

# Rehearsal

## A Configuration Verification Tool for Puppet

Rian Shambaugh, Aaron Weiss, and Arjun Guha







```
apt-get install apache2
```



```
apt-get install apache2  
vim /etc/apache2/sites-enabled/default
```



```
apt-get install apache2  
vim /etc/apache2/sites-enabled/default  
service apache2 restart
```



# 502 Bad Gateway

The server returned an invalid or incomplete response.

Display a menu



```
apt-get install apache2  
vim /etc/apache2/sites-enabled/default  
service apache2 restart
```



```
apt-get install apache2  
vim /etc/apache2/sites-enabled/default  
service apache2 restart
```

```
vim /etc/apache2/sites-enabled/default  
service apache2 restart
```





```
apt-get install apache2  
vim /etc/apache2/sites-enabled/default  
service apache2 restart
```

```
vim /etc/apache2/sites-enabled/default  
service apache2 restart
```

```
iptables -dport ssh -j DROP
```

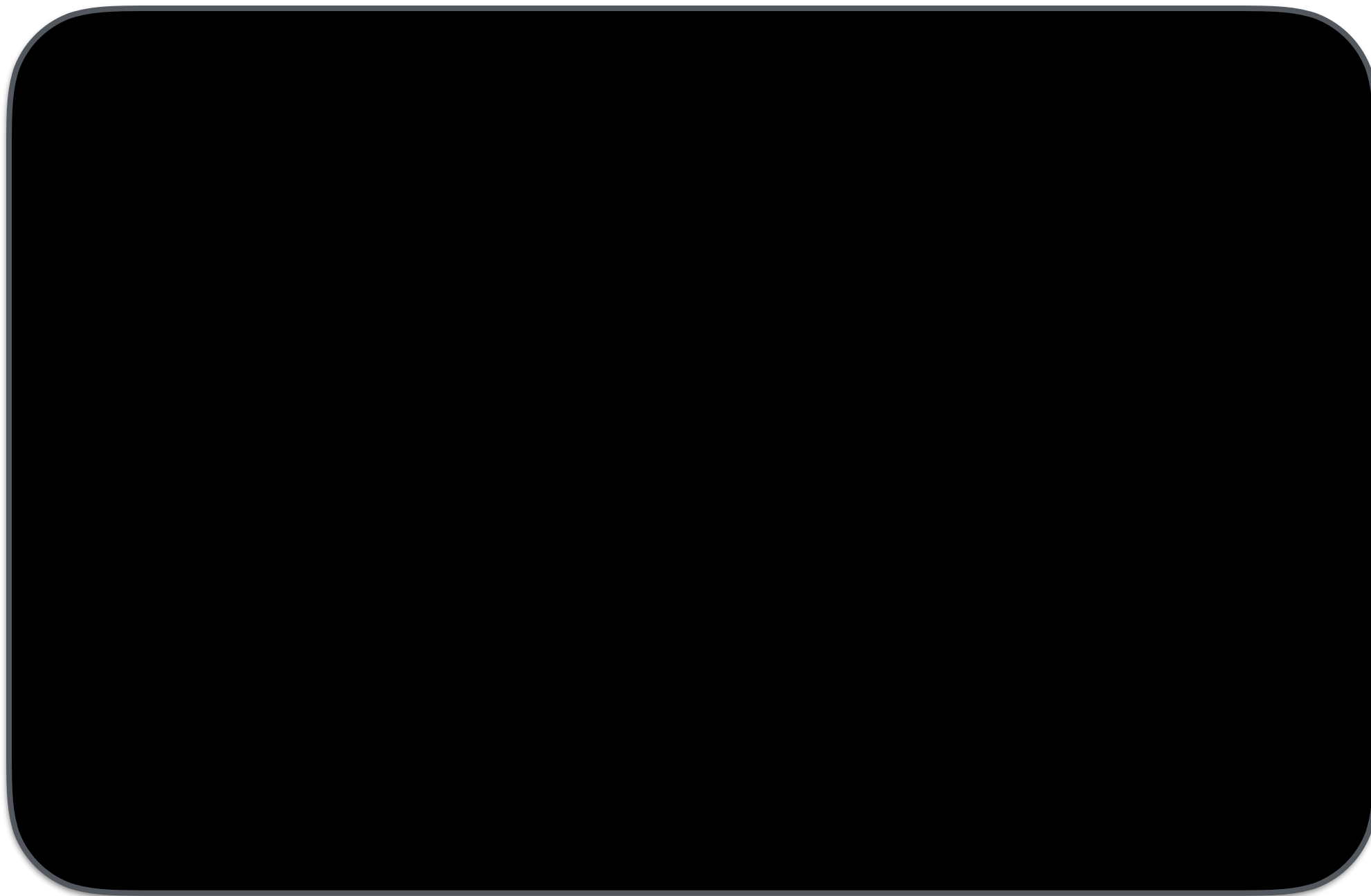
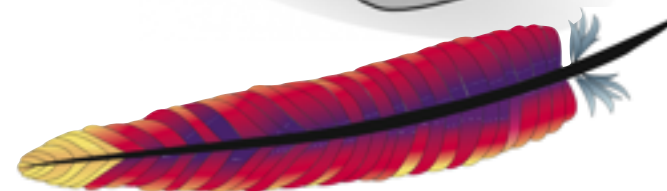
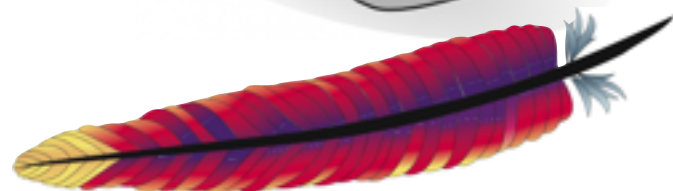
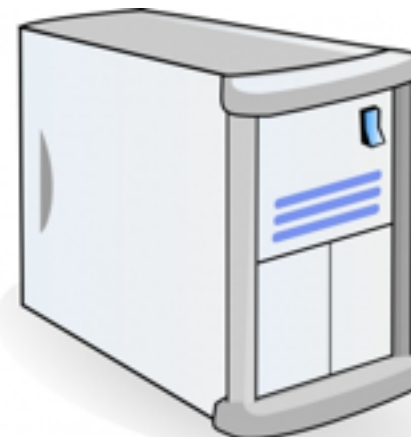
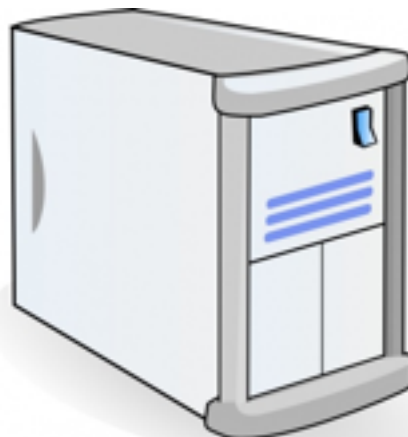


```
apt-get install apache2  
vim /etc/apache2/sites-enabled/default  
service apache2 restart
```

```
vim /etc/apache2/sites-enabled/default  
service apache2 restart
```

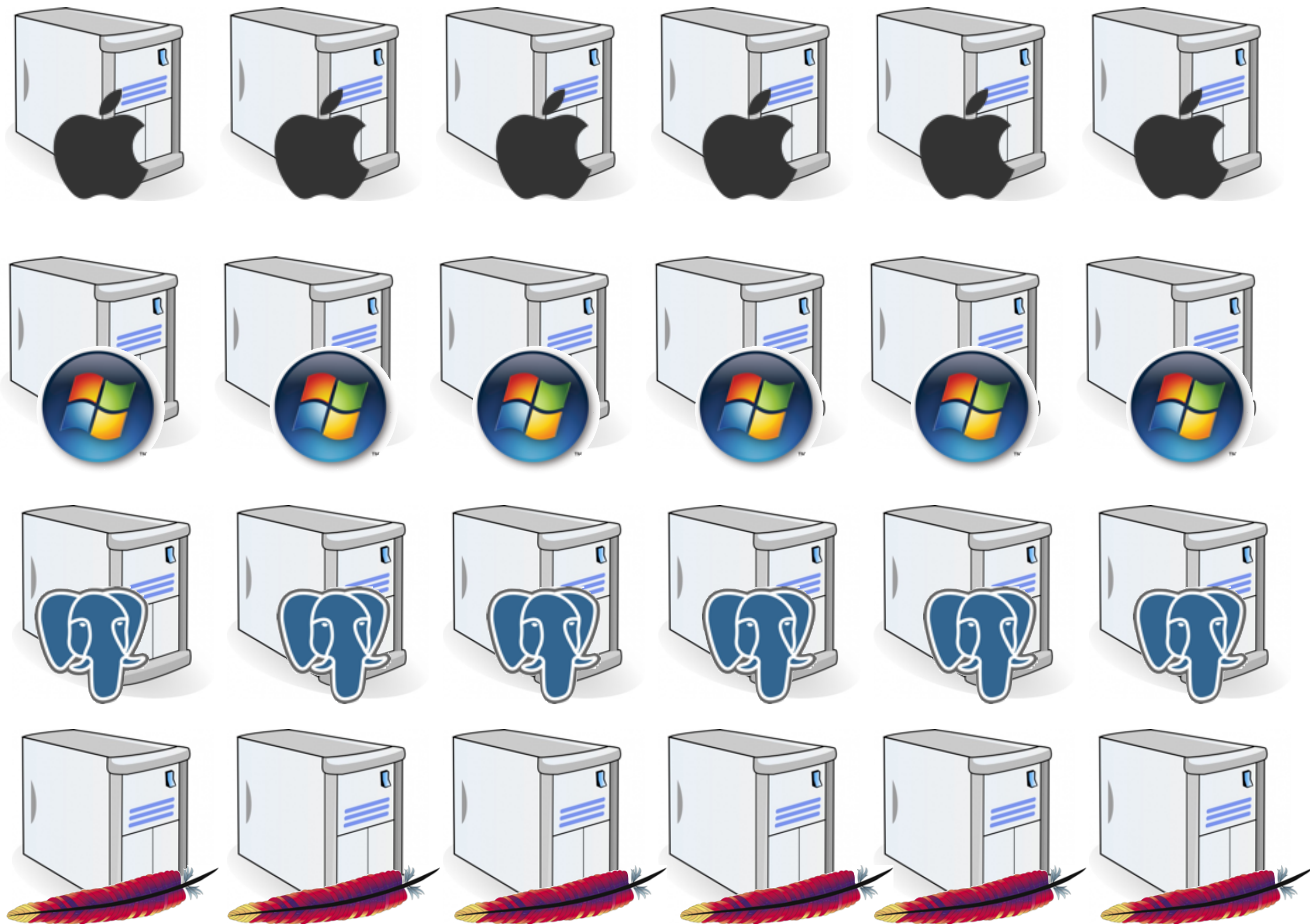
```
iptables -dport ssh -j DROP
```

```
mount backup.local:/backup /mnt/backup  
crontab -e
```





*What were  
those  
commands?*





## New York Stock Exchange:

"a software update went out [...] it returned an error. [...] There was clearly a difference in the configuration going into production [from the test environment]"

Airbus military plane crash: "The error was in configuration settings programmed into the electronic control unit (ECU) of the engines and not in the code itself."



## Facebook:

"Facebook was down or unreachable for many of you for approximately 2.5 hours. [...] An automated system for verifying configuration values ended up causing much more damage than it fixed."







## New York Stock Exchange:

"a software update went out [...] it returned an error. [...] There was clearly a difference in the configuration going into production [from the test environment]"

Airbus military  
configuration  
electronic con  
not in the cod



**Kingston Police**  
@MPSKingston



 **Follow**

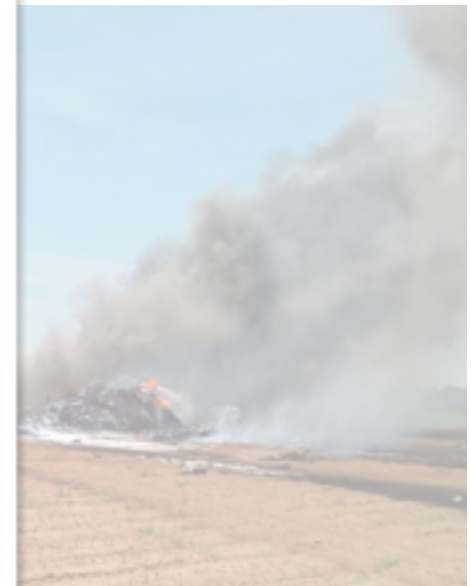
Yes we can confirm Facebook is down,  
please don't call us! What a great opportunity  
to spend some time with your family...  
[#FacebookDown](#)

RETWEETS  
**1,475**

FAVORITES  
**1,203**



4:27 PM - 28 Sep 2015



"Facebook was down or unreachable for many of you for approximately 2.5 hours. [...] An automated system for verifying configuration values ended up causing much more damage than it fixed."

