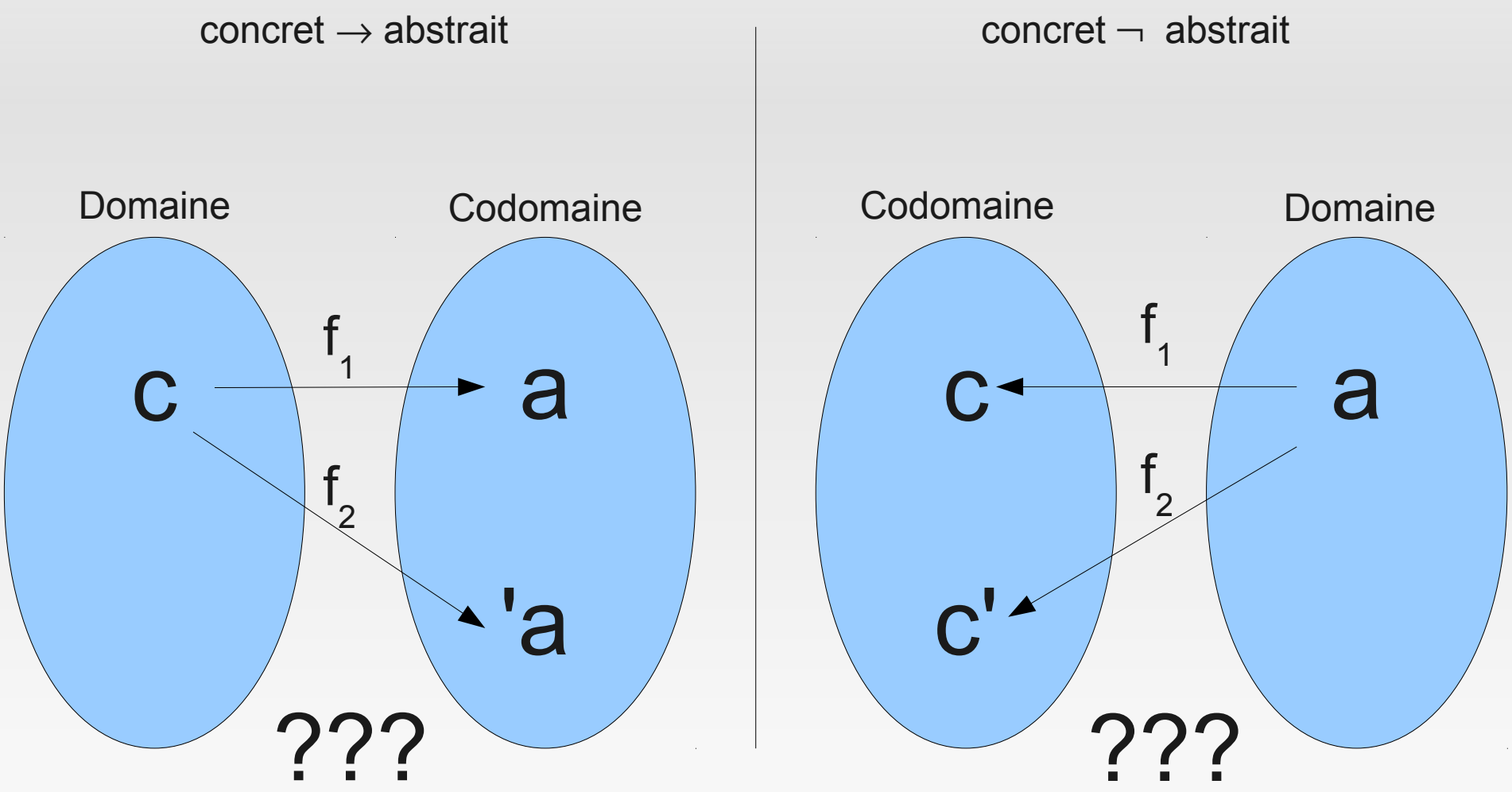
Relation bidirectionnelle



Relation bidirectionnelle



Relation bidirectionnelle



Relation bidirectionnelle

- Transformations bijectives
  - La transformation et son inverse doit s'appliquer
- Toute ambigüité doit être évitée
  - Intuitivement: si plusieurs transitions s'appliquent, laquelle choisir?

Préliminaires

Langage régulier

- Reconnus par une machine à états finie (DFA)
- Représentation compacte sous forme d'expression régulière (POSIX)
- Construction de Thompson
- déterministe + minimale = unique

Notation POSIX

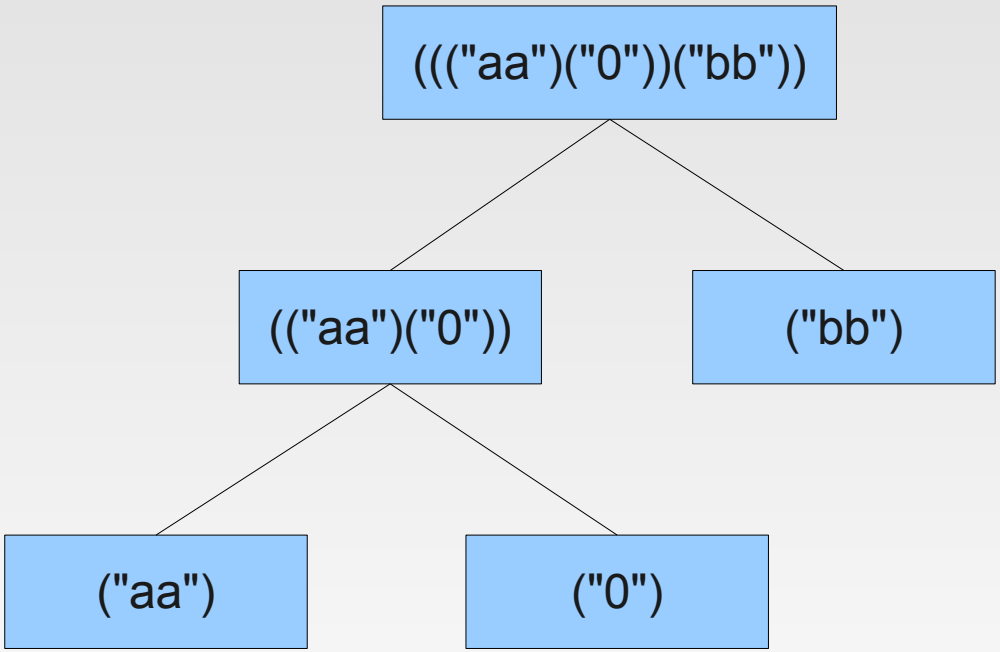
- Intervalle de caractères : [a-z]
- zéro ou un : ?
- un ou plusieurs : +
- zéro ou plusieurs (Kleene) : *
- union : |
- concaténation : juxtaposition
- complément : ^
- groupe de capture : ()

Exemples POSIX

- [a-z]
  - {"a","b","c", ... }
- [a-z]+
  - { "a", "aa", "ab", "b", "ba", "bb", ...}
- [a-z]+[0-9][a-z]+
  - { "a0a", "aa0aa", ... }

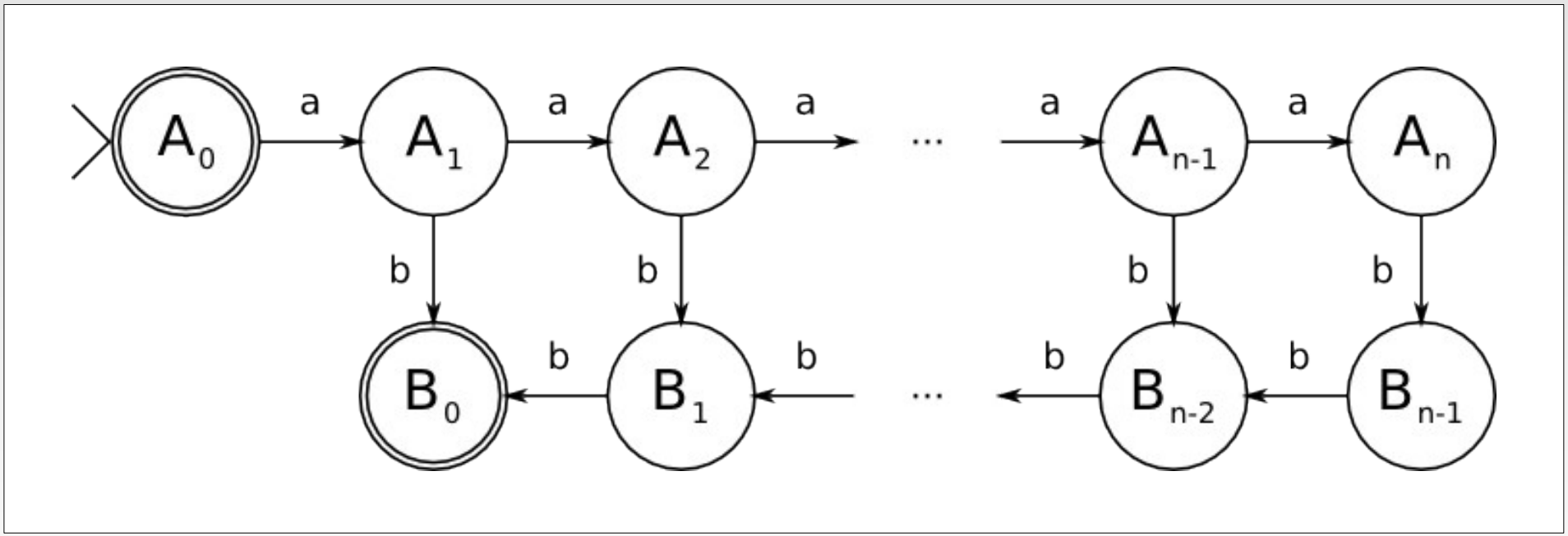
Exemple de groupes de capture

- ((([a-z]+)([0-9]))([a-z]+)) avec "aa0bb"



