

Nantes 2009 :
Rendez-vous Libres!



Augeas - a configuration API

Raphaël Pinson

Configuration Management

Sitewide configuration

Local configuration



Editing of Configuration Data

(1) Keyhole approaches

(2) Greenfield approaches

(3) Templating

Missing pieces

- Handle configuration data uniformly
- Policy/delegation
- Remotable API

Augeas lays the foundation for addressing these



Design Goals

(1) Deal with configuration data in its current place

Design Goals

(2) Expose abstract tree view of configuration data

Design Goals

(3) Preserve “unimportant” detail



Design Goals

(4) Describe new file formats easily and safely



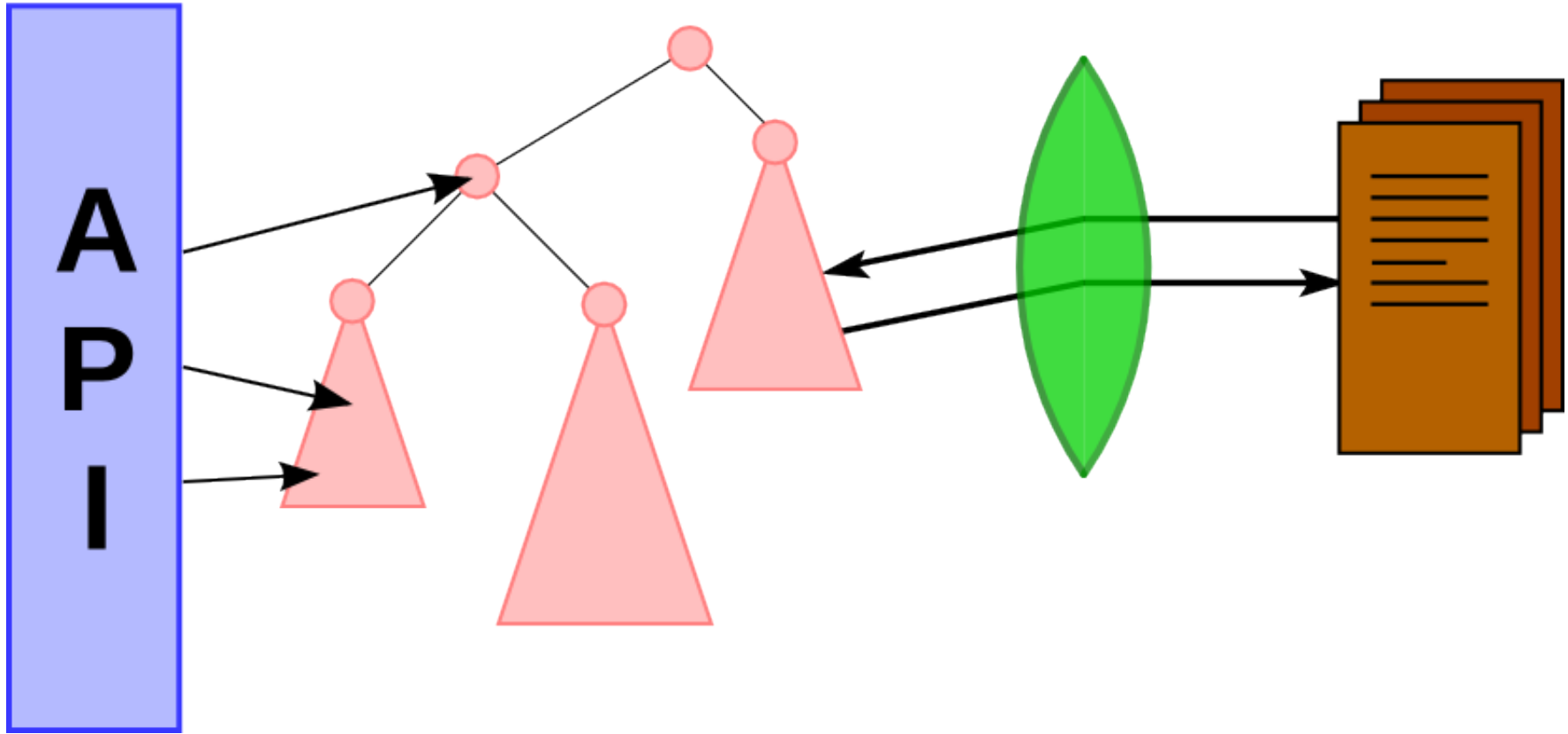
Design Goals

(5) Language neutral implementation

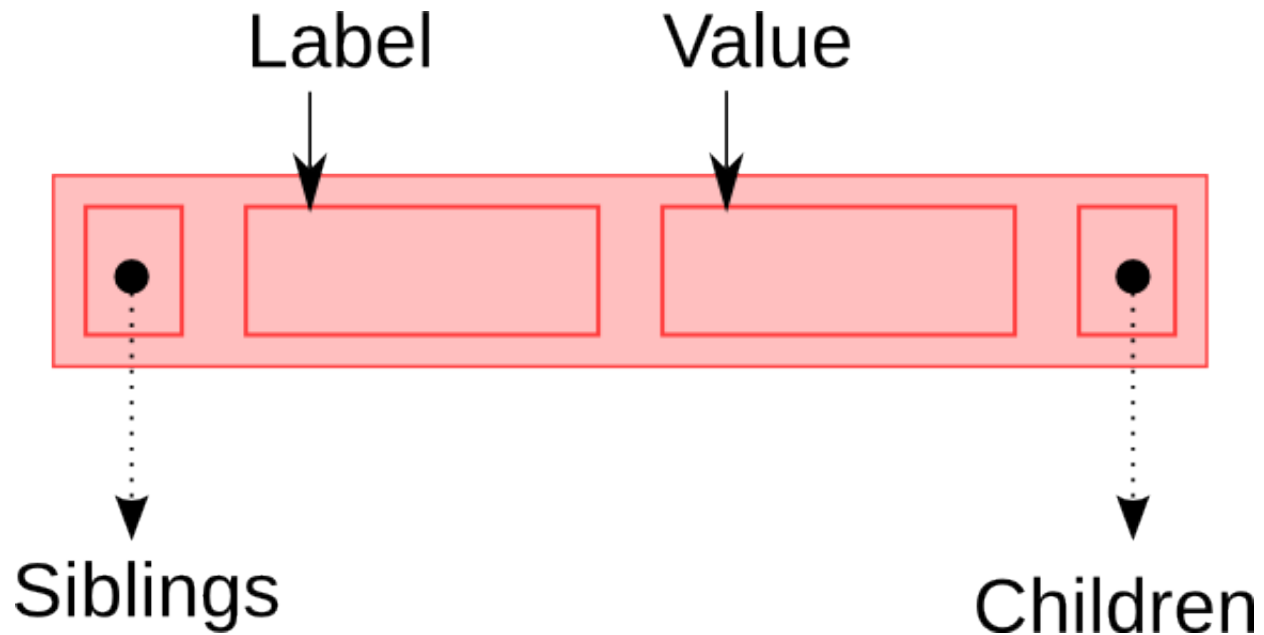
Design Goals

(6) Focus on configuration editing