# Configuration Management Tools





# 30,000+ organizations use Puppet



**Nestlé**



**NYSE**

**OfficeMax**



**Raytheon**

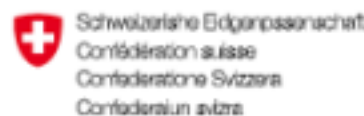


salesforce



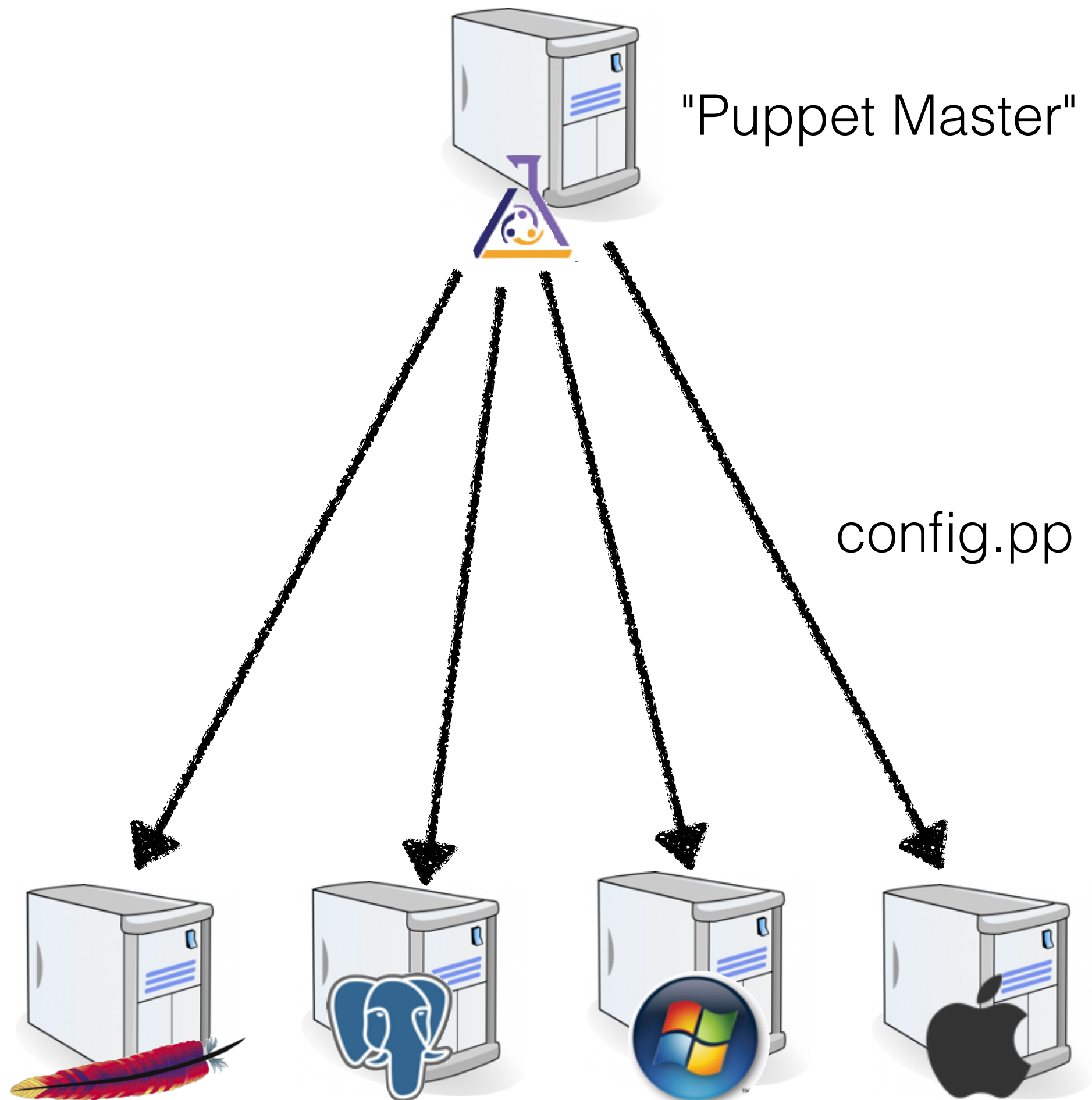
**SONY**

**SPICEWORKS**  
IT'S EVERYTHING IT



Trans**Union**





```
file{"/home/arjun/.ssh/authorized_keys":  
  content => "ssh-rsa AAAAB3NzaC1yc2EAAAADA ..."  
}
```

```
file{"/home/arjun/.ssh/authorized_keys":  
    content => "ssh-rsa AAAAB3NzaC1yc2EAAAADA ..."  
    source  => "puppet://default_keys"  
}
```

```
file{"/home/arjun/.ssh/authorized_keys":  
  content => "ssh-rsa AAAAB3NzaC1yc2EAAAADA ..."  
  source  => "puppet:///default_keys",  
  owner   => arjun,  
  group   => arjun,  
  mode    => 0600  
}
```

```
file{"/home/arjun/.ssh/authorized_keys":  
  content => "ssh-rsa AAAAB3NzaC1yc2EAAAADA ..."  
  source  => "puppet:///default_keys",  
  owner   => arjun,  
  group   => arjun,  
  mode    => 0600,  
  force   => true  
}
```

```
file{"/home/arjun/.ssh/authorized_keys":  
    content => "ssh-rsa AAAAB3NzaC1yc2EAAAADA ..."  
    source  => "puppet:///default_keys",  
    owner   => arjun,  
    group   => arjun,  
    mode    => 0600,  
    force   => true,  
    backup  => ".old"  
}
```

```
file{"/home/arjun/.ssh/authorized_keys":  
  content => "ssh-rsa AAAAB3NzaC1yc2EAAAADA ..."  
  source  => "puppet://default_keys",  
  owner   => arjun,  
  group   => arjun,  
  mode    => 0600,  
  force   => true,  
  backup  => ".old",  
  replace => false  
}
```



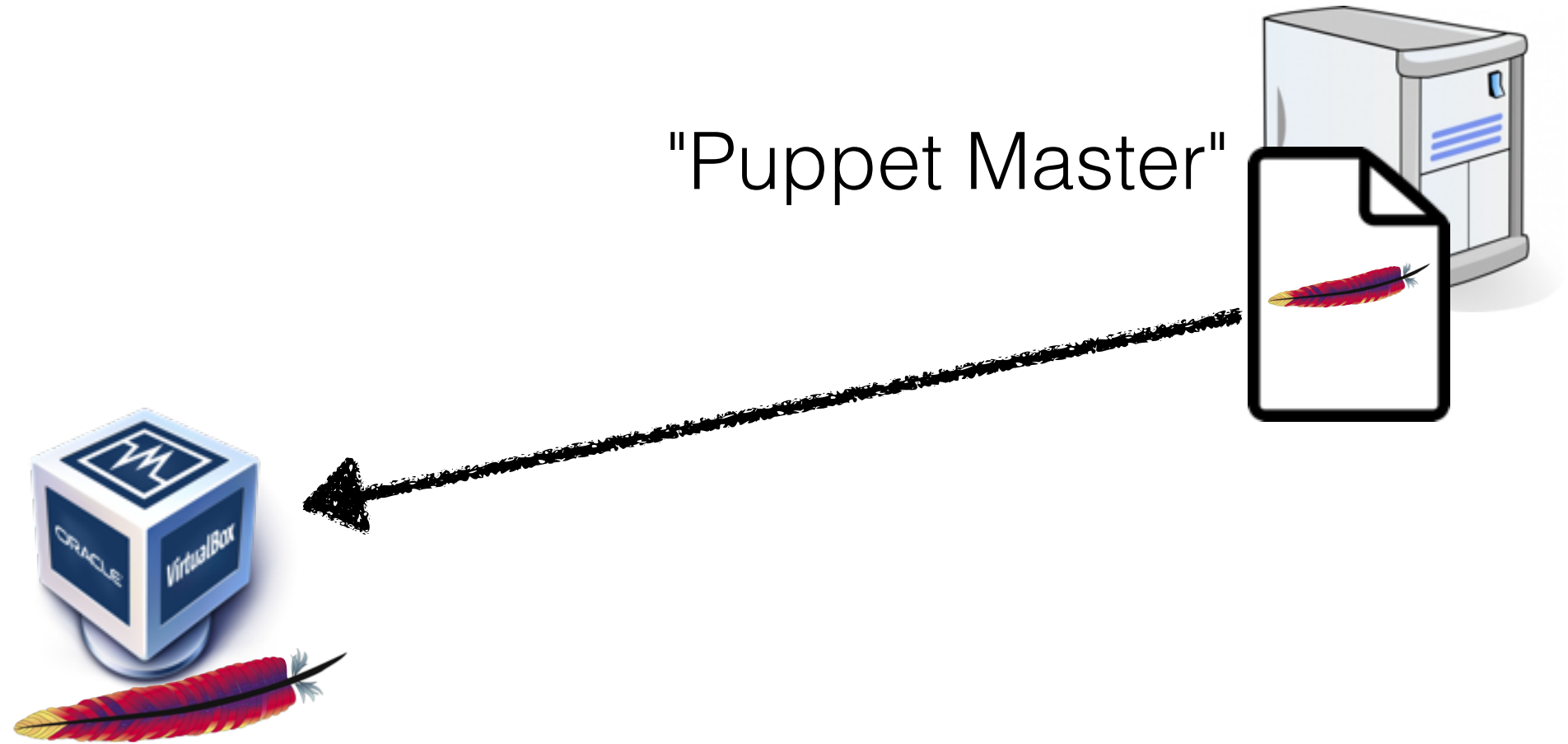
```
file{"/home/arjun/.ssh/authorized_keys":
  content => "ssh-rsa AAAAB3NzaC1yc2EAAAADA ... "
  source  => "puppet://default_keys",
  owner   => arjun,
  group   => arjun,
  mode    => 0600,
  force   => true,
  backup  => ".old",
  replace => false
}

ssh_authorized_key{"mykey":
  key => AAAAB3NzaC1yc2EAAAADA,
  user => arjun,
  type => rsa
}
```

```
file{"/home/arjun/.ssh/authorized_keys":  
  content => "ssh-rsa AAAAB3NzaC1yc2EAAAADA ..."  
  source  => "puppet://default_keys",  
  owner   => arjun,  
  group   => arjun,  
  mode    => 0600,  
  force   => true,  
  backup  => ".old",  
  replace => false  
}  
  
ssh_authorized_key{"mykey":  
  key => AAAAB3NzaC1yc2EAAAADA,  
  user => arjun,  
  type => rsa  
}
```

*Declarative specification of machine state.*

# 1. Testing



1. Testing



$\equiv$

2. Production



$\equiv$



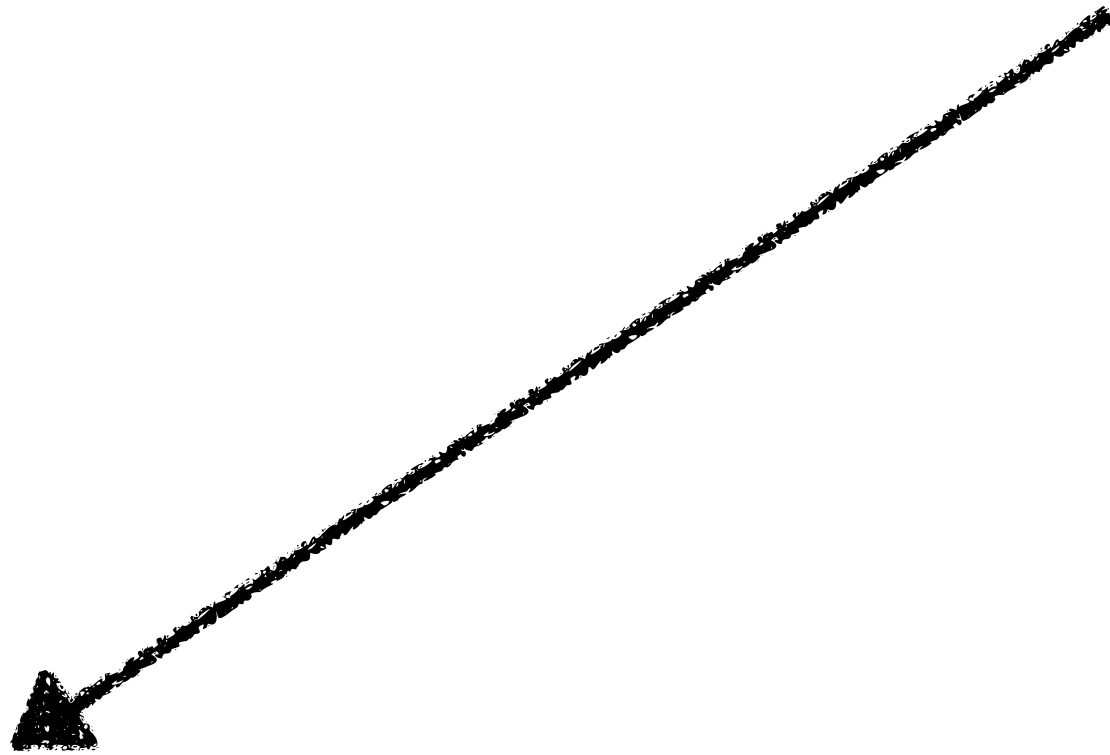
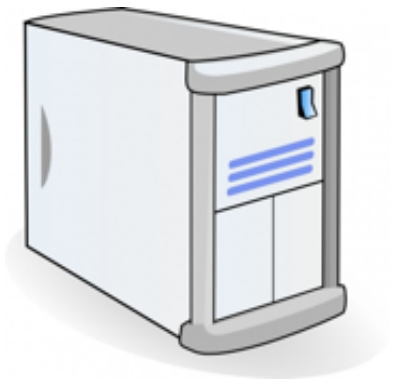
$\equiv$