Limitation des DFA

$$L = \{ a^i b^i | i \geq 0 \}$$



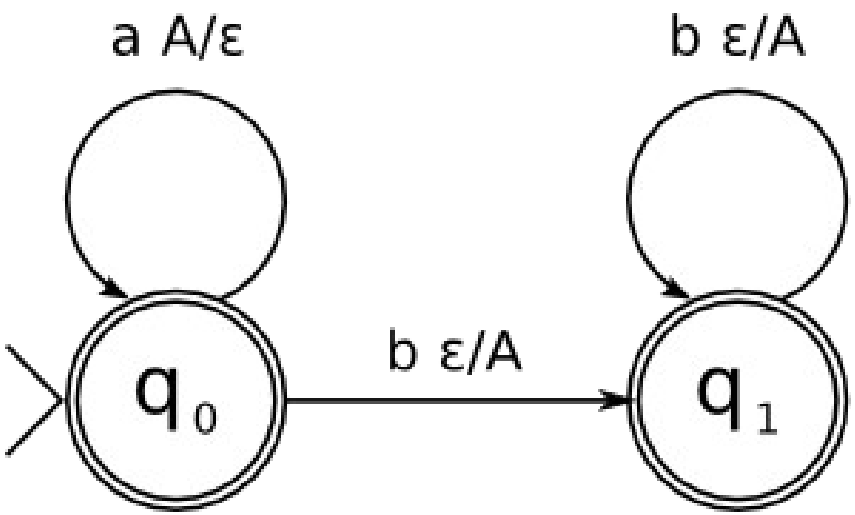
Langage hors contexte

- Reconnus par une machine à pile
- Représentation sous forme de grammaire hors contexte

Exemple de machine à pile

$$L = \{ a^i b^i | i \geq 0 \}$$

$S \rightarrow a\ S\ b \mid \varepsilon$



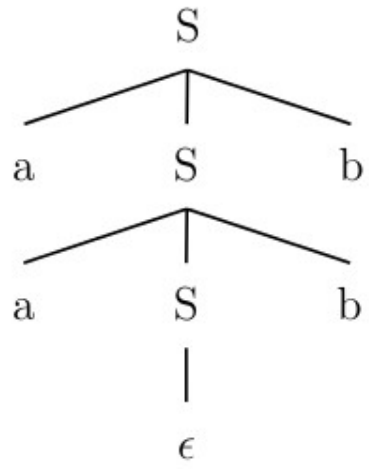
Pile	État	Entrée
{}	q_0	aaabbb
{A}	q_0	aabbb
{AA}	q_0	abbb
{AAA}	q_0	bbb
{AA}	q_1	bb
{A}	q_1	b
{}	q_1	ϵ

Analyseur syntaxique

- Reconstruit la structure d'une chaine
- LR(k) : grammaire h. c. déterministe O(n)
- Earley : grammaire h. c. quelconque $O(n^3)$

$S \rightarrow a\ S\ b \mid \varepsilon$

entrée: "aabb"



Sources d'ambigüités

- Ambigüité des groupes de captures des expressions régulières
- Langages réguliers
- **Décidable**

- Ambigüité des grammaires hors contexte
- Langages hors contextes
- **Non-décidable**

Ambigüité de capture

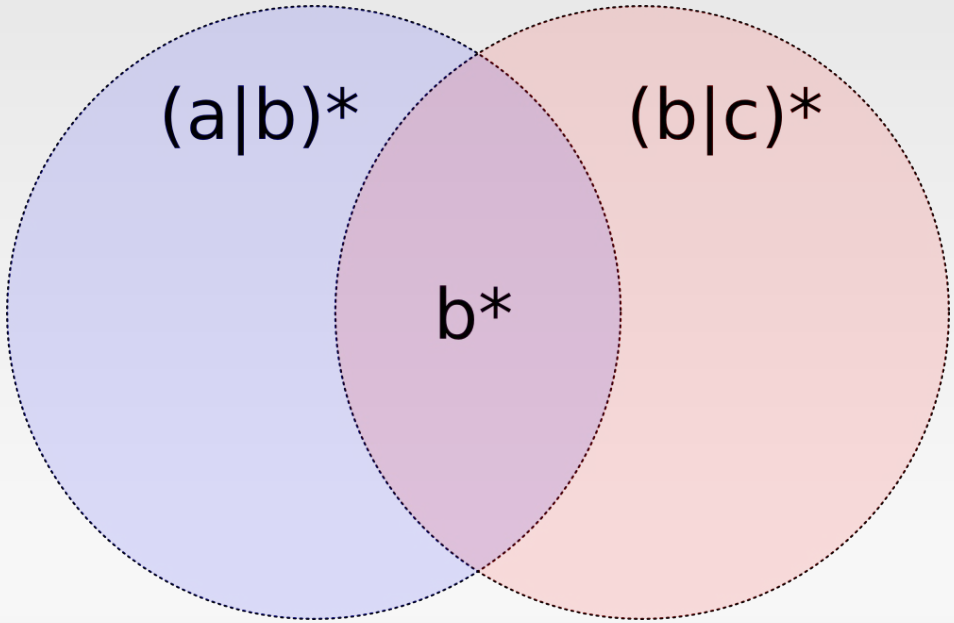
- Concaténation:
  - (a*)(a*) = a*
  - (a*)(a*) avec "aaa" = ?

$$L = \{ a^i | i \geq 0 \}$$

(a*)...	...(a*)
ε	aaa
a	aa
aa	a
aaa	ε

Ambigüité de capture (suite)

- Union:
  - (a|b)*|(b|c)* avec "ba" = ("ba")()
  - (a|b)*|(b|c)* avec "bb" = ("bb")() ou ()("bb") ???



Détection

- Concaténation: opérateur de chevauchement
- Répétition Kleene : cas particulier de concaténation
- Union: opérateur d'intersection

Opérateur de chevauchement

$$X \bowtie Y = xvy | x, y \in \Sigma^* \wedge v \in \Sigma^+ \wedge x, xv \in X \wedge vy, y \in Y$$
$$X \bowtie Y = \varnothing$$

- Il ne doit pas être possible de scinder la chaine de plusieurs manières
- (x[a]?) chevauche ([a]?y) = [a]
  - ("xa")("y") ou ("x")("ay")

Répétition Kleene: $l \bowtie l* = \emptyset$

Intersection

$$X \cap Y = \overline{\overline{X} \cup \overline{Y}}$$

$$X \cap Y = \emptyset$$

- Il ne doit pas y avoir de chaine commune entre les langages X et Y
- Obtenu par le complément de l'union des compléments de X et Y
- Langages réguliers clos pour le complément

Exemple d'intersection

$$\mathcal{L}_1 = (a|b)^* \qquad \mathcal{L}_2 = (b|c)^*$$

$$\begin{aligned}
\mathcal{L}_1 \cap \mathcal{L}_2 &= \overline{\overline{\mathcal{L}_1} \cup \overline{\mathcal{L}_2}} \\
&= \overline{\overline{(a|b)^*} \cup \overline{(b|c)^*}} \\
&= \overline{(a|b)^*(c)(a|b|c)^* \cup (b|c)^*(a)(a|b|c)^*} \\
&= \overline{b^*(a|c)(a|b|c)^*} \\
&= b^*
\end{aligned}$$

Ambigüité des grammaires h. c.

$$G = (V, \Sigma, P, S) \qquad w \in \Sigma^* \qquad S \overset{*}{\Rightarrow}_L w$$

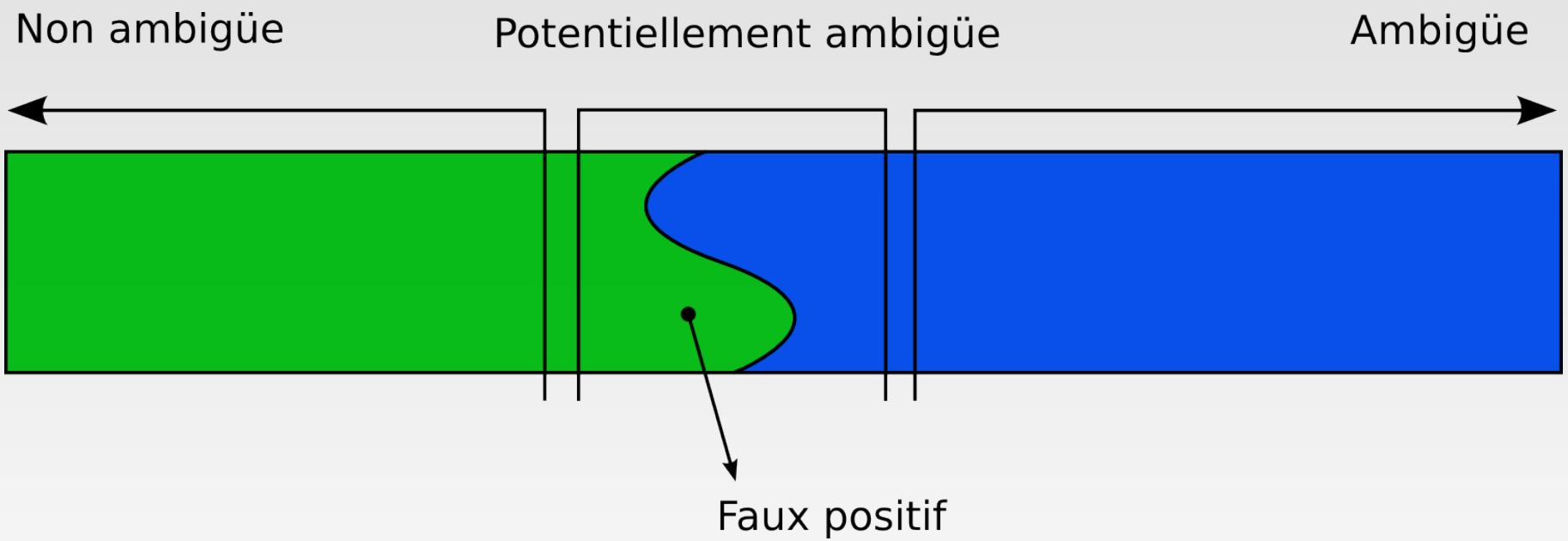
- Problème non décidable
- Propriété de clôture des opérateurs d'intersection et de complément
- Problème de postcorrespondance