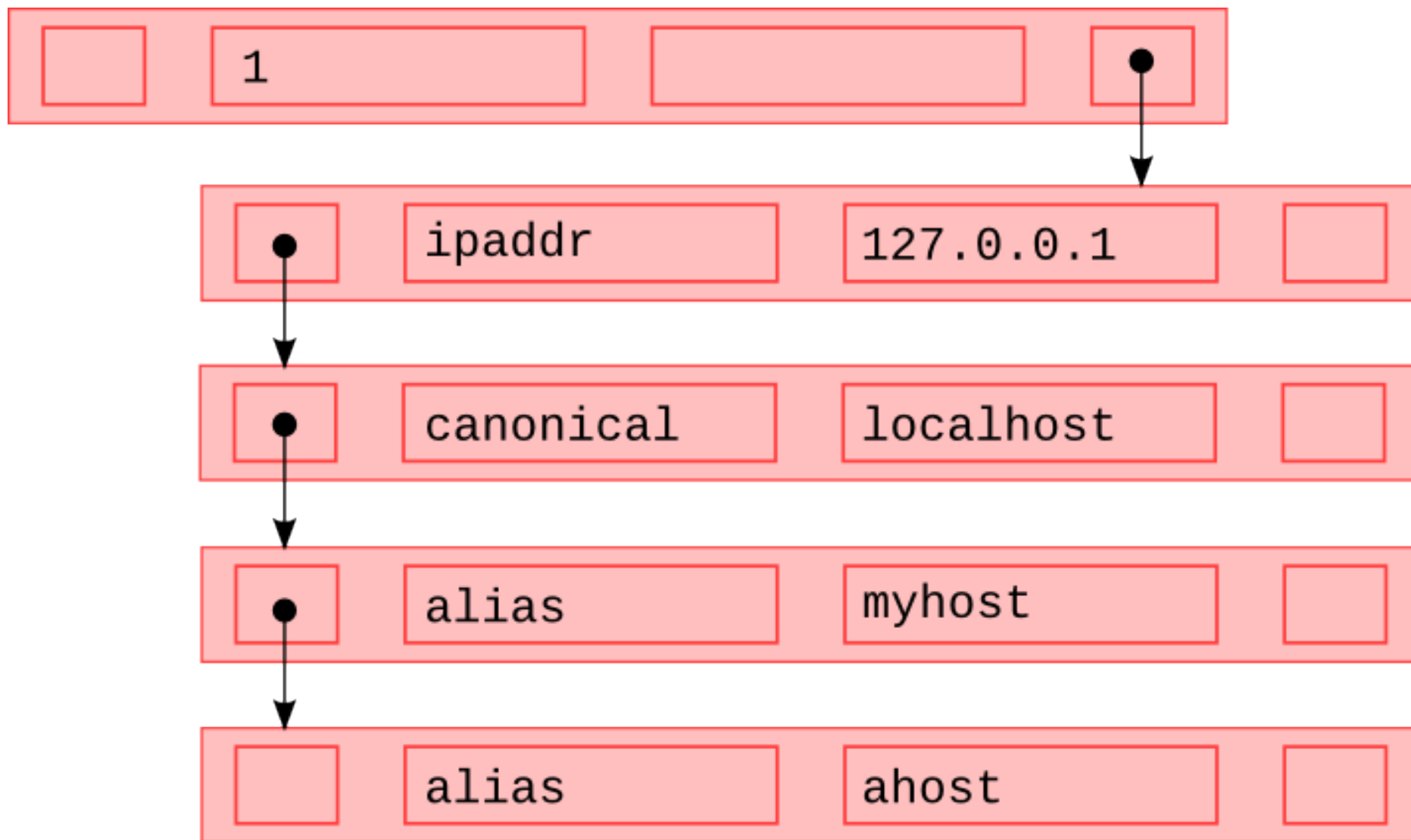
Overall architecture



The Augeas Tree



The Augeas Tree



The public Augeas API

- Small number of calls to modify tree
 - ***init/close***
 - ***get/set*** value associated with a node
 - ***match*** nodes with path expression
 - ***insert*** before/after existing node
 - ***rm*** subtree
 - ***save*** tree back to file
- Possible additions
 - copy/move subtrees
 - load specific files



The public Augeas API

C API (`libaugeas.so`)

Command line tool `augtool`

Language bindings for Python, Ruby, Ocaml, Perl, Java, ...



Example: /etc/hosts

Format:

```
# ipaddr □ canonical (□ alias)* \n
127.0.0.1 □ localhost □ localhost.localdomain □ host.domain
```

Schema:

/files/etc/hosts

1/

`ipaddr` = 127.0.0.1

`canonical` = localhost

`alias` = localhost.localdomain

`alias` = host.domain



Example: `/etc/hosts`

```
augtool> set /files/etc/hosts/1/alias[2] myhost.domain
```

Schema:

```
/files/etc/hosts
```

```
1/
```

```
ipaddr = 127.0.0.1
```

```
canonical = localhost
```

```
alias = localhost.localdomain
```

```
alias = myhost.domain
```



Example: `/etc/hosts`

```
augtool> ins alias after /files/etc/hosts/1/alias[1]
```

Schema:

```
/files/etc/hosts
```

```
1/
```

```
ipaddr = 127.0.0.1
```

```
canonical = localhost
```

```
alias = localhost.localdomain
```

```
alias
```

```
alias = myhost.domain
```



Example: `/etc/hosts`

```
augtool> set /files/etc/hosts/1/alias[2] myhost
```

Schema:

```
/files/etc/hosts
```

```
1/
```

```
ipaddr = 127.0.0.1
```

```
canonical = localhost
```

```
alias = localhost.localdomain
```

```
alias = myhost
```

```
alias = myhost.domain
```