"Puppet Master"

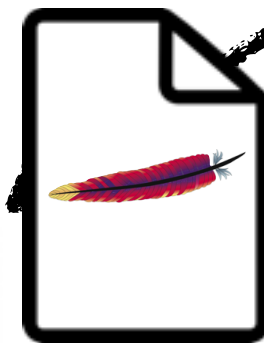
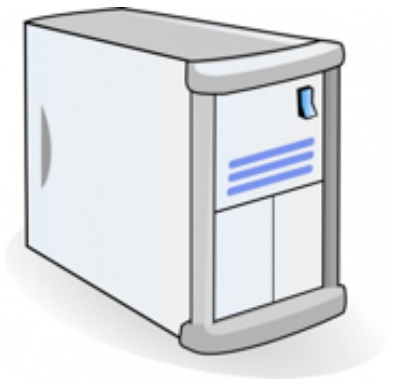


"Puppet Master"



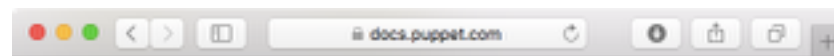
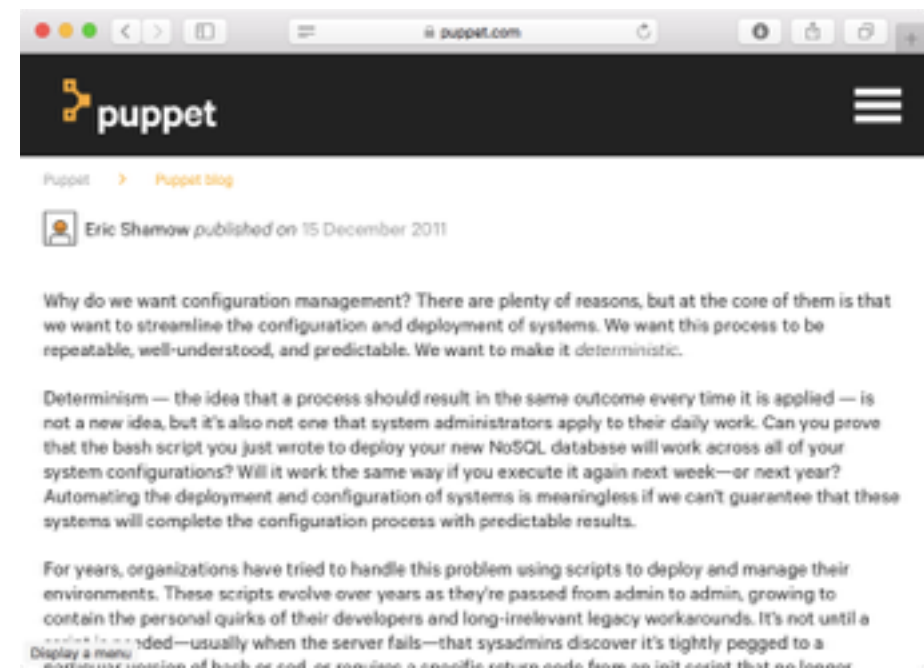
Key Property: manifest application should be *idempotent*

"Puppet Master"



Key Property: manifest application should be *idempotent*

*"Why do we want configuration management? [...] We want this process to be repeatable, well-understood, and predictable. We want to make it deterministic."*



*"One big difference between Puppet and most other tools is that Puppet configurations are idempotent, meaning they can safely be run multiple times."*



# Twisted Semantics of Puppet



```
user{"carol":  
  ensure => present  
}
```

```
file{"/home/carol/.vimrc":  
  contents => "syntax on"  
}
```

```
package{"vim":  
  ensure => present  
}
```

# Resources can be installed in any valid order

```
user{"carol":  
  ensure => present  
}
```

```
file{"/home/carol/.vimrc":  
  contents => "syntax on"  
}
```

```
package{"vim":  
  ensure => present  
}
```

```
User["carol"] -> File["/home/carol/.vimrc"]
```

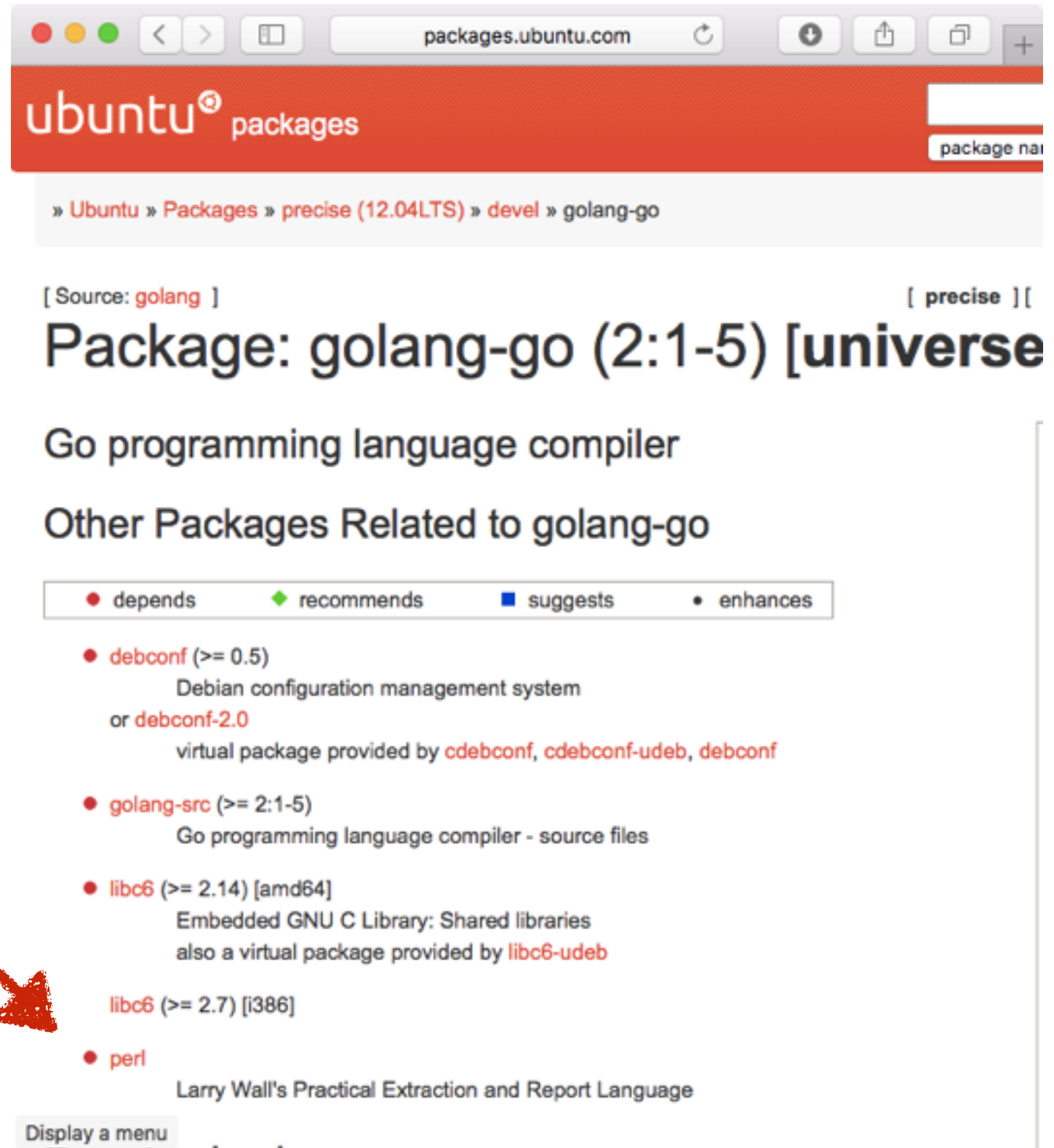
```
package{ 'golang-go':  
    ensure => present  
}
```

```
package{ 'perl':  
    ensure => absent  
}
```

# Silent non-determinism

```
package{'golang-go':  
  ensure => present  
}
```

```
package{'perl':  
  ensure => absent  
}
```



The screenshot shows the Ubuntu Packages website for the `golang-go` package. The browser address bar shows `packages.ubuntu.com`. The page title is "Package: golang-go (2:1-5) [universe]". Below the title, it says "Go programming language compiler". There is a section "Other Packages Related to golang-go" with a legend for dependencies: red circle for "depends", green diamond for "recommends", blue square for "suggests", and black dot for "enhances". The list of related packages includes:

- `debconf` (`>= 0.5`): Debian configuration management system or `debconf-2.0` (virtual package provided by `cdebconf`, `cdebconf-udeb`, `debconf`)
- `golang-src` (`>= 2:1-5`): Go programming language compiler - source files
- `libc6` (`>= 2.14`) [amd64]: Embedded GNU C Library: Shared libraries also a virtual package provided by `libc6-udeb`
- `libc6` (`>= 2.7`) [i386]
- `perl`: Larry Wall's Practical Extraction and Report Language

A red arrow points from the `perl` package in the list to the `perl` package in the Puppet code block on the left.

# Simple fix: pick an installation order?

```
file{"/home/carol/.vimrc":  
  contents => "syntax on"  
}
```

```
package{"vim":  
  ensure => present  
}
```

```
user{"carol":  
  ensure => present  
}
```

```
User["carol"] ->  
Package["vim"] ->  
File["/home/carol/.vimrc"]
```

```
package{"golang-go":  
  ensure => present  
}
```

```
package{"perl":  
  ensure => absent  
}
```

```
Package["perl"] ->  
Package["golang-go"]
```

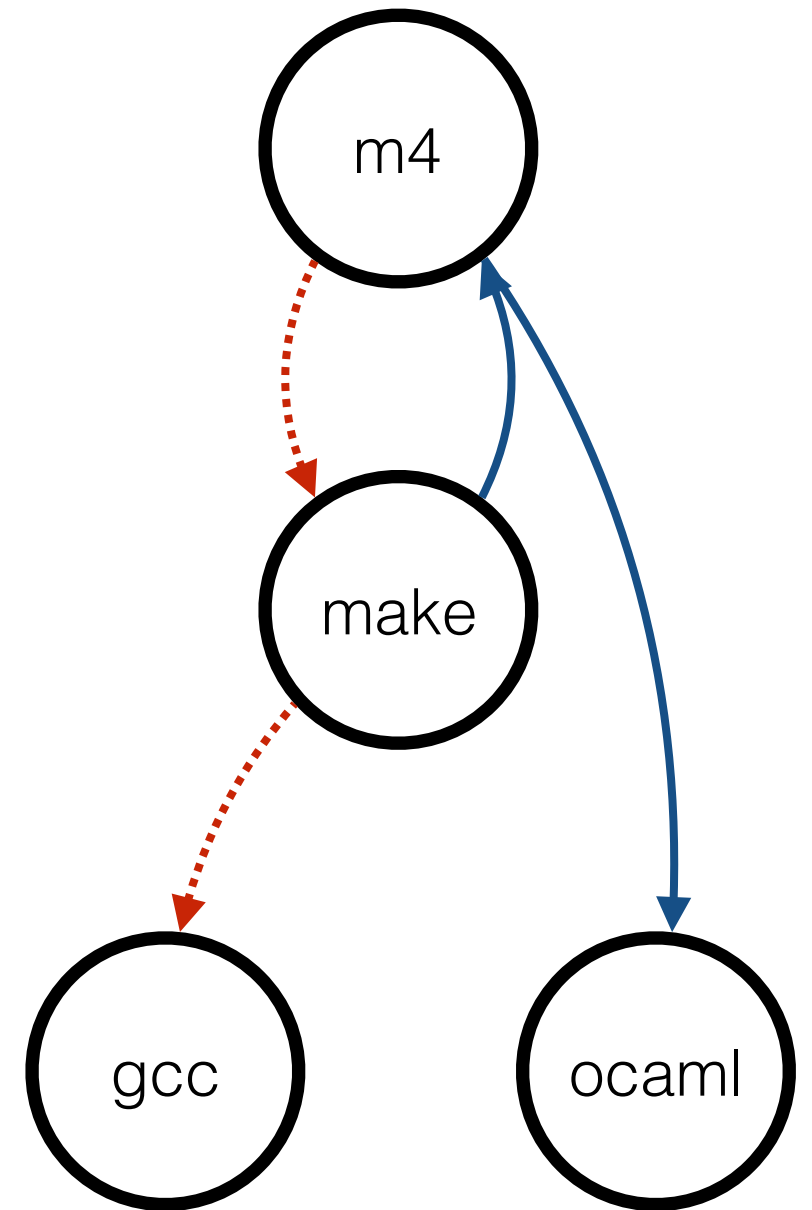
```
define cpp() {  
  
    package{'m4':ensure => present }  
    package{'make': ensure => present }  
    package{'gcc': ensure => present }  
}
```

```
define ocaml() {  
  
    package{'m4':ensure => present }  
    package{'make': ensure => present }  
    package{'ocaml': ensure => present }  
}
```

```
cpp{ }  
ocaml{ }
```