$$L_1 = \{a^i b^i c^j | i, j \geq 0\}$$ hors contexte

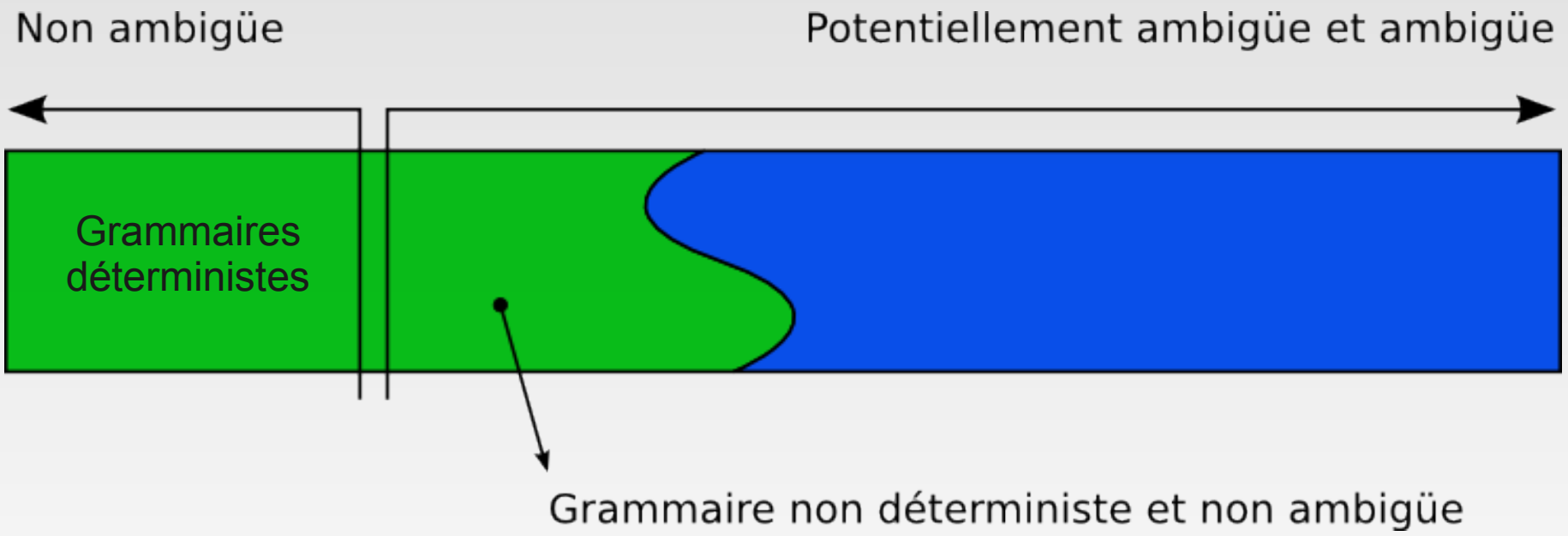
$$L_2 = \{a^j b^i c^i | i, j \geq 0\}$$ hors contexte

$$L_1 \cap L_2 = \{a^i b^i c^i | i \geq 0\}$$ contextuel

Détection par approximation



Critère LR(k)


Non ambigüe
Potentiellement ambigüe et ambigüe
Grammaires déterministes
Grammaire non déterministe et non ambigüe

Ambigüité et déterminisme

- déterministe → non ambigu
- ambigu → non déterministe
- non déterministe → ambigu ou non

Exemple critère LR(k)

$$L = c^+ a \mid c^+ b$$

Nécessite k = ∞

Grammaire non déterministe et non ambigüe

$$\begin{aligned}
&S \to A\ C\ a \\
&S \to B\ D\ b \\
&A \to c \\
&B \to c \\
&C \to C\ c \\
&C \to \epsilon \\
&D \to D\ c \\
&D \to \epsilon
\end{aligned}$$

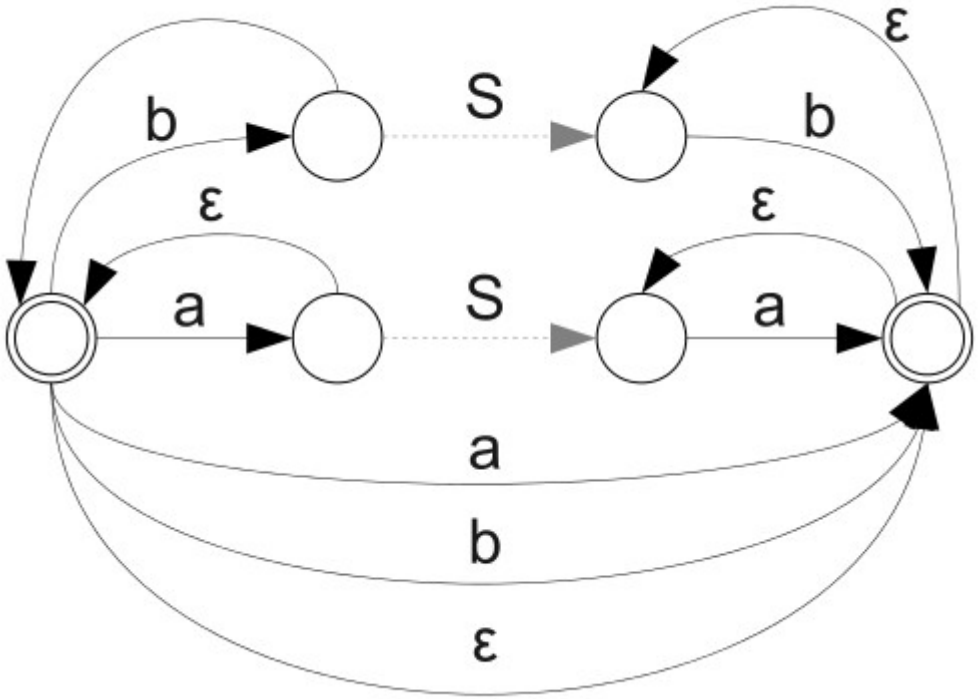
Approximation régulière

$$\mathcal{L}_G(\alpha) \subseteq \mathcal{A}_G(\alpha) \; \forall \; \alpha \in \{V, \Sigma\}^*$$

- Surensemble propre: préserve les ambigüités
- Peut ajouter des ambigüités qui ne sont pas dans le langage d'origine : faux positifs
- Plus précis que LR(k)

Réseau de transition récursif

$$S \rightarrow aSa \mid bSb \mid a \mid b \mid \varepsilon$$



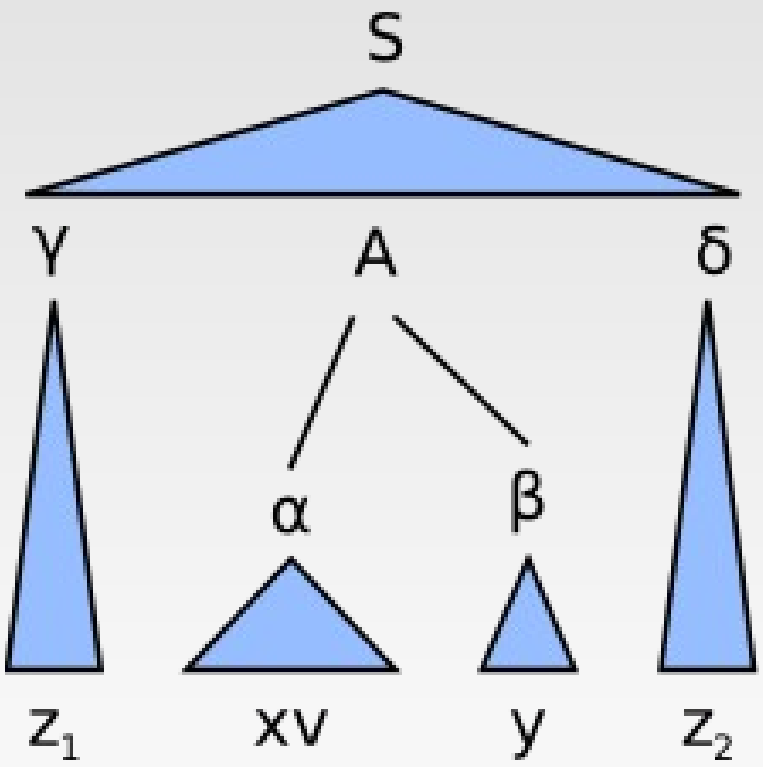
S → a(a|b)*a
S → b(a|b)*b
S → a
S → b
S → ε