Example: /etc/hosts

```
augtool> save
```

New /etc/hosts:

```
# ipaddr □ canonical (□ alias)* \n
127.0.0.1 □ localhost □ localhost.localdomain □ myhost □
    myhost.domain
```



Example: logrotate configuration

Trees under

```
/files/etc/logrotate.conf
```

```
/files/etc/logrotate.d/a_rule
```

Schema

```
/rule/key = value
```

Add a file to a logrotate rule :

```
R=/files/etc/logrotate.d/base-config/rule[1]
```

```
augtool> ins file after $R/file
```

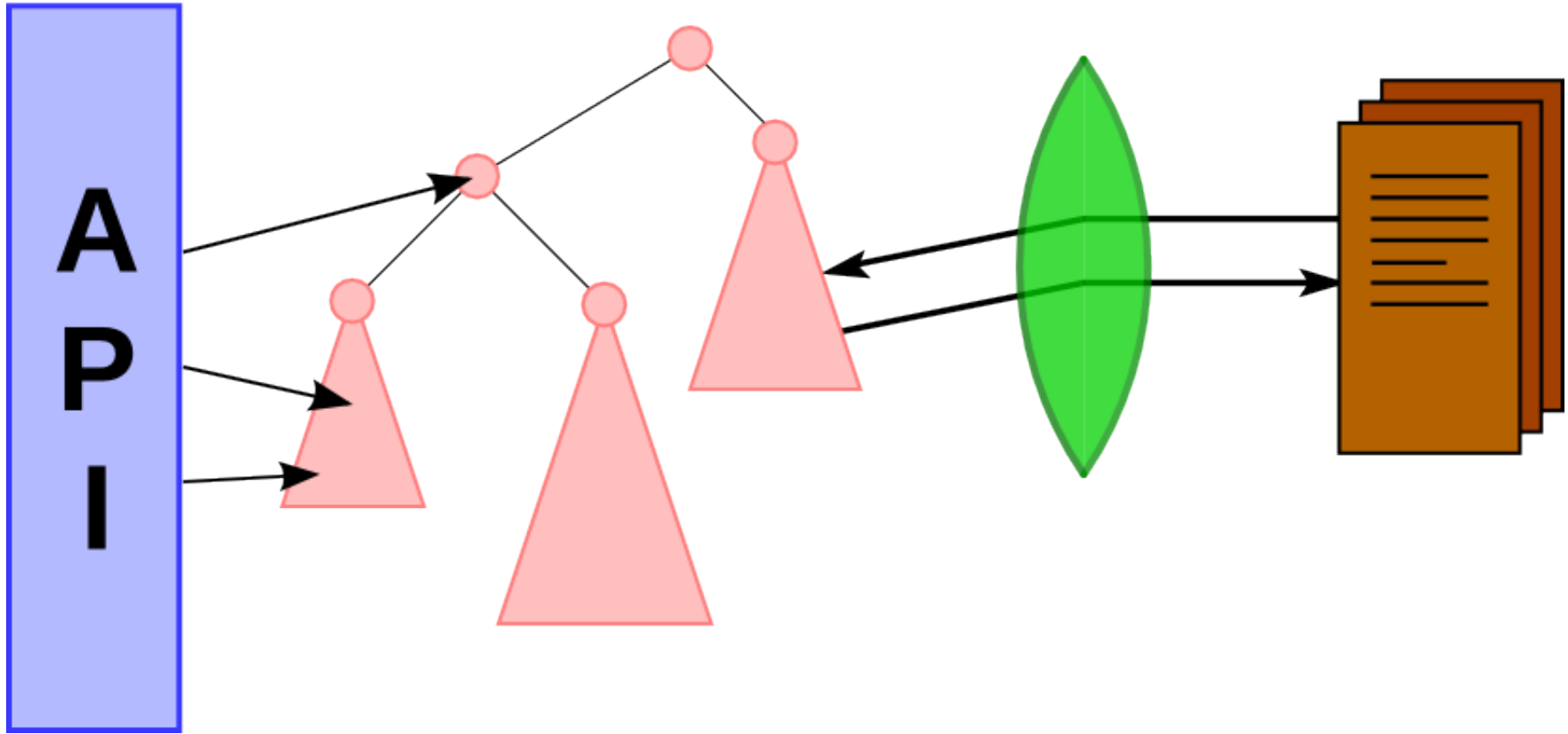
```
augtool> set $R/file[2] "/var/log/test.log"
```