# Cyclic dependencies

```
define cpp() {  
    package{'m4':ensure => present }  
    package{'make': ensure => present }  
    package{'gcc': ensure => present }  
}  
  
define ocaml() {  
    package{'m4':ensure => present }  
    package{'make': ensure => present }  
    package{'ocaml': ensure => present }  
}  
  
cpp{ }  
ocaml{ }
```



# Non-Idempotence

```
file{"file-1.txt"  
  ensure => file,  
  source => "file-2.txt"  
}
```

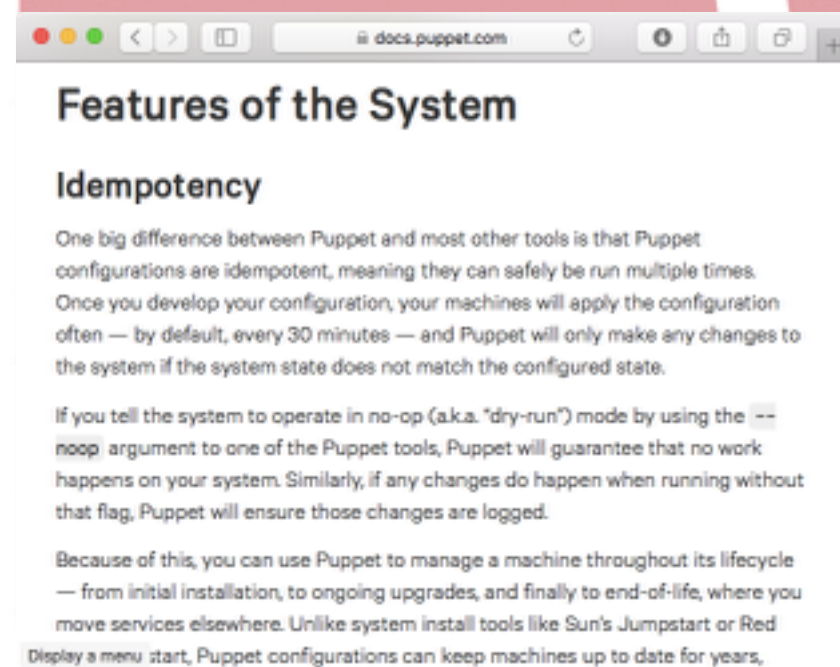
```
file{"file-2.txt":  
  ensure => absent  
}
```

```
File["file-1.txt"] -> File["file-2.txt"]
```



*"Why do we want configuration management? [...] We want this process to be repeatable, well-understood, and predictable. We want to make it deterministic."*

# WRONG



**Features of the System**

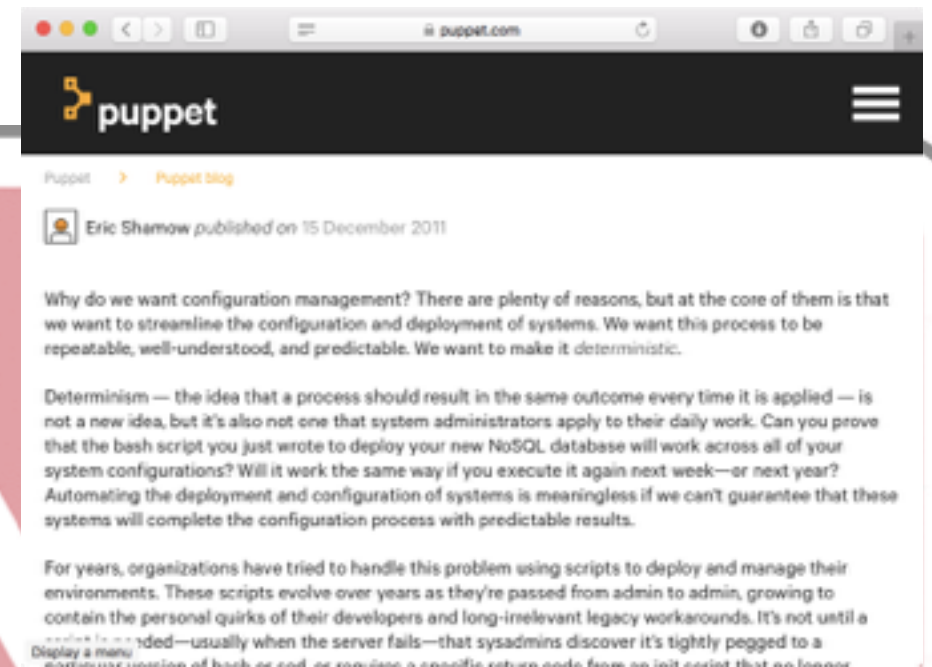
### Idempotency

One big difference between Puppet and most other tools is that Puppet configurations are idempotent, meaning they can safely be run multiple times. Once you develop your configuration, your machines will apply the configuration often — by default, every 30 minutes — and Puppet will only make any changes to the system if the system state does not match the configured state.

If you tell the system to operate in no-op (aka. "dry-run") mode by using the `--noop` argument to one of the Puppet tools, Puppet will guarantee that no work happens on your system. Similarly, if any changes do happen when running without that flag, Puppet will ensure those changes are logged.

Because of this, you can use Puppet to manage a machine throughout its lifecycle — from initial installation, to ongoing upgrades, and finally to end-of-life, where you move services elsewhere. Unlike system install tools like Sun's Jumpstart or Red Hat's Anaconda, Puppet configurations can keep machines up to date for years.

*"One big difference between Puppet and most other tools is that Puppet configurations are idempotent, meaning they can safely be run multiple times."*



**puppet**

Puppet > Puppet blog

Eric Sharrow published on 15 December 2011

Why do we want configuration management? There are plenty of reasons, but at the core of them is that we want to streamline the configuration and deployment of systems. We want this process to be repeatable, well-understood, and predictable. We want to make it deterministic.

Determinism — the idea that a process should result in the same outcome every time it is applied — is not a new idea, but it's also not one that system administrators apply to their daily work. Can you prove that the bash script you just wrote to deploy your new NoSQL database will work across all of your system configurations? Will it work the same way if you execute it again next week—or next year? Automating the deployment and configuration of systems is meaningless if we can't guarantee that these systems will complete the configuration process with predictable results.

For years, organizations have tried to handle this problem using scripts to deploy and manage their environments. These scripts evolve over years as they're passed from admin to admin, growing to contain the personal quirks of their developers and long-irrelevant legacy workarounds. It's not until a server is broken—usually when the server fails—that sysadmins discover it's tightly pegged to a specific version of bash, or perl, or even a specific version of a specific library. It's not until a

# Type Reference

## Generated References

These pages are generated from the puppet source code. You are looking at references for:

### Puppet 4.2.0

#### Type Reference

[Configuration Reference](#)

[Function Reference](#)

[Metaparameter Reference](#)

[Report Reference](#)

[Direction Reference](#)

[Developer Documentation](#)

[Plan Pages](#)

### Other Versions

[Latest \(newest official release\)](#)

[Latest \(includes pre-releases\)](#)

[Current 4.x release](#)

[Current 3.x release](#)

[Click here for all historical versions.](#)

### Puppet Reference Manual

Display a menu

- [augeas](#)
- [computer](#)
- [cron](#)
- [exec](#)
- [file](#)
- [filebucket](#)
- [group](#)
- [host](#)
- [interface](#)
- [k5login](#)
- [macauthorization](#)
- [mailalias](#)
- [maillist](#)
- [mcx](#)
- [mount](#)
- [nagios\\_command](#)
- [nagios\\_contact](#)
- [nagios\\_contactgroup](#)
- [nagios\\_host](#)
- [nagios\\_hostdependency](#)
- [nagios\\_hostescalation](#)
- [nagios\\_hostextinfo](#)
- [nagios\\_hostgroup](#)
- [nagios\\_service](#)
- » [nagios\\_servicedependency](#)
- » [nagios\\_serviceescalation](#)
- » [nagios\\_serviceextinfo](#)
- » [nagios\\_servicegroup](#)
- » [nagios\\_timeperiod](#)
- » [notify](#)
- » [package](#)
- » [resources](#)
- » [router](#)
- » [schedule](#)
- » [scheduled\\_task](#)
- » [selboolean](#)
- » [selmodule](#)
- » [service](#)
- » [ssh\\_authorized\\_key](#)
- » [sshkey](#)
- » [stage](#)
- » [tidy](#)
- » [user](#)
- » [vlan](#)
- » [yumrepo](#)
- » [zfs](#)
- » [zone](#)
- » [zpool](#)

# Many, many resource types...

The screenshot shows a web browser window with the address bar displaying 'docs.puppetlabs.com'. The page title is 'Type Reference'. On the left, there is a 'Docs Quick Nav' sidebar with a hamburger menu icon. Below the sidebar, there is a section titled 'Generated References' with a paragraph explaining that these pages are generated from Puppet source code. Below this, there is a list of links for various reference types, including 'Configuration Reference', 'Function Reference', 'Letparameter Reference', 'Report Reference', 'Direction Reference', 'Developer Documentation', and 'Plan Pages'. There is also a section titled 'Other Versions' with links for 'stable (newest official release)', 'latest (includes pre-releases)', 'current 4.x release', 'current 3.x release', and a link to 'see here for all historical versions'. At the bottom of the sidebar, there is a link to the 'Puppet Reference Manual' and a button that says 'Display a menu'. The main content area of the page is titled 'Type Reference' and contains a list of resource types, including 'augeas', 'computer', 'cron', 'exec', 'file', 'filebucket', 'group', 'host', 'interface', 'k5login', 'macauthorization', 'mailalias', 'maillist', 'mcx', 'mount', 'nagios\_command', 'nagios\_contact', 'nagios\_contactgroup', 'nagios\_host', 'nagios\_hostdependency', 'nagios\_hostescalation', 'nagios\_hostextinfo', 'nagios\_hostgroup', and 'nagios\_service'. The list continues with 'nagios\_servicedependency', 'nagios\_serviceescalation', 'nagios\_serviceextinfo', 'nagios\_servicegroup', 'nagios\_timeperiod', 'notify', 'package', 'resources', 'router', 'schedule', 'scheduled\_task', 'selboolean', 'selmodule', 'service', 'ssh\_authorized\_key', 'sshkey', 'stage', 'tidy', 'user', 'vlan', 'yumrepo', 'zfs', 'zone', and 'zpool'.

docs.puppetlabs.com

Docs Quick Nav

## Type Reference

### Generated References

These pages are generated from the Puppet source code. You are looking at references for:

**Puppet 4.2.0**

- Type Reference
- Configuration Reference
- Function Reference
- Letparameter Reference
- Report Reference
- Direction Reference
- Developer Documentation
- Plan Pages

**Other Versions**

- stable (newest official release)
- latest (includes pre-releases)
- current 4.x release
- current 3.x release
- see here for all historical versions.

**Puppet Reference Manual**

Display a menu

- augeas
- computer
- cron
- exec
- file
- filebucket
- group
- host
- interface
- k5login
- macauthorization
- mailalias
- maillist
- mcx
- mount
- nagios\_command
- nagios\_contact
- nagios\_contactgroup
- nagios\_host
- nagios\_hostdependency
- nagios\_hostescalation
- nagios\_hostextinfo
- nagios\_hostgroup
- nagios\_service
- » nagios\_servicedependency
- » nagios\_serviceescalation
- » nagios\_serviceextinfo
- » nagios\_servicegroup
- » nagios\_timeperiod
- » notify
- » package
- » resources
- » router
- » schedule
- » scheduled\_task
- » selboolean
- » selmodule
- » service
- » ssh\_authorized\_key
- » sshkey
- » stage
- » tidy
- » user
- » vlan
- » yumrepo
- » zfs
- » zone
- » zpool

## Puppet Language Features by Release

| Feature  | 0.23.x | 0.24.x     | 0.25.x | 2.6.x                     | 2.7.0              | 3.x | 3.2.x      | 3.4.x      |
|--|--------|------------|--------|---------------------------|--------------------|-----|------------|------------|
| Dynamic variable scope (declaring classes/resources assigns parent scope)                                | X      | X          | X      | X                         | X (deprecated)     |     |            |            |
| Appending to attributes in class inheritance (+>)  | X      | X          | X      | X                         | X                  | X   | X          | X          |
| Multi-line C-style comments  |        | X (0.24.7) | X      | X                         | X                  | X   | X          | X          |
| Arrays of resource references allowed in relationships   |        | X          | X      | X                         | X                  | X   | X          | X          |
| Overrides in class inheritance   |        | X          | X      | X                         | X                  | X   | X          | X          |
| Appending to variables in child scopes (+=)  |        | X          | X      | X                         | X                  | X   | X          | X          |
| Class names starting with 0-9  |        | X          | X      | X                         |                    |     |            |            |
| Regular expressions in node definitions  |        |            | X      | X                         | X                  | X   | X          | X          |
| Assigning expressions to variables   |        |            | X      | X                         | X                  | X   | X          | X          |
| Regular expressions in conditionals/expresions   |        |            | X      | X                         | X                  | X   | X          | X          |
| <code>elsif</code> in if statements  |        |            |        | X                         | X                  | X   | X          | X          |
| Chaining Resources   |        |            |        | X                         | X                  | X   | X          | X          |
| Hashes   |        |            |        | X (partial until 2.6.7)** | X                  | X   | X          | X          |
| Class Parameters   |        |            |        | X                         | X                  | X   | X          | X          |
| Run Stages   |        |            |        | X                         | X                  | X   | X          | X          |
| The "in" operator  |        |            |        | X                         | X                  | X   | X          | X          |
| <code>\$title</code> , <code>\$name</code> , and <code>\$module_name</code> available in parameter lists |        |            |        | X (2.6.5)                 | X                  | X   | X          | X          |
| Optional trailing comma in parameter lists   |        |            |        |                           | X (2.7.8)          | X   | X          | X          |
| Hyphens/dashes allowed in variable names *   |        |            |        |                           | X (2.7.3 - 2.7.14) |     |            |            |
| Automatic class parameter lookup via data bindings   |        |            |        |                           |                    | X   | X          | X          |
| "Unless" conditionals  |        |            |        |                           |                    | X   | X          | X          |
| Iteration over arrays and hashes   |        |            |        |                           |                    |     | X (future) | X (future) |
| The modulo (%) operator  |        |            |        |                           |                    |     | X          | X          |
| <code>\$trusted</code> hash  |        |            |        |                           |                    |     |            | X          |