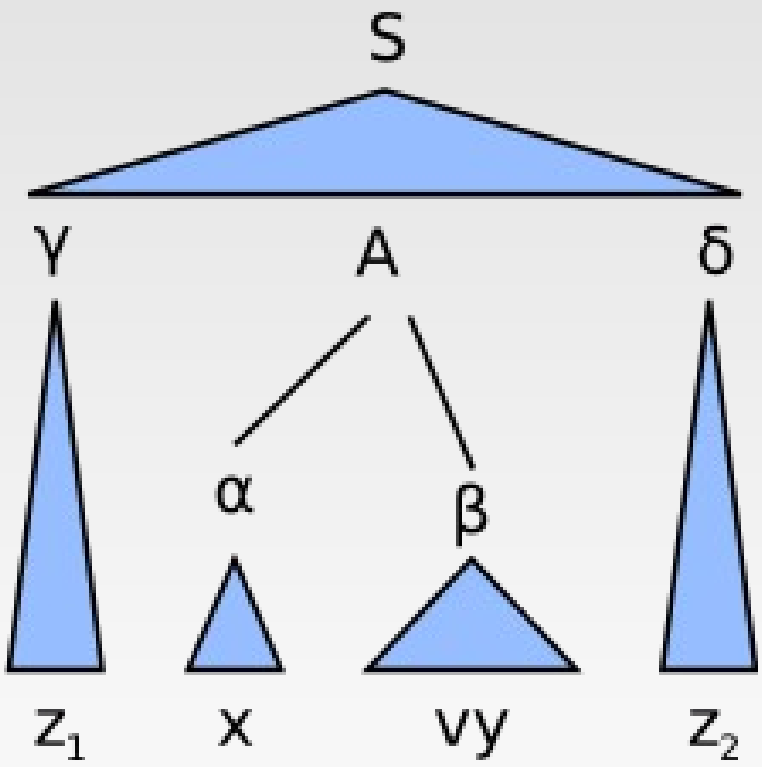
Division du problème

- Ambigüité horizontale
- Ambigüité verticale
- Définitions équivalentes
- Ne rend pas le problème décidable

Ambigüité horizontale

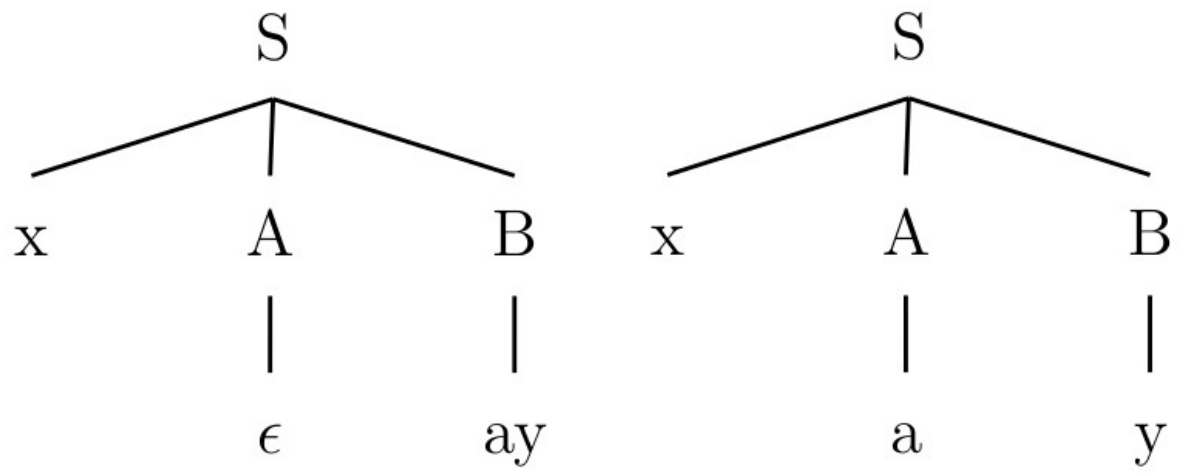
$$\exists A \in V, (A \to \alpha\beta) \in P : \mathcal{L}_G(\alpha) \bowtie \mathcal{L}_G(\beta) \neq \emptyset$$





Exemple ambigüité horizontale

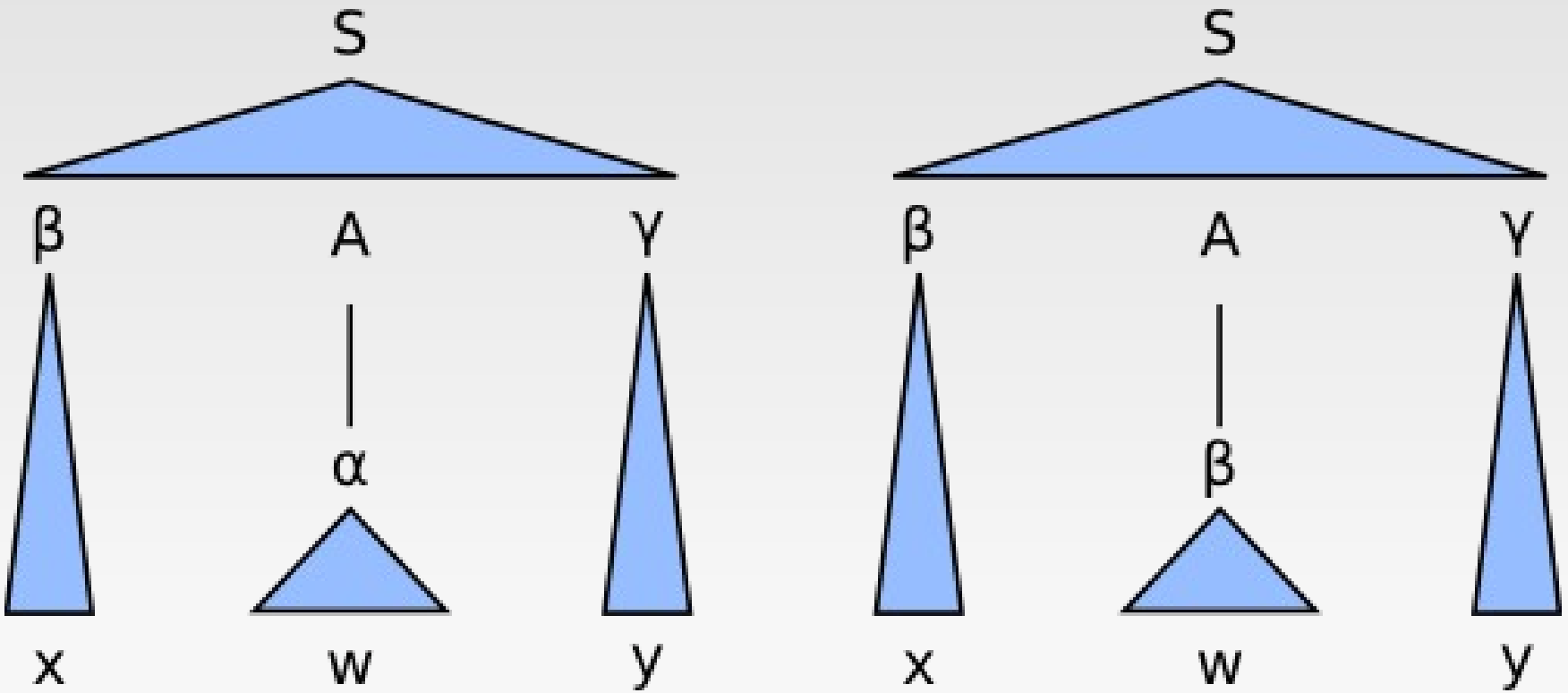
$$S \rightarrow x\ A\ B$$
$$A \rightarrow a$$
$$A \rightarrow \epsilon$$
$$B \rightarrow ay$$
$$B \rightarrow y$$



avec "xay"

Ambigüité verticale

$$\exists A \in V, (A \to \alpha), (A \to \beta) \in P, \alpha \neq \beta : \mathcal{L}_G(\alpha) \cap \mathcal{L}_G(\beta) \neq \emptyset$$



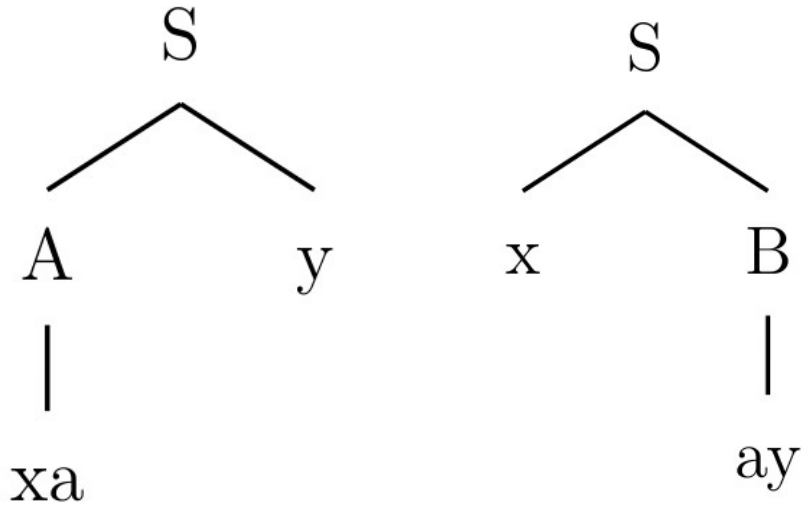
Exemple ambigüité verticale

$S \rightarrow A\ y$

$S \rightarrow x\ B$

$A \rightarrow xa$

$B \rightarrow ay$



avec "xay"

Exemple synthèse (horizontale)

Règle	Approximation
(1) S $\rightarrow$ a S a	$a(a\|b)^*a$
(2) S $\rightarrow$ b S b	$b(a\|b)^*b$
(3) S $\rightarrow$ a	a
(4) S $\rightarrow$ b	b
(5) S $\rightarrow$ ϵ	ϵ