Example: configuring an Acer Aspire One

- Grub
- Fstab

Overall architecture



Example: hosts.aug

```
module Hosts =
  autoload xfm

  let ws = del /[ \t]+/ " "
  let eol = del "\n" "\n"

  let comment = [ del /(#.*|[ \t]*)\n/ "\n" ]

  let word = /^[^# \n\t]+/

  let record =
    [ seq "host" . [ label "ipaddr" . store word ] .
      ws . [ label "canonical" . store word ] .
      [ label "alias" . ws . store word ]* .
    eol ]
```



Schema description

```
module Yum =  
  autoload xfm  
  
  let lns = ...  
  
  let filter = (incl "/etc/yum.conf")  
    . (incl "/etc/yum.repos.d/*")  
    . Util.stdexcl  
  
  let xfm = transform lns filter
```



Schema description

```
module Yum =  
  autoload xfm  
  
  let lns = ...  
  
  let filter = (incl "/etc/yum.conf")  
               . (incl "/etc/yum.repos.d/*")  
               . Util.stdexcl  
  
  let xfm = transform lns filter
```



Schema description

```
module Yum =  
  autoload xfm  
  
  let lns = ...  
  
  let filter = (incl "/etc/yum.conf")  
    . (incl "/etc/yum.repos.d/*")  
    . Util.stdexcl  
  
  let xfm = transform lns filter
```

Schema description

```
module Yum =  
  autoload xfm  
  
  let lns = ...  
  
  let filter = (incl "/etc/yum.conf")  
               . (incl "/etc/yum.repos.d/*")  
               . Util.stdexcl  
  
  let xfm = transform lns filter
```



Schema description

```
module Yum =
```

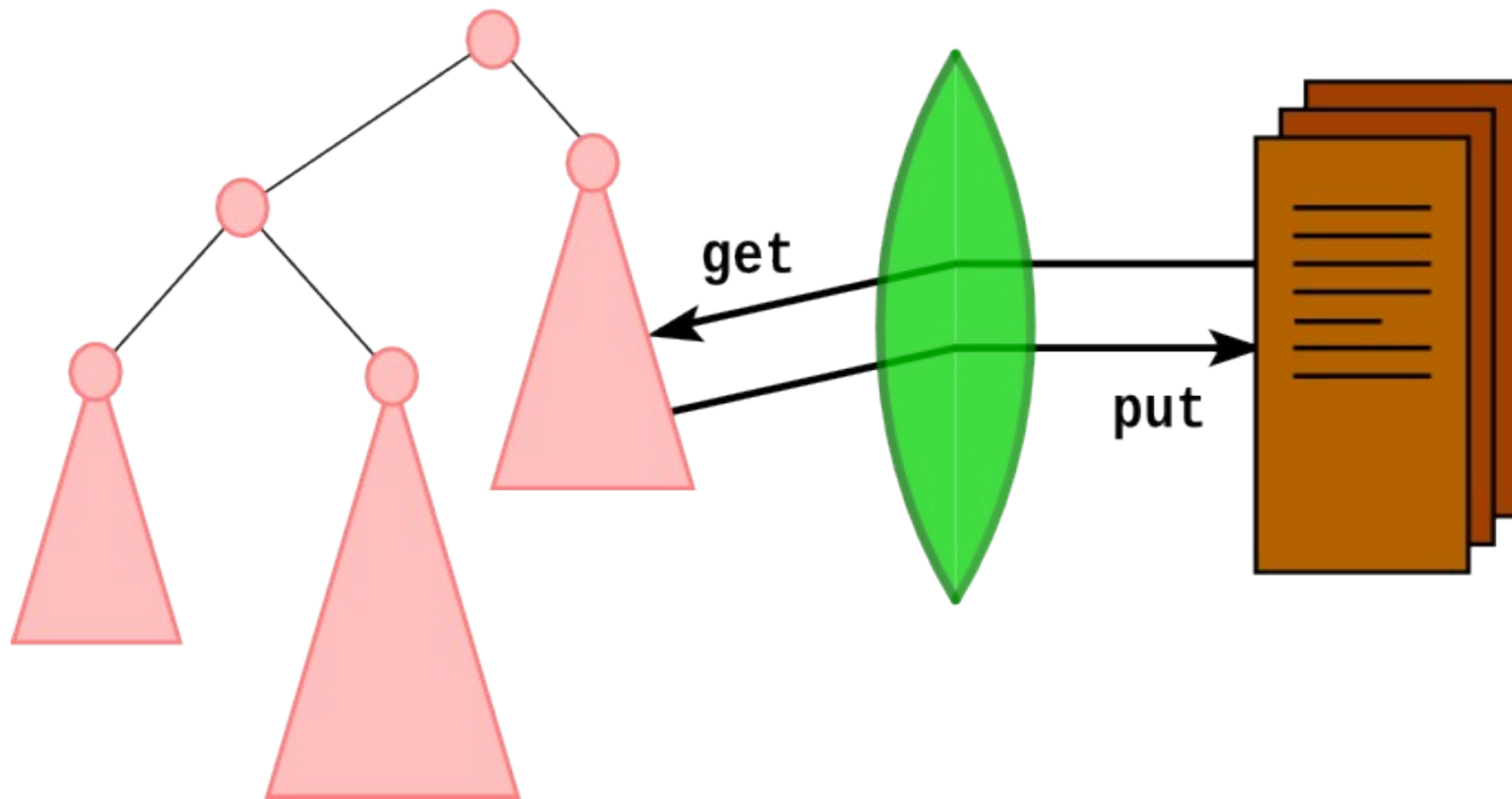
```
  autoload xfm
```

```
  let lns = ...
```

```
  let filter = (incl "/etc/yum.conf")  
               . (incl "/etc/yum.repos.d/*")  
               . Util.stdexcl
```

```
  let xfm = transform lns filter
```