## Puppet Language Features by Release

| Feature  | 0.23.x | 0.24.x     | 0.25.x | 2.6.x                     | 2.7.0              | 3.x | 3.2.x      | 3.4.x      |
|--|--------|------------|--------|---------------------------|--------------------|-----|------------|------------|
| Dynamic variable scope (declaring classes/resources assigns parent scope)                                | X      | X          | X      | X                         | X (deprecated)     |     |            |            |
| Appending to attributes in class inheritance (+>)  | X      | X          | X      | X                         | X                  | X   | X          | X          |
| Multi-line C-style comments  |        | X (0.24.7) | X      | X                         | X                  | X   | X          | X          |
| Arrays of resource references allowed in relationships   |        | X          | X      | X                         | X                  | X   | X          | X          |
| Overrides in class inheritance   |        | X          | X      | X                         | X                  | X   | X          | X          |
| Appending to variables in child scopes (+=)  |        | X          | X      | X                         | X                  | X   | X          | X          |
| Class names starting with 0-9  |        | X          | X      | X                         |                    |     |            |            |
| Regular expressions in node definitions  |        |            | X      | X                         | X                  | X   | X          |            |
| Assigning expressions to variables   |        |            | X      | X                         | X                  | X   | X          |            |
| Regular expressions in conditionals/expresions   |        |            | X      | X                         | X                  | X   | X          |            |
| <code>elsif</code> in if statements  |        |            |        | X                         | X                  | X   | X          |            |
| Chaining Resources   |        |            |        | X                         | X                  | X   | X          |            |
| Hashes   |        |            |        | X (partial until 2.6.7)** | X                  | X   | X          |            |
| Class Parameters   |        |            |        | X                         | X                  | X   | X          |            |
| Run Stages   |        |            |        | X                         | X                  | X   | X          |            |
| The "in" operator  |        |            |        | X                         | X                  | X   | X          |            |
| <code>\$title</code> , <code>\$name</code> , and <code>\$module_name</code> available in parameter lists |        |            |        | X (2.6.5)                 | X                  | X   | X          |            |
| Optional trailing comma in parameter lists   |        |            |        |                           | X (2.7.8)          | X   | X          | X          |
| Hyphens/dashes allowed in variable names *   |        |            |        |                           | X (2.7.3 - 2.7.14) |     |            |            |
| Automatic class parameter lookup via data bindings   |        |            |        |                           |                    | X   | X          | X          |
| "Unless" conditionals  |        |            |        |                           |                    | X   | X          | X          |
| Iteration over arrays and hashes   |        |            |        |                           |                    |     | X (future) | X (future) |
| The modulo (%) operator  |        |            |        |                           |                    |     | X          | X          |
| <code>\$trusted</code> hash  |        |            |        |                           |                    |     |            | X          |



# Simplifying Puppet



## Puppet

stages

virtual resources

"classes"

default values

expressions

conditionals

defined types

anchors

inheritance

single-assignment variables

...

# Simplifying Puppet



## Puppet

stages

virtual resources

"classes"

default values

expressions

conditionals

defined types

anchors

inheritance

single-assignment variables

...



## Resource Graph

resources

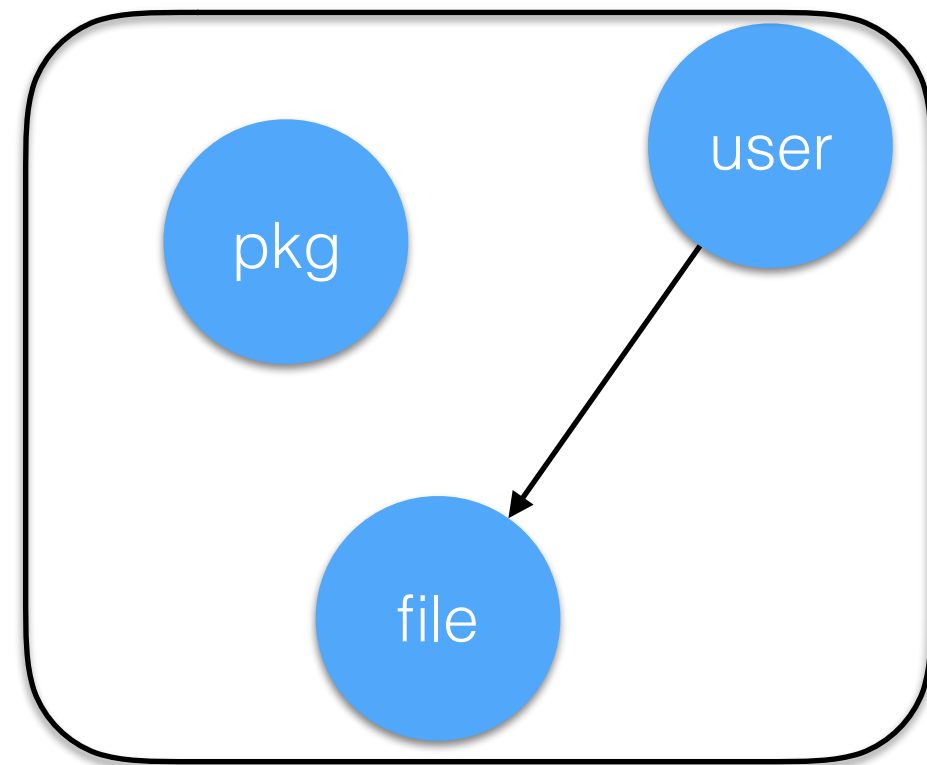
attributes

edges

Datalog

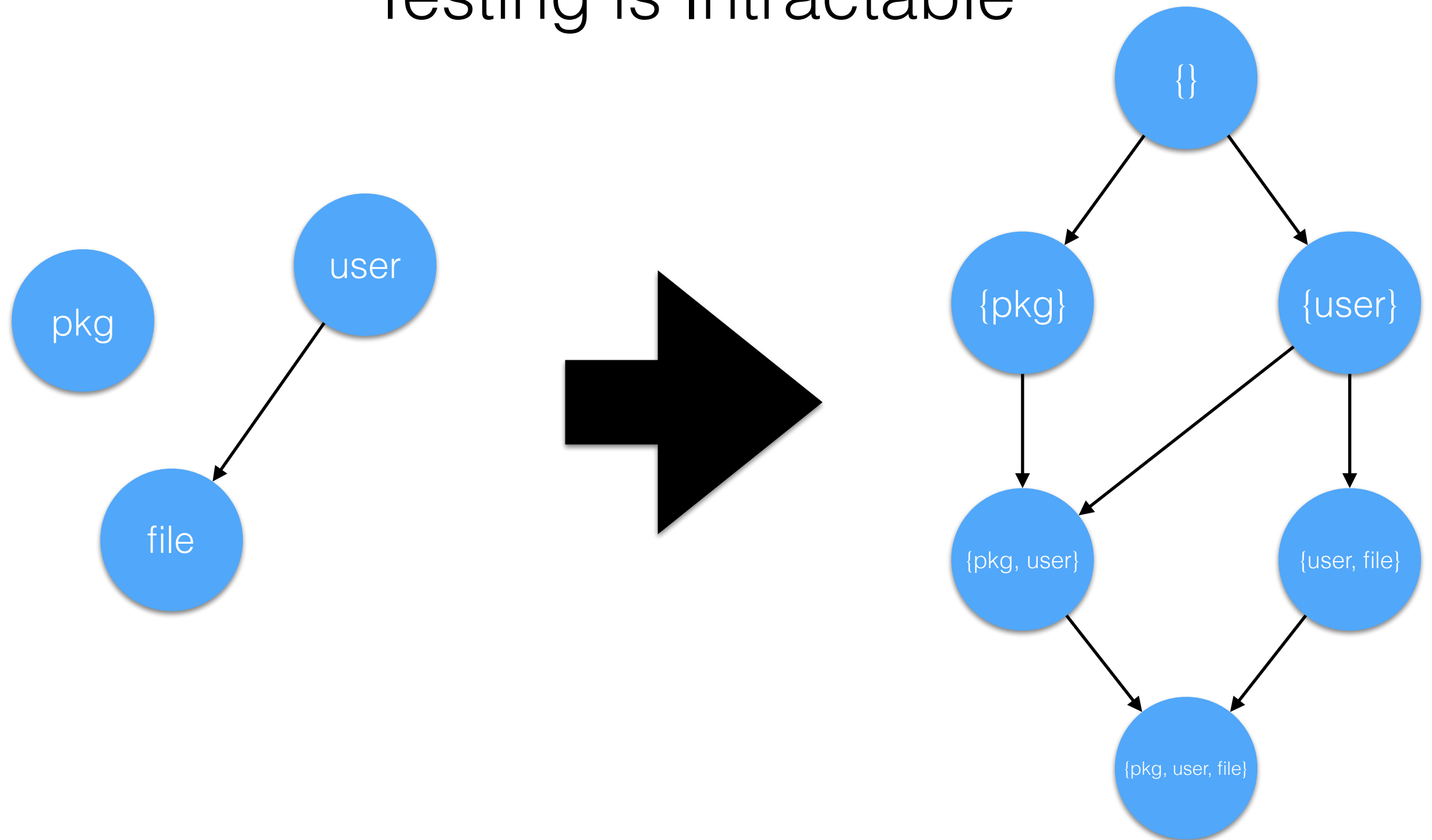
# Properties of ~~Puppet~~ Resource Graphs