Production	Préfixe	Suffixe	Chevauchement
(1)	a	S a	$a \bowtie (a\|b)^*a = \emptyset$
(1)	a S	a	$a(a\|b)^* \bowtie a = \emptyset$
(2)	b	S b	$b \bowtie (a\|b)^*b = \emptyset$
(2)	b S	b	$b(a\|b)^* \bowtie b = \emptyset$

Exemple synthèse (verticale)

Règle	Approximation
(1) S → a S a	$a(a\|b)^*a$
(2) S → b S b	$b(a\|b)^*b$
(3) S → a	a
(4) S → b	b
(5) S → ϵ	ϵ

Productions	Intersection
(1,2)	$a(a\|b)^*a \cap b(a\|b)^*b = \emptyset$
(1,3)	$a(a\|b)^*a \cap a = \emptyset$
(1,4)	$a(a\|b)^*a \cap b = \emptyset$
(1,5)	$a(a\|b)^*a \cap \epsilon = \emptyset$
(2,3)	$b(a\|b)^*b \cap a = \emptyset$
(2,4)	$b(a\|b)^*b \cap b = \emptyset$
(2,5)	$b(a\|b)^*b \cap \epsilon = \emptyset$
(3,4)	$a \cap b = \emptyset$
(3,5)	$a \cap \epsilon = \emptyset$
(4,5)	$b \cap \epsilon = \emptyset$

XSugar

Principe de fonctionnement

$$S \rightarrow \alpha \Leftrightarrow \alpha'$$

- Correspondance entre deux grammaires
- Analyse avec une grammaire (Earley)
- Sérialisation de l'arbre syntaxique avec sa contrepartie
  - Parcourir l'arbre en postordre et concaténer les symboles terminaux

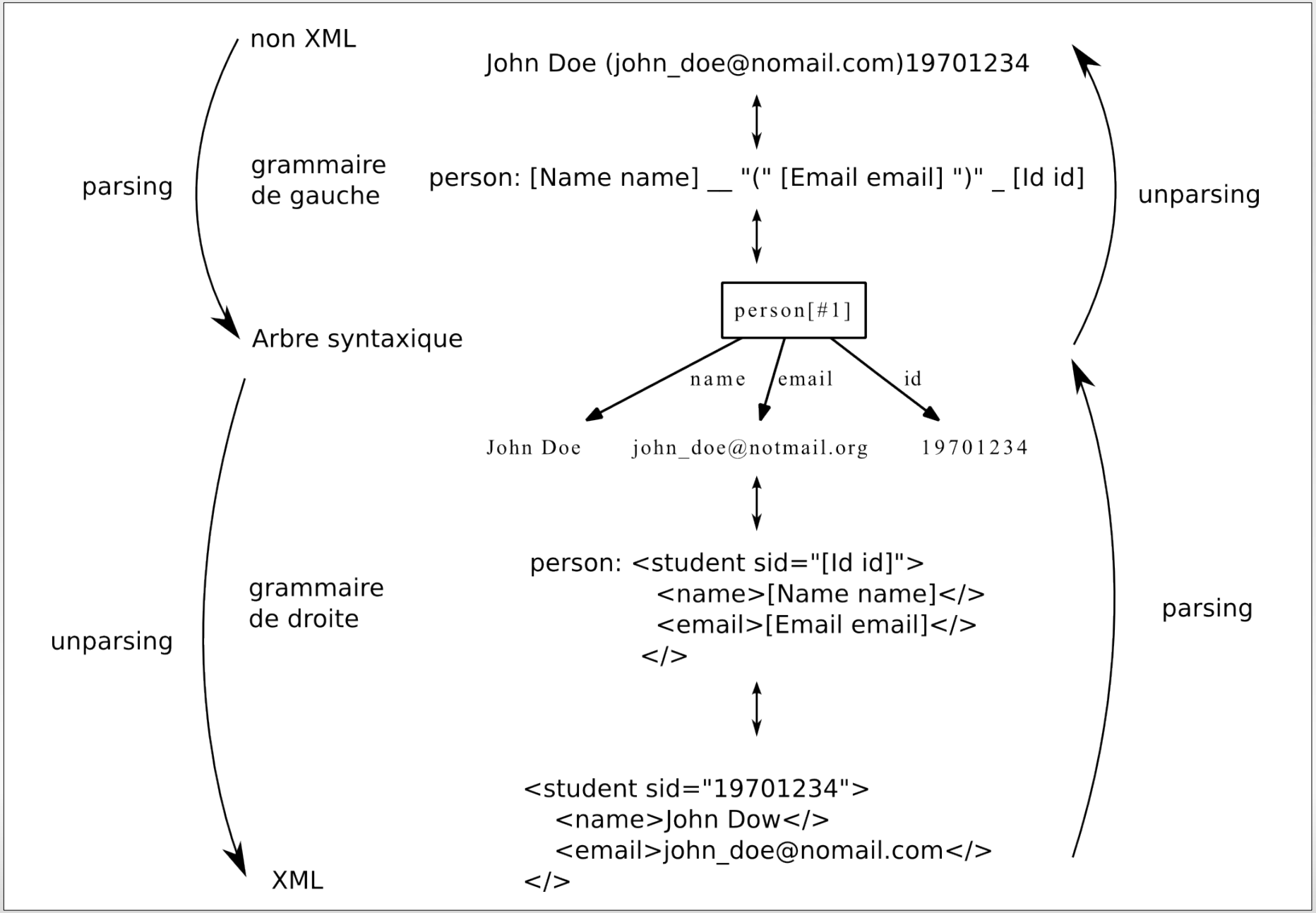
Feuille de style

```
n : [xs x_s] [ys y_s] = <A> [ys y_s] [xs x_s]</>

xs : "x" [xs x_s] = <X></> [xs x_s]
   : "x"          = <X></>

ys : "y" [ys y_s] = <Y> [ys y_s] </>
   : "y"          = <Y></>
```

"xxyyy" ↔

```
<?xml version="1.0" encoding="UTF-8"?>
<A>
  <Y>
    <Y>
      <Y/>
    </Y>
  </Y>
  <X/>
  <X/>
</A>
```

Principe de fonctionnement (suite)



Vérification statique

- Ambigüité verticale et horizontale
- Grammaire de gauche et de droite
- Si aucune ambigüité, alors réversibilité garantie

Limitation

- Pertes de caractères : terminaux d'expression régulière sans correspondance

```
X = /[x]+/
S : "a" [X] = <a></>

"axx" → <a></> → "ax"
```

Le nombre exact de "x" est perdu lors d'un aller-retour

Bidirectionnalité stricte

- Modification dynamique de la feuille de style

```
X = /[x]+/
S >: "a" [X x] = <a>
                   <strict>[X x]</>
                 </>
   : "a" [X] = <a></>
```

```
"axx" → <a><strict>xx</></> → "axx"
```

Le nombre exact de "x" est préservé

Résultats

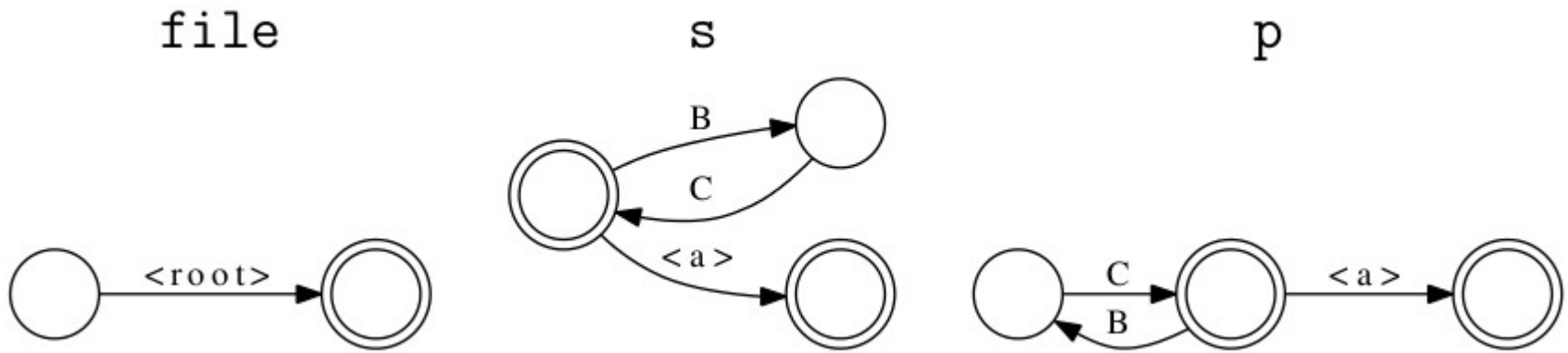
- Testé avec 10 feuilles de style
- 9/10 fonctionnent
- Perte de caractère pour le contenu mixte

```
<root><a>.x.y.</a><b>.p.r.<c></c></b><root>

<root>\n
....<a>.x.y.</a>\n
....<b>\n
.........p.q.\n
........<c></c>\n
....</b>\n
<root>
```

getTextTrim()

Détection du contenu mixte

```
xmlns = "http://example.com/"

B = [b]+
C = [c]+

file : [s ss] = <root> [s ss] </>

s : [p pp] = <a> [p pp] </>
  : [B b] [p pp] = [B b] [p pp]
  : =

p : [C c] [s ss] = [C c] [s ss]
```