Lenses



Lenses

Concrete View \leftrightarrow Abstract View

Bidirectional programming

Concrete \rightarrow Abstract $+$ Abstract \rightarrow Concrete

- Harmony (U Penn) does it for trees
- Boomerang (U Penn) does it for strings
- Theoretical groundwork by B. Pierce, N. Foster et.al.



Lenses for Augeas

String \leftrightarrow Tree

get: String \rightarrow Tree

put: Tree x String \rightarrow String

Lens Laws

The get and put of every lens must fulfill:

$$\text{put (get c) c} = \text{c}$$

$$\text{get (put a c)} = \text{a}$$

- Capture intuitive notions of “minimal” edits
- Constraints enforced by typechecker



Lens primitives

- Tree labels
 - key re
 - label str
 - seq str
- Tree values
 - store re
- Omit from tree
 - del re str



Lens combinators

- $l_1 \cdot l_2$: Lens concatenation
- $l_1 | l_2$: Lens union
- l^* , l^+ : Lens iteration
- $[l]$: Subtree combinator

Lens development

- Build up lenses from small parts
- Reuse common constructs
 - Comment goes from # to end of line
- Unit test facility in Augeas language
 - Run get direction
 - Run get direction, modify tree, run put direction
 - Compare to fixed value
 - Assert exception
 - Print result



Lens development

Process “key=value”



Lens development

Process "key=value"

```
let eq = del "=" "="
```

Lens development

Process "key=value"

```
let eq = del "=" "="
```

```
let lns = [ key /[a-z]+/ . eq . store /.+/ ]
```

Lens development

Process "key=value"

```
let eq = del "=" "="
```

```
let lns = [ key /[a-z]+/ . eq . store /.+/ ]
```



Lens development

Process "key=value"

```
let eq = del "=" "="
```

```
let lns = [ key /[a-z]+/ . eq . store /.+/ ]
```

Lens development

Process "key=value"

```
let eq = del "=" "="
```

```
let lns = [ key /[a-z]+/ . eq . store /.+/ ]
```

```
test lns get "foo=bar" = ?
```

Lens development

Process "key=value"

```
let eq = del "=" "="
```

```
let lns = [ key /[a-z]+/ . eq . store /.+/ ]
```

```
test lns get "foo=bar" = { "foo" = "bar" }
```



Lens development

Process "key=value"

```
let eq = del "=" "="
```

```
let lns = [ key /[a-z]+/ . eq . store /.+/ ]
```

```
test lns get "foo2=bar1" = *
```