1. Configuration application should be deterministic
2. Configuration application should be idempotent

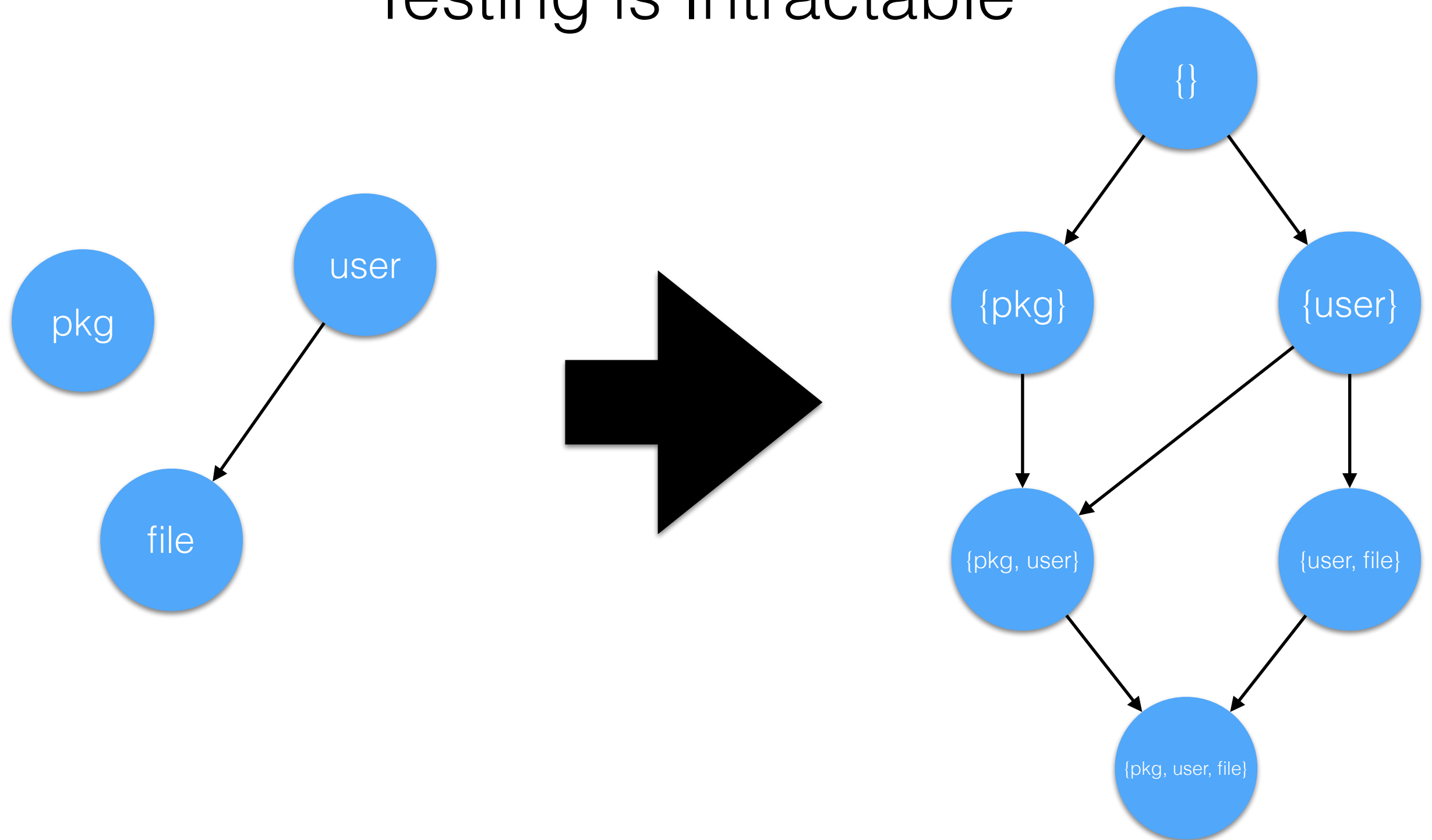




# Testing is Intractable

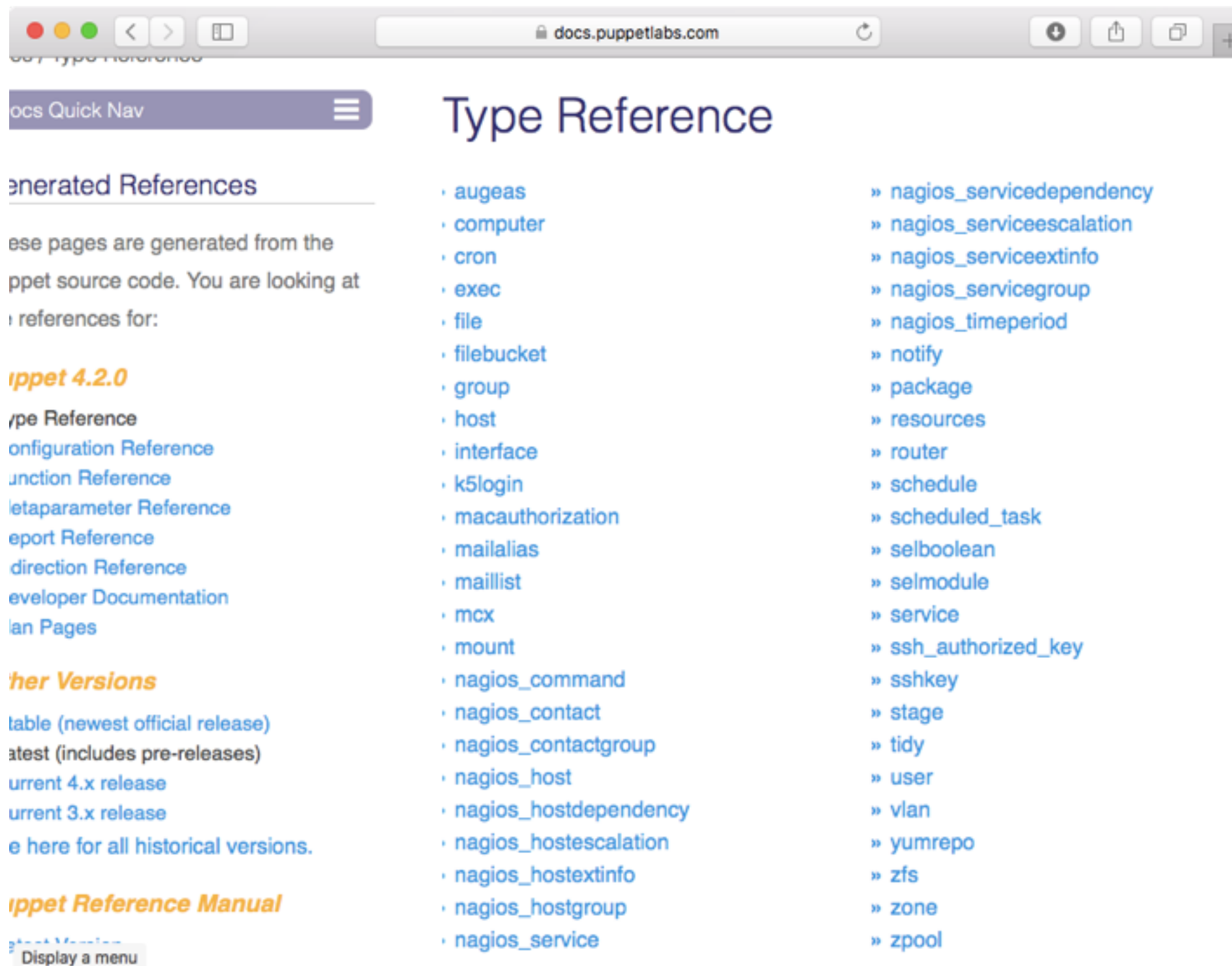


# Testing is Intractable

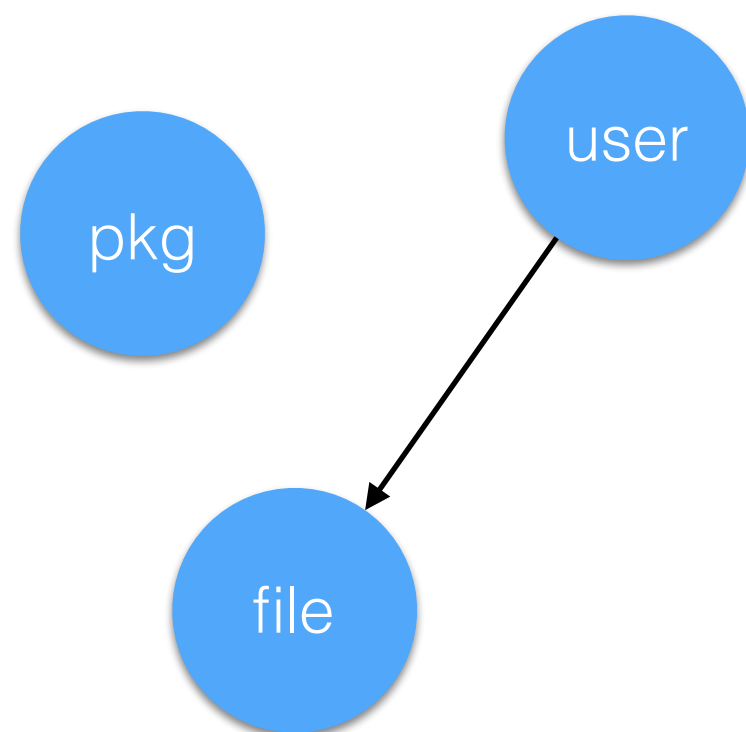


static analysis...

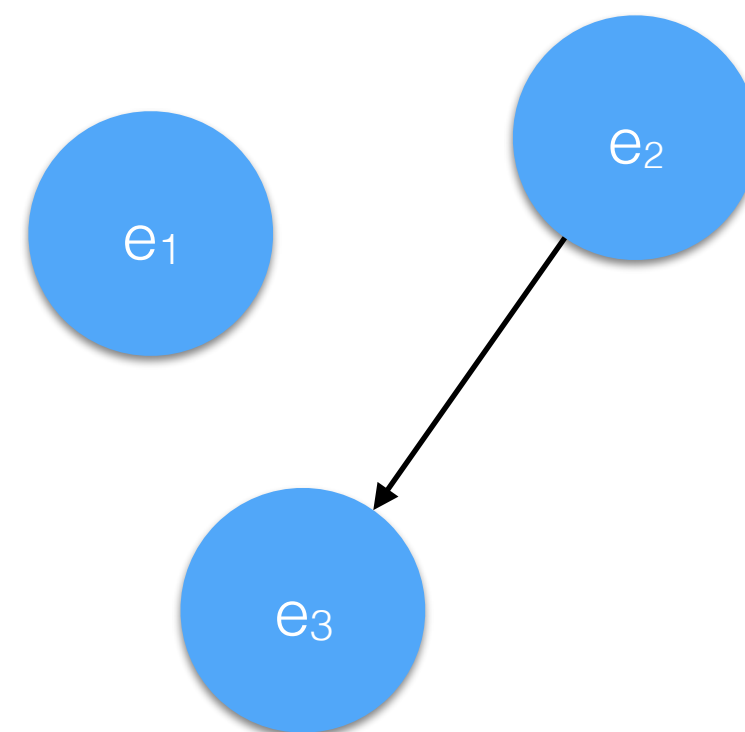
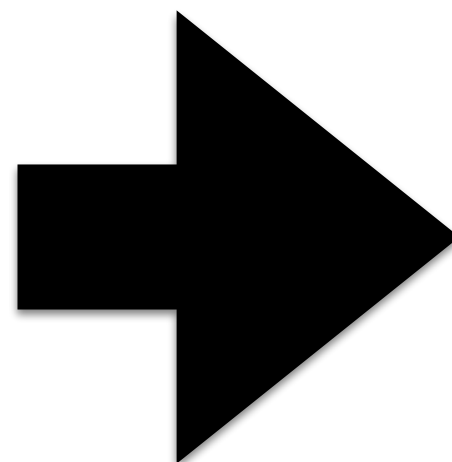
# All resources mutate machine state.



## Endless possible interactions...



Resources



File-system  
manipulating programs

```
file{"/home/arjun/.ssh/auth_keys":  
  content => "ssh-rsa ..."  
}
```

```
if (isFile("/home/arjun/.ssh/auth_keys")) {  
  rm("/home/arjun/.ssh/auth_keys")  
}  
creat("/home/arjun/.ssh/auth_keys", "ssh-rsa ...")
```

```
file{"/home/arjun/.ssh/auth_keys":  
  content => "ssh-rsa ..."  
}
```

```
if (isFile("/home/arjun/.ssh/auth_keys")) {  
  rm("/home/arjun/.ssh/auth_keys")  
}  
creat("/home/arjun/.ssh/auth_keys", "ssh-rsa ...")
```

```
file{"/home/arjun/.ssh/auth_keys":  
  content => "ssh-rsa ...",  
  replace => false  
}
```

```
if (!isFile("/home/arjun/.ssh/auth_keys")) {  
  creat("/home/arjun/.ssh/auth_keys", "ssh-rsa ...")  
}
```

```
file{"/home/arjun/.ssh/auth_keys":  
  content => "ssh-rsa ..."  
}
```

```
if (isFile("/home/arjun/.ssh/auth_keys")) {  
  rm("/home/arjun/.ssh/auth_keys")  
}  
creat("/home/arjun/.ssh/auth_keys", "ssh-rsa ...")
```

```
file{"/home/arjun/.ssh/auth_keys":  
  content => "ssh-rsa ...",  
  replace => false  
}
```

```
if (!isFile("/home/arjun/.ssh/auth_keys")) {  
  creat("/home/arjun/.ssh/auth_keys", "ssh-rsa ...")  
}
```

```
file{"/home/arjun/.ssh/auth_keys":  
  content => "ssh-rsa ...",  
  force    => true  
}
```

```
if (!isDir("/home")) {  
  mkdir("/home")  
}  
if (!isDir("/home/arjun")) {  
  mkdir("/home/arjun")  
}  
if (!isDir("/home/arjun.ssh")) {  
  mkdir("/home/arjun/.ssh")  
}  
  
if (isFile("/home/arjun/.ssh/auth_keys")) {  
  rm("/home/arjun/.ssh/auth_keys")  
}  
creat("/home/arjun/.ssh/auth_keys", "ssh-rsa ...")
```



repoquery -l vim-enhanced

```
package{"vim-enhanced":  
  ensure => present  
}
```

/usr/bin/vimtutor  
/usr/bin/vimdiff  
/usr/bin/vim  
/usr/bin/rvim  
/etc/profile.d/vim.sh  
/etc/profile.d/vim.csh

...  
→ creat("/usr/bin/vim", ...)   
...

apt-file -F list vim-enhanced



Unique value  
for each file

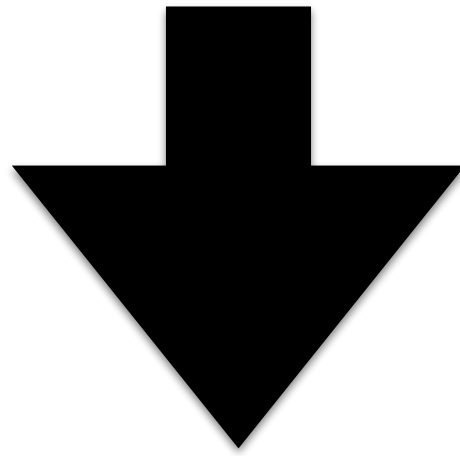


$\llbracket e \rrbracket: \text{filesystem} \rightarrow \text{filesystem} + \text{error}$

$\llbracket G \rrbracket: \text{filesystem} \rightarrow 2^{\text{filesystem}} + \text{error}$