Conclusion

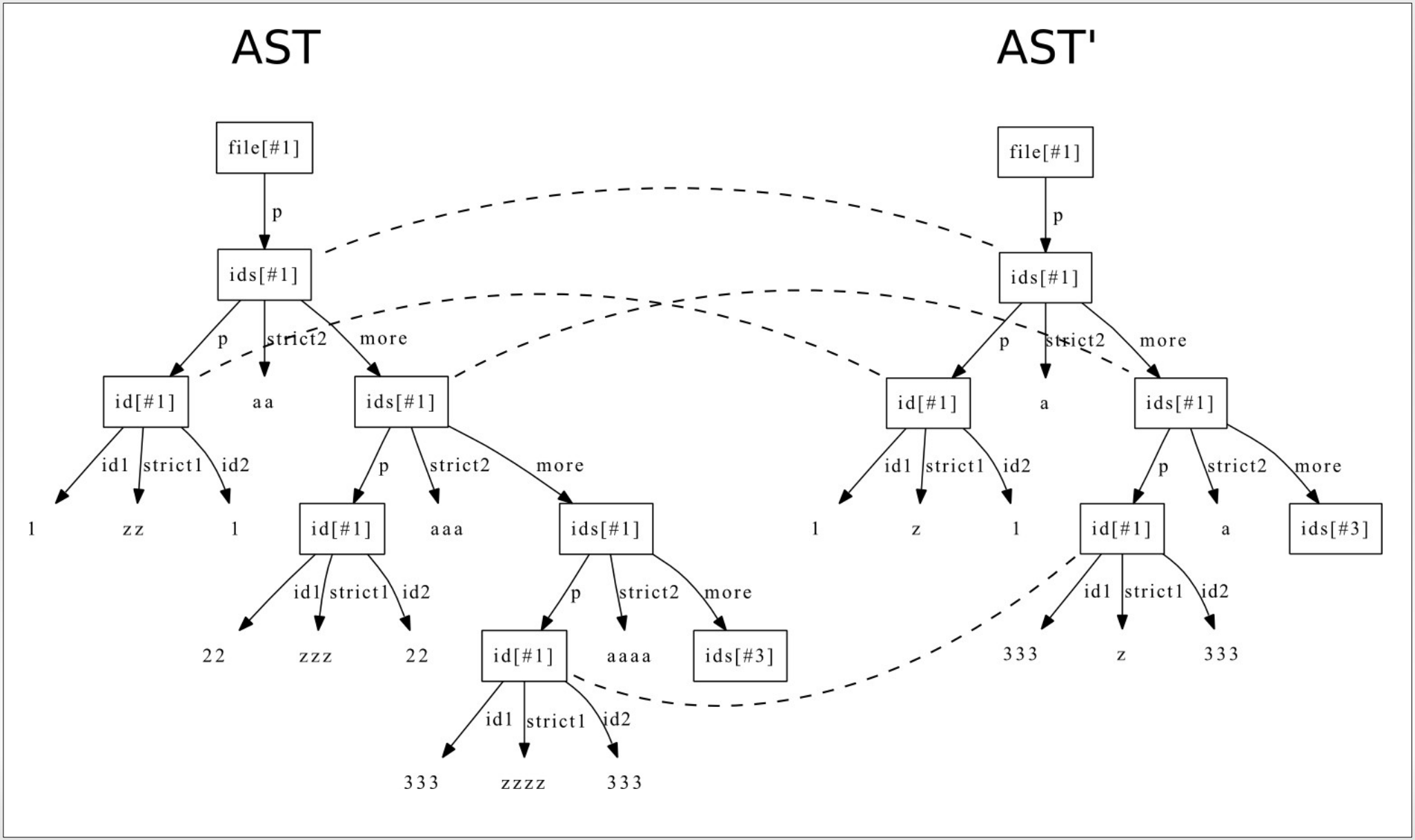
- Bidirectionnalité stricte atteinte
- Diminution de l'abstraction
- Modification difficile du XML

Fusion des arbres syntaxiques

$$non\ XML \overset{G_O}{\rightarrow} AST_O \overset{D_O}{\rightarrow} XML \overset{mod}{\rightarrow} XML'$$
$$XML' \overset{D_O}{\rightarrow} AST'_O \overset{G_O}{\rightarrow} non\ XML' \overset{G_S}{\rightarrow} AST'_S$$
$$non\ XML \overset{G_S}{\rightarrow} AST_S$$

$$AST_s \rightarrow AST_s'$$

Algorithme


AST
AST'
file[#1]
p
ids[#1]
p
strict2
more
id[#1]
aa
ids[#1]
id1
strict1
id2
1
zz
1
p
strict2
more
id[#1]
aaa
ids[#1]
id1
strict1
id2
22
zzz
22
p
strict2
more
id[#1]
aaaa
ids[#3]
id1
strict1
id2
333
zzzz
333
file[#1]
p
ids[#1]
p
strict2
more
id[#1]
a
ids[#1]
id1
strict1
id2
1
z
1
p
strict2
more
id[#1]
a
ids[#3]
id1
strict1
id2
333
z
333

Résultats

- Testé avec 9 feuilles de styles
- 5 scénarios de modification du XML
  - Ajout, modification, suppression, déplacement, ...
- Modification sécuritaire
- Problème d'alignement

Entrée	Vue abstraite	Vue abstraite modifiée	Sortie
C 1	C	A	A 1
B 2	B	B	B 2
A 3	A	C	C 3

Conclusion

- Meilleure abstraction
- Recouvrement efficace
- Alignement par séquence

Augeas

Principe de fonctionnement

- Basé sur des lentilles
  - Combinaison de transformations complémentaires

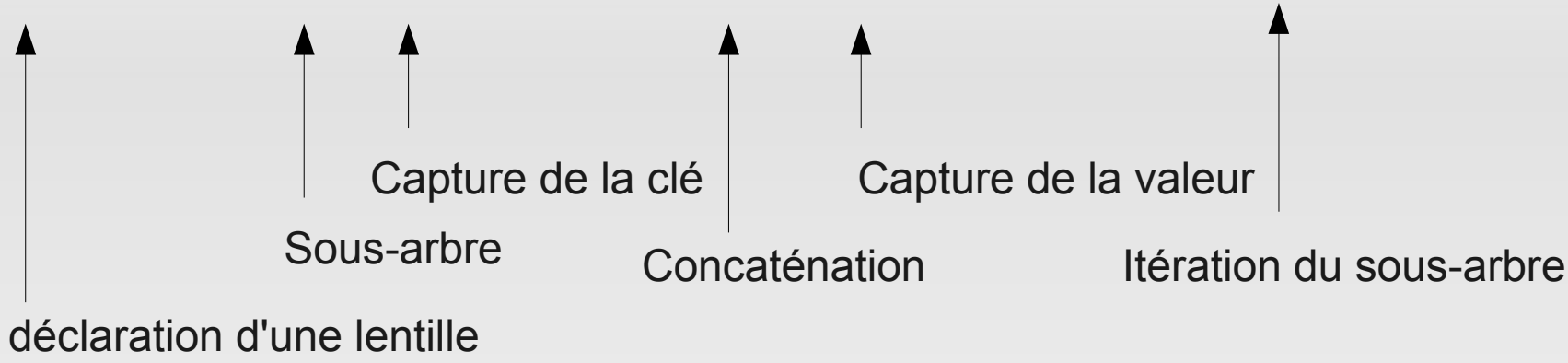
$$l.get \in C \to A \qquad l.put(l.get(c)) = c$$

$$l.put \in A \times C \to C \qquad l.get(l.put(a, c)) = a$$

$$l.create \in A \to C \qquad l.get(l.create(a)) = a$$

Principe de fonctionnement (suite)

```
let kv = [ key /[a]+/ . store /[b]+/ ] *
```




```
kv.get("abaabb") = {"a"="b"} {"aa"="bb"}

kv.put({"a"="b"} {"aa"="bb"}) = "abaabb"
```

```
struct tree {
    char *label;
    char *value;
    struct list *children;
}
```

Lentilles primitives

- `key regexp`
- `label string`
- `store regexp`
- `value string`
- `del regexp string`
- `counter string`
- `seq string`

- `key /[a]+/`
- `label "x"`
- `store /[b]+/`
- `value "y"`
- `del /[ \n]+/ " "`
- `counter "record"`
- `seq "record"`

Combinaison

- l_1 . l_2
- l_1 | l_2
- l*, l+, l?
- [l]