Lens development

Process "key=value"

```
let eq = del "=" "="
```

```
let lns = [key /[a-z]+/ . eq . store /.+/ ]
```

```
test lns put "foo=bar"  
      after set "foo" "baz"  
      = ?
```



Lens development

Process "key=value"

```
let eq = del "=" "="
```

```
let lns = [key /[a-z]+/ . eq . store /.+/ ]
```

```
test lns put "foo=bar"
```

```
    after set "foo" "baz"
```

```
    = ?
```

Lens development

Process "key=value"

```
let eq = del "=" "="
```

```
let lns = [key /[a-z]+/ . eq . store /.+/ ]
```

```
test lns put "foo=bar"
```

```
    after set "foo" "baz"
```

```
    = "foo=baz"
```

Lens development

Process "key=value"

```
let eq = del /[ \t]+=[ \t]+/ "="
```

```
let lns = [ key /[a-z]+/ . eq . store /.+/ ]
```



Lens development

Process "key=value"

```
let eq = del /[ \t]+=[ \t]+/ "="
```

```
let ins = [ key /[a-z]+/ . eq . store /.+/ ]
```

Lens development

Process "key=value"

```
let eq = del /[ \t]+=[ \t]+/ "="
```

```
let lns = [ key /[a-z]+/ . eq . store /[a-z]+/ ]
```

```
test lns put "foo \t= bar"
```

```
    after set "foo" "baz"
```

```
    = "foo \t= baz"
```

Arrays – using *seq*

hosts/

1/

ipaddr

canonical

alias

alias

2/

ipaddr

canonical

alias



Arrays – using identical labels

hosts/

1/

ipaddr

canonical

alias

alias

2/

ipaddr

canonical

alias



Handling comments

```
let comment = del /#.*\n/ "#\n"
```

```
let lns = (record|comment)*
```



Handling comments

```
let comment = [ del /#.*\n/ "#\n" ]
```

```
let ins = (record|comment)*
```

Other possibilities :

- Managing comments as fields
- Managing commented values (parsable)

Cf http://augeas.net/page/Dealing_with_comments



The lens typechecker

- Each lens has associated *ctype* and *atype*
 - Regular languages
- Checks during lens construction
 - *del re str : str* must match *re*
 - *l1 . l2* : unambiguously splittable
 - *l1 | l2* : disjoint regular languages
- `libfa` for finite automata computations
- Restricts Augeas to regular file formats

Supported file formats

/etc/hosts /etc/inittab yum config /etc/fstab /etc/exports
/etc/security/limits.conf monit openvpn puppet.conf
/etc/aliases /etc/darkice.cfg /etc/ssh/sshd_config ntp
/etc/bb-hosts dhclient.conf dnsmasq.conf dpkg.cfg
gdm.conf /etc/group /etc/network/interfaces
shell vars in /etc/sysconfig/ squid.conf logrotate
rsyncd.conf ifcfg-* samba sysconfig
apt preferences/sources dput sudoers pam.d
slapd.conf soma ldap.conf havp.conf
grub.conf webmin xinetd.d vsftpd.conf

[your contribution here](#)



What about `httpd.conf` ?

- Mostly tedious boilerplate
- Except:

```
...  
    <IfModule mod_proxy.c>  
        ...  
    </IfModule>  
...
```

- Arbitrary nesting, not regular
 - Need recursion + regular approximation

A higher level service

DBus service backed by Augeas

+

PolicyKit mechanism for authentication

=

Local configuration service

UI independent

File format independent

Fine grained permissioning

Harald Hoyer has prototype for `system-config-boot`



Projects using Augeas

- Puppet : configuration deployment tool
 - <http://puppet.reductivelabs.com>
- Config::Augeas & Config::Model
 - Perl modules by Dominique Dumont for configuration editing
 - Can use Augeas as a backend
 - <http://search.cpan.org/dist/Config-Augeas>
- Ebox : site administration
 - <http://ebox-platform.com>



Supported platforms

- Red Hat Linux flavors
 - Fedora, RHEL, CentOS, ...
- Other Linux flavors
 - Debian, Ubuntu, ...
- FreeBSD
- OS/X port on the way

Minimal dependencies

- Anything with a GNU libc (or equivalent gnulib support)

More information

- Project website <http://augeas.net/>
 - Read the “Quick Tour” first
- Mailing list augeas-devel@redhat.com
- IRC #augeas on freenode