Determinism:  $\forall \sigma . |\llbracket G \rrbracket \sigma| = 1$

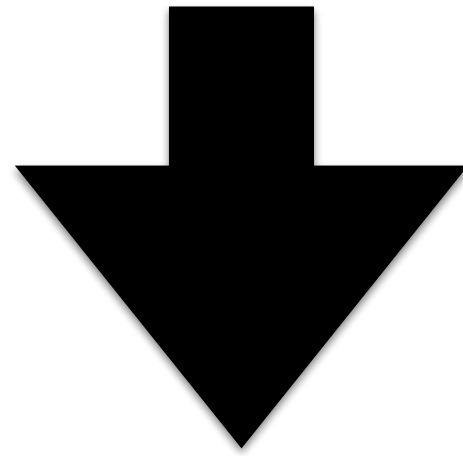
$$\llbracket G \rrbracket: filesystem \rightarrow 2^{filesystem} + \text{error}$$



$$R_G \in filesystem \times (filesystem + \text{error})$$

Non-Determinism:  $\exists \sigma_1 \sigma_2 \sigma_3 . \sigma_1 R_G \sigma_2 \ \&\& \ \sigma_1 R_G \sigma_3 \ \&\& \ \sigma_2 \neq \sigma_3$

$$\llbracket G \rrbracket: filesystem \rightarrow 2^{filesystem} + \text{error}$$



$$R_G \in filesystem \times (filesystem + \text{error})$$

Non-Determinism:  $\exists \sigma_1 \sigma_2 \sigma_3 . \sigma_1 R_G \sigma_2 \ \&\& \ \sigma_1 R_G \sigma_3 \ \&\& \ \sigma_2 \neq \sigma_3$

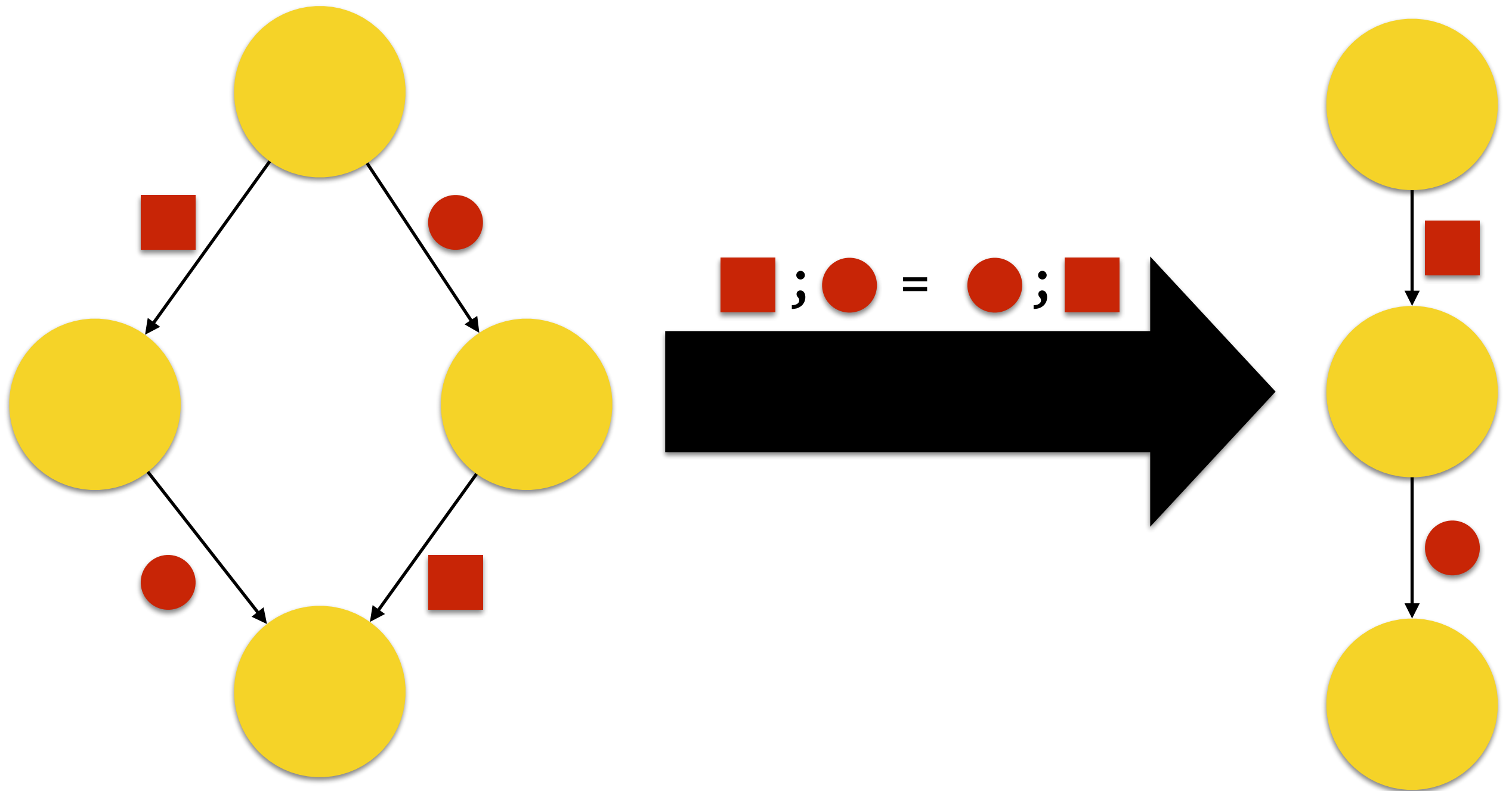
Finite state?

Each state size  
is exponential

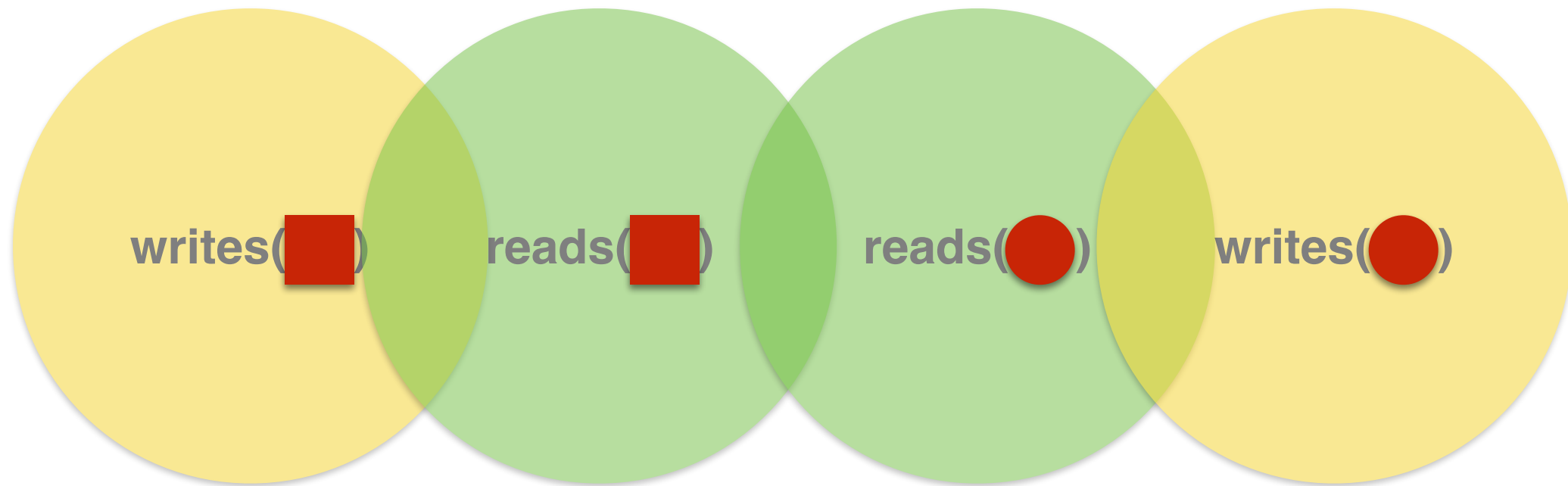
Formula size is  
exponential

> 1 billion variables

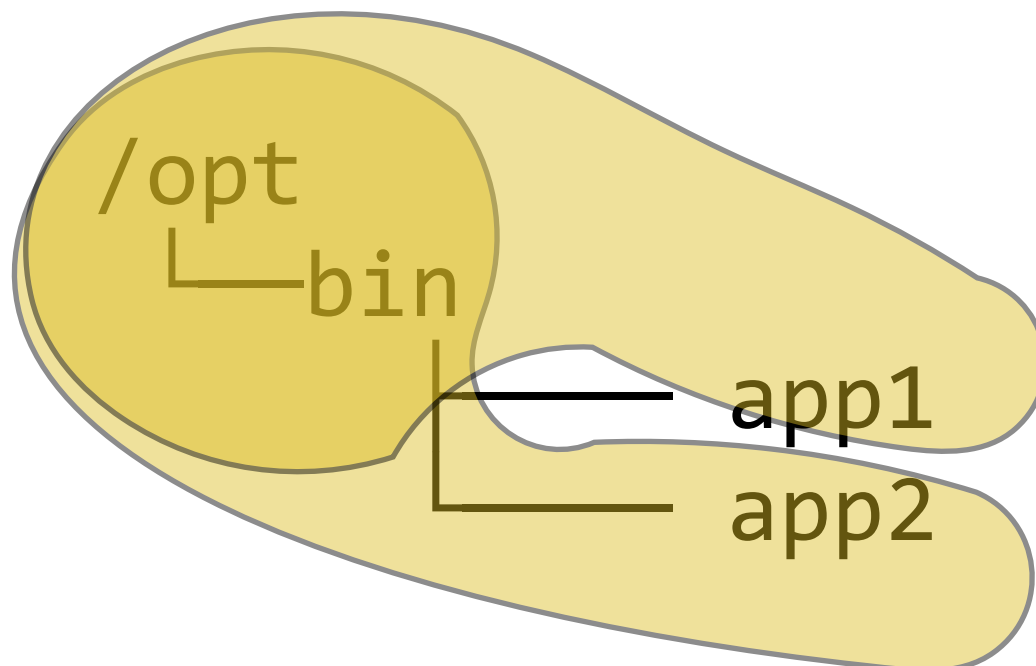
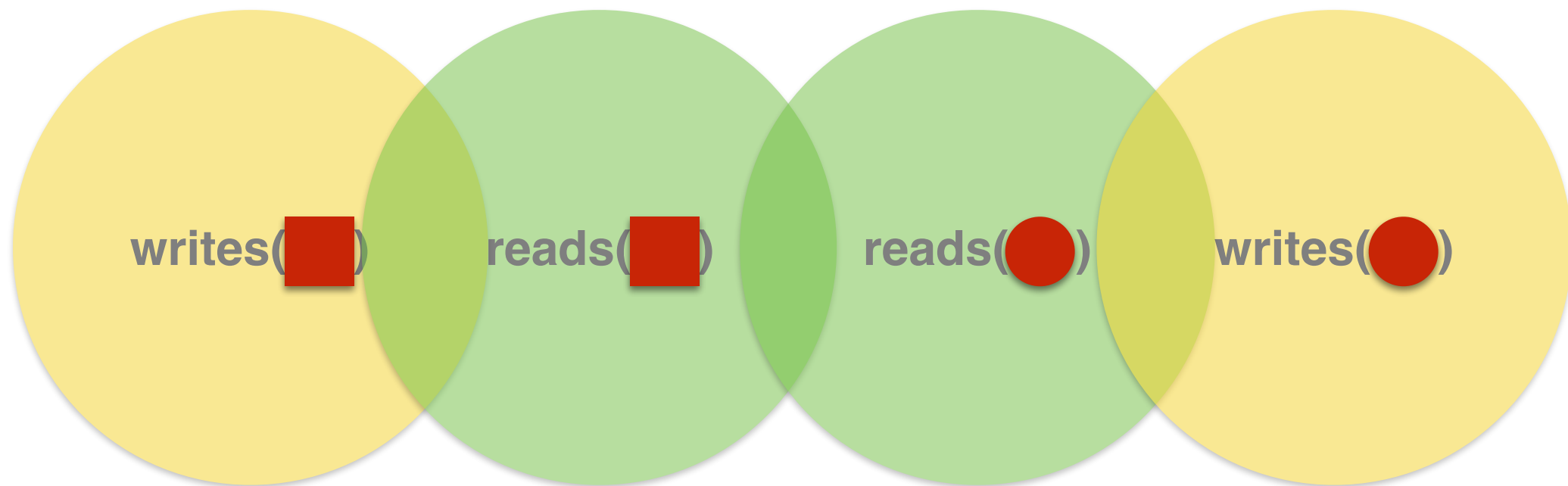
# Partial-order reduction