- Concaténation
- Union
- Répétition
- Sous-arbre

Lentille récursive

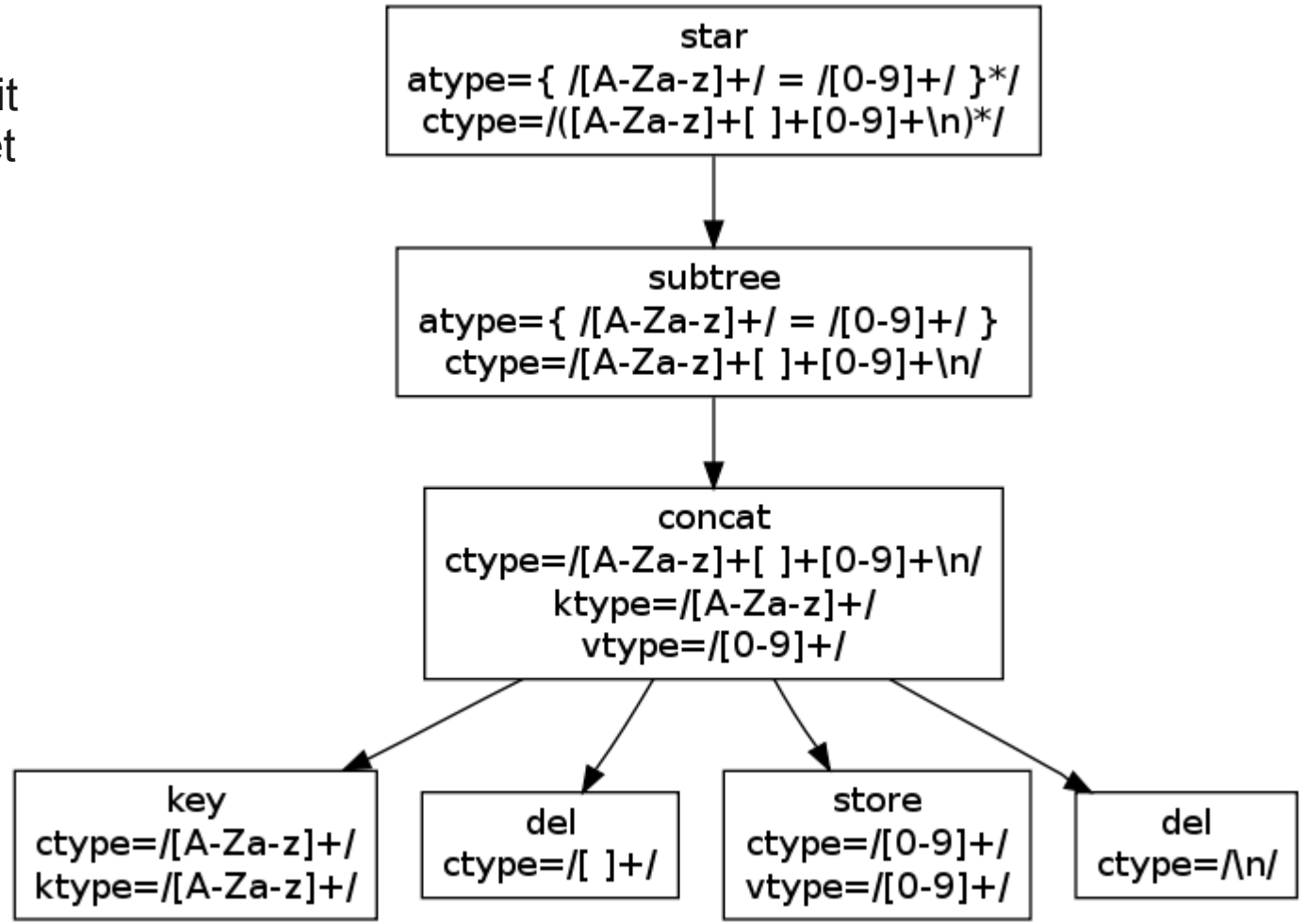
- let ab = [key "a" . store "b"]+
- let rec ab = [key "a" . store "b"] . ab?
- let rec ab = [key "a" . ab . store "b"]?
  - "aabb" = { "a" = "b" { "a" = "b" } }

$$L = \{ a^i b^i | i \geq 0 \}$$

Hiérarchie des lentilles

```
let kv = [ key /[a-zA-Z]+/ . del /[ ]+/ " " .
           store /[0-9]+/ . del "\n" "\n" ]*
```

atype : abstrait
ctype : concret
ktype : clé
vtype : valeur



Limitation : langages balisés (1)

```
module Xmlprob =

let dels (s:string) = del s s
let content = store /[a-z]*/
let xml1 = [ dels "<" . key /[a-z]+/ . dels ">" .
             content .
             dels "</" . del /[a-z]+/ "x" . dels ">" ] *

test xml1 get "<a>yyy</a>" = { "a" = "yyy" }
test xml1 get "<a>yyy</b>" = { "a" = "yyy" }
test xml1 put "" after set "/a" "yyy"   = "<a>yyy</x>"
```

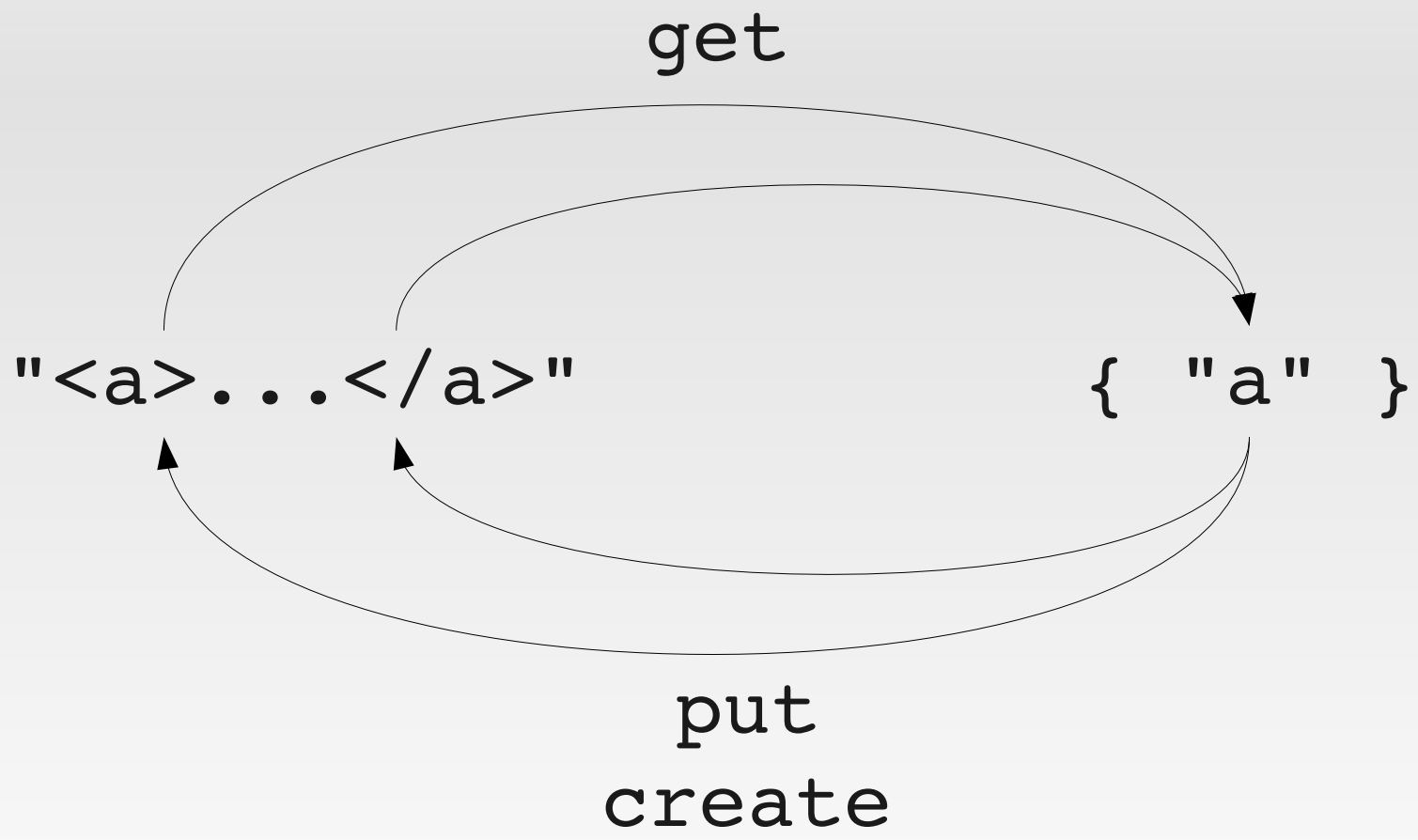
- Accepte des chaines invalides
- Problème avec la fonction "create"

Limitation : langages balisés (2)

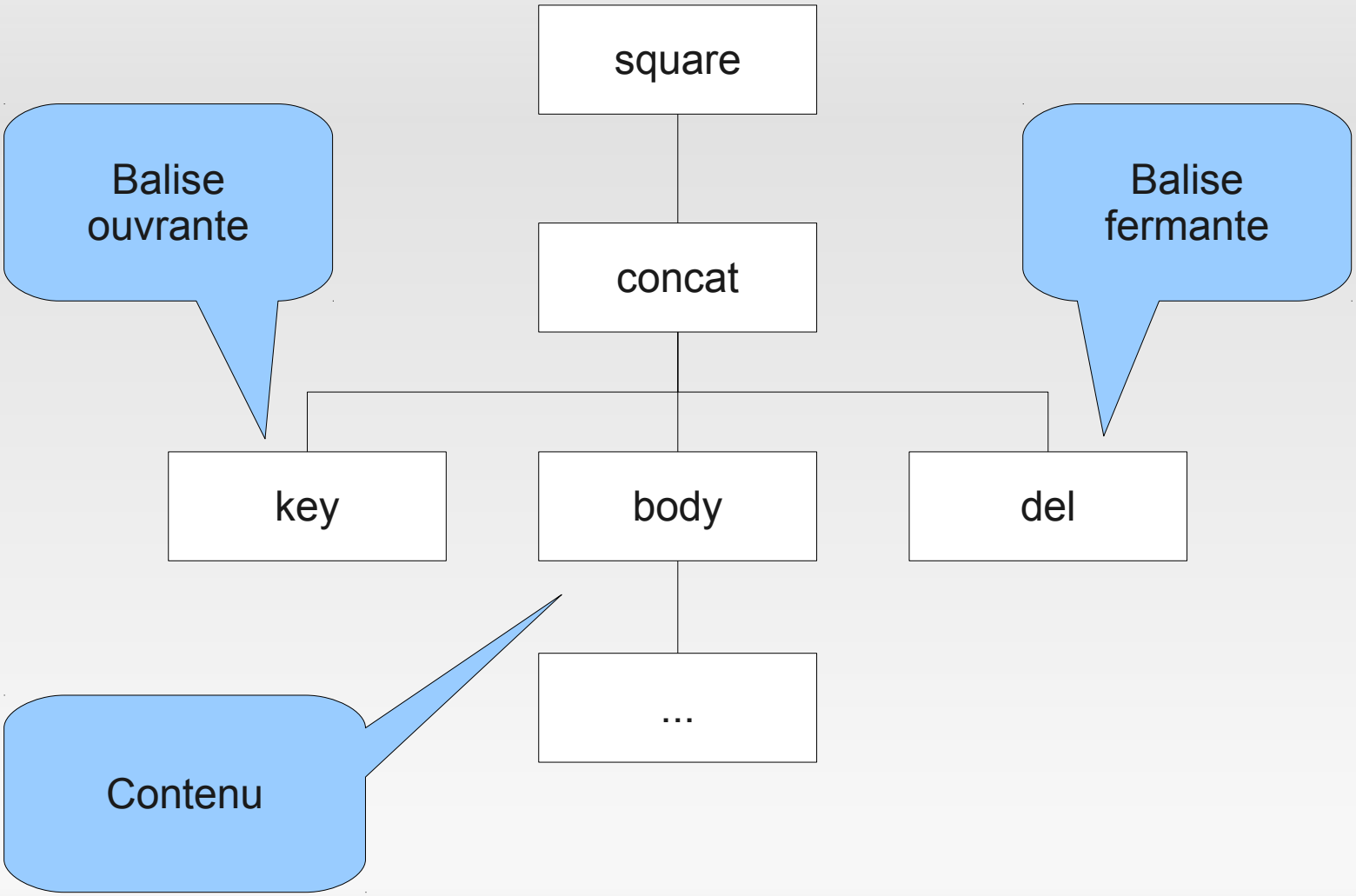
```
module Xmlfix =

let dels (s:string) = del s s
let xml2 (tag:string) = [ dels "<" . key tag . dels ">" .
                          content .
                          dels "</" . del tag tag . dels ">"]*

let a_tag = xml2 "a"

test a_tag get "<a>yyy</a>" = { "a" = "yyy" }
test a_tag get "<a>yyy</b>" = *
test a_tag put "" after set "/a" "yyy"   = "<a>yyy</a>"
```

- Toutes les balises acceptées doivent être listées... inflexible

Solution : lentille *square*



Hiérarchie de *square*



Palindromes

$$S \rightarrow aSa \mid bSb \mid a \mid b \mid \varepsilon$$

```
module Pal =

let i (s:regexp) = [ key s . value "single" ]
let p (tag:regexp) (content:lens) = [ square tag content ]

let rec s = p /[a-z]/ s | (i /[a-z]/)?

test s get "aabcbaa" =
  { "a"
    { "a"
      { "b"
        { "c" = "single" }
      }
    }
  }

test s put "aabcbaa" after clear "/a/a/b/c" = "aabccbaa"
```

Lentille XML générique

```
module Simple_xml =

let dels (s:string) = del s s
let content         = store /[a-z]*/
let body            = dels ">" . content . dels "</"
let xml             = [ dels "<" .
                        square /[a-z]+/ body .
                        dels ">" ]*

test xml get "<a>yyy</a>" = { "a" = "yyy" }
test xml get "<a>yyy</b>" = *
test xml put "" after set "/a" "yyy"   = "<a>yyy</a>"
```

- Refuse la chaine invalide
- Fonction "create" cohérente

Ambigüité *put*

```
let content = (text|element|empty)*

<x>abc</x> get→ {"x"{"#text"="abc"}}

set /a/#text[2] "def" → {"a"{"#text"="abc"}{"#text"="def"}}

{"a"{"#text"="abc"}{"#text"="def"}} put→ <x>abcdef</x>

<x>abcdef</x> get→ {"x"{"#text"="abcdef"}}

{"a"{"#text"="abc"}{"#text"="def"}} ≠ {"x"{"#text"="abcdef"}}
```

- Loi $l.get(l.put(a,c)) = a$ non respectée
- Ambigüité acceptable

Lentille Apache httpd

```
<VirtualHost *:80>
    ServerAdmin webmaster@localhost
    ....
</VirtualHost>
```

```
Début section
    Directives
    ...
Fin section
```

- `sections   atype = {/[a-zA-Z]+/}`
- `directives atype = {/[a-zA-Z]+/}`
- Ambigüité *put*!

Solution statique

- Lister toutes les sections et directives
- let sections = "VirtualHost" | ...
- let directives = "ServerAdmin" | ...
- Lentille énorme
  - ~20 sections
  - ~200 directives
- Inflexible: refuse des configurations valides
  - Apache est modulaire!

Solution hybride

- sections << directives
- let sections = "VirtualHost" | ...
- let directives = /[a-zA-Z]+/ - sections
- Problème de performance
- Noms de sections encore en dur...

Solution générique

- Utilise la lentille *square* + *c*lés synthétiques

```
{ "VirtualHost"
 { "ServerAdmin" }
}
```

```
{/[a-zA-Z]+/}
{/[a-zA-Z]+/}
Ambigüité put!
```

```
{ "#section" = "VirtualHost"
 { "#directive" = "ServerAdmin" }
}
```

```
{"#sec"=/[a-zA-Z]+/}
{"#dir"=/[a-zA-Z]+/}
disjoint
```

```
{ "VirtualHost" = "#section"
 { "ServerAdmin" = "#directive" }
}
```

```
{/[a-zA-Z]+/="#sec"}
{/[a-zA-Z]+/="#dir"}
disjoint
```

Résultats

Lentille	Temps (s.)		Mémoire (Mo)	
	avec verif.	sans vérif.	avec véfif.	sans vérif.
Exacte	5,314	0,336	1536	62
Hybride	3,257	0,077	125	6
Générique	0,090	0,048	3	1

- Générique v.s. Exacte (avec vérif.)
  - ~58 fois plus rapide
  - ~500 fois moins de mémoire

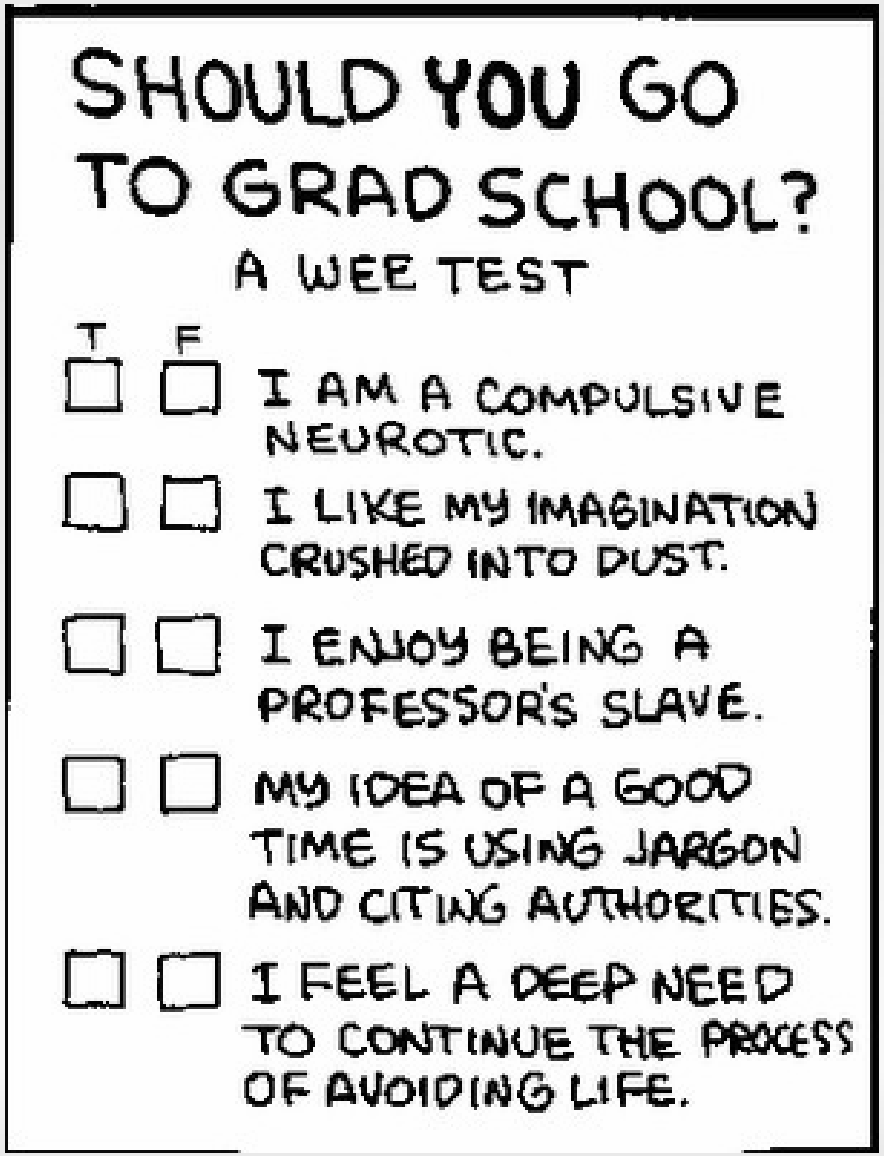
Conclusion et travaux futurs

- XSugar
  - meilleur algorithme de fusion d'arbre
- Augeas
  - Analyse statique des ambigüités get
  - Préfixe de clé pour éviter les ambigüités put
- Modifications et lentilles Augeas publiées
- Objectifs atteints

Remerciements

- Gabriel Girard, directeur, UdeS
- Richard St-Denis, codirecteur, UdeS
- Anders Moeller, Université d'Aarhus, Danemark
- David Lutterkort, RedHat, San Francisco
- Benoît des Ligneris, Révolution Linux, Sherbrooke
- Famille et amis

À propos des études supérieures...



Questions, commentaires

- Pour me joindre:
- Francis Giraldeau <francis.giraldeau@usherbrooke.ca>
- http://pages.usherbrooke.ca/fgiraldeau
- Références disponibles sur le site