# Typical commutativity check

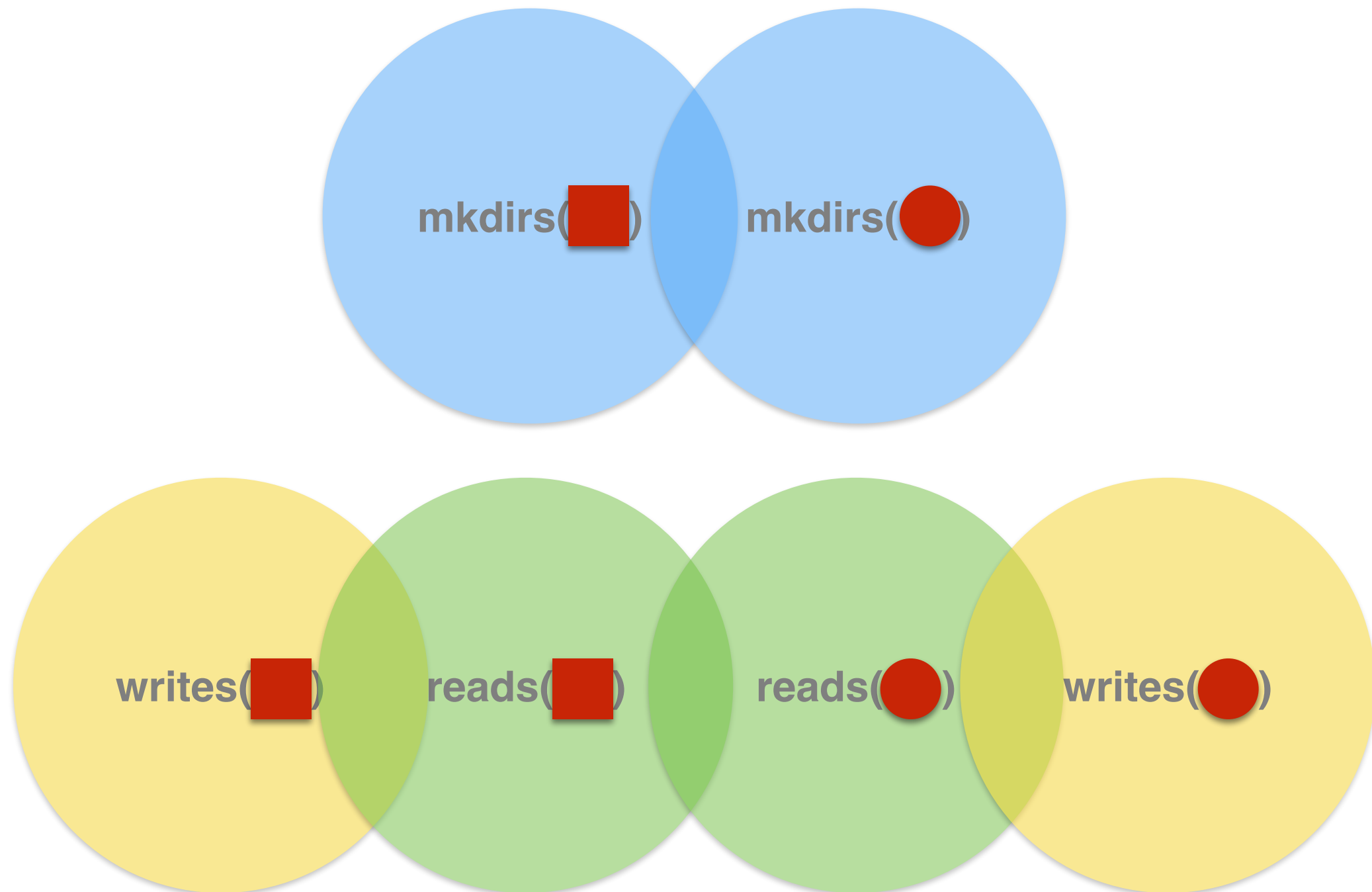


# Typical commutativity check

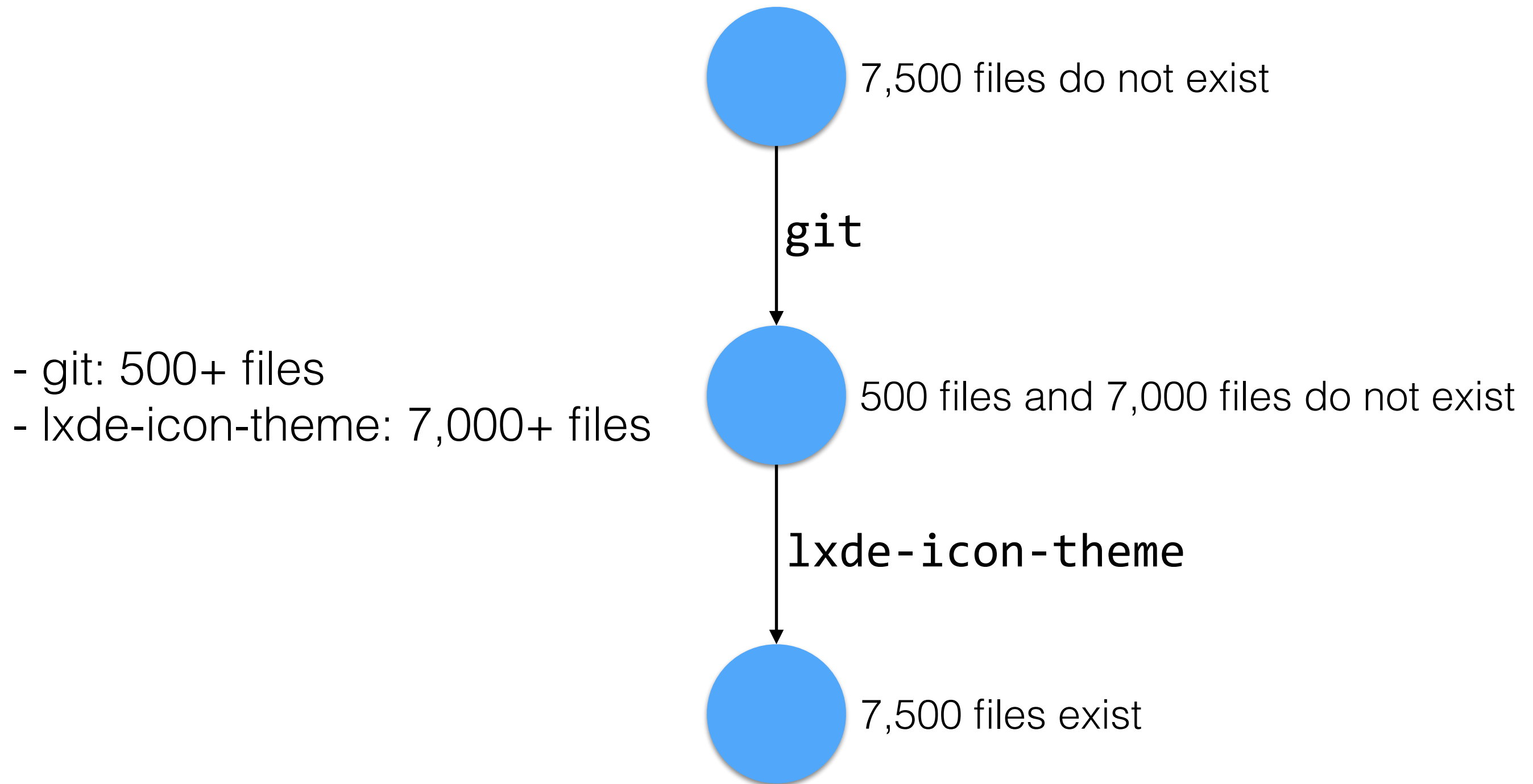


packages may need to create  
overlapping directories

# Our commutativity check



# States are massive

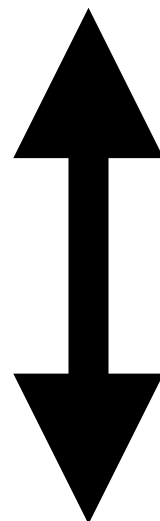




$$\llbracket \blacksquare ; \bullet \rrbracket \sigma \neq \llbracket \bullet ; \blacksquare \rrbracket \sigma$$

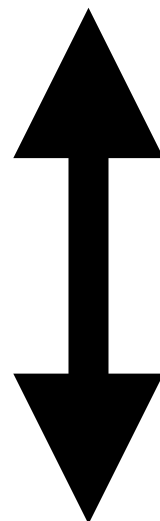
$$\llbracket \text{[red square with blue triangle]} ; \text{[red circle]} \rrbracket \sigma \neq \llbracket \text{[red circle]} ; \text{[red square with blue triangle]} \rrbracket \sigma$$

$$\llbracket \boxed{\triangle} ; \bigcirc \rrbracket \sigma \neq \llbracket \bigcirc ; \boxed{\triangle} \rrbracket \sigma$$

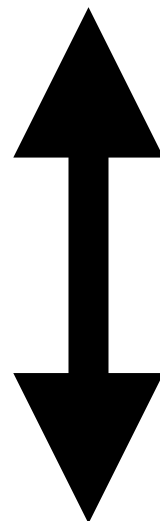


$$\llbracket \boxed{\triangle} ; \bigcirc ; \triangle \rrbracket \sigma \neq \llbracket \bigcirc ; \boxed{\triangle} ; \triangle \rrbracket \sigma$$

$$\llbracket \boxed{\triangle} ; \bullet \rrbracket \sigma \neq \llbracket \bullet ; \boxed{\triangle} \rrbracket \sigma$$




$$\llbracket \boxed{\triangle} ; \bullet ; \triangle \rrbracket \sigma \neq \llbracket \bullet ; \boxed{\triangle} ; \triangle \rrbracket \sigma$$



$$\llbracket \boxed{\triangle} ; \bullet \rrbracket \sigma \neq \llbracket \bullet ; \boxed{\triangle} \rrbracket \sigma$$

```
package{"apache2":  
  ensure => present  
}
```

```
file{"/etc/apache2/apache2.conf":  
  content => ...  
}
```



apache2.conf  
is *not* pruned  
from the  
package

```
package{"apache2":  
  ensure => present  
}  
  
file{"/etc/apache2/apache2.conf":  
  content => ...  
}
```

apache2.conf  
is *not* pruned  
from the  
package

```
package{"apache2":  
  ensure => present  
}  
  
file{"/etc/apache2/apache2.conf":  
  content => ...  
}
```

/home/arjun is  
*not* pruned

```
file{"/home/arjun/.ssh/auth_keys":  
  content => "ssh-rsa ..."  
}  
  
user{"arjun":  
  ensure => present  
}  
  
user{"rian":  
  ensure => present  
}
```

/home/rian is  
pruned

Finite state

$$\exists \sigma. \llbracket p \rrbracket \sigma \neq \llbracket q \rrbracket \sigma$$



Finite state

$$\exists \sigma. \llbracket p \rrbracket \sigma \neq \llbracket q \rrbracket \sigma$$

```
p = if (isFile(/a)) { rm(/a) }; creat(/a, "...")  
q = creat(/a, "...")
```

$$\textit{dom}(\sigma) = \{ /a \}$$

i.e., all paths that  
appear in p and q

# Finite state

$$\exists \sigma. \llbracket p \rrbracket \sigma \neq \llbracket q \rrbracket \sigma$$

```
p = if (isFile(/a)) { rm(/a) }; creat(/a, "...")  
q = creat(/a, "...")
```

$$\textit{dom}(\sigma) = \{ /a \}$$

i.e., all paths that  
appear in p and q

```
p' = if (isDir(/a)) { skip } else { error }  
q' = if (isEmptyDir(/a)) { skip } else { error }
```

Demo

```
382     managehome => true,
383     shell      => '/usr/bin/irssi',
384     require    => Class['irssi'],
385 }
386
387 file { "${username}_irssi_dir":
388     ensure => $dir_ensure,
389     path   => "/home/${username}/.irssi",
390     mode   => '0755',
391     owner  => $username,
392     group  => $username,
393     require => User[$username],
394 }
395
396 file { "${username}_irssi_config":
397     ensure => $file_ensure,
398     path   => "/home/${username}/.irssi/config",
399     content => template('users/irssi_config'),
400     require => File["${username}_irssi_dir"],
401 }
402
403 ssh_authorized_key { "${username}_key":
404     ensure => $ensure,
405     key     => $key,
406     type    => $type,
407     user    => $username,
408     require => User[$username],
409 }
410 }
```

Enter postcondition (optional)

CentOS 6

Run Rehearsal

Display a menu

```
388     ensure => $dir_ensure,
389     path   => "/home/${username}/.irssi",
390     mode   => '0755',
391     owner  => $username,
392     group  => $username,
393     require => User[$username],
394 }
395
396 file { "${username}_irssi_config":
397     ensure => $file_ensure,
398     path   => "/home/${username}/.irssi/config",
399     content => template('users/irssi_config'),
400     require => File["${username}_irssi_dir"],
401 }
402
403 ssh_authorized_key { "${username}_key":
404     ensure => $ensure,
405     key    => $key,
406     type   => $type,
407     user   => $username,
408     require => User[$username],
409 }
410 }
```

Enter postcondition (optional)

CentOS 6

Run Rehearsal

Checking if manifest is deterministic ... FAILED.  
These dependencies may be missing:  
File["sshd\_config"] -> Package["openssh-server"]  
Package["collectd-rrdtool"] -> File["collectd\_swap"]

Display a menu

```
389     path    => "/home/${username}/.irssi",
390     mode    => '0755',
391     owner   => $username,
392     group   => $username,
393     require => User[$username],
394 }
395
396 file { "${username}_irssi_config":
397     ensure => $file_ensure,
398     path   => "/home/${username}/.irssi/config",
399     content => template('users/irssi_config'),
400     require => File["${username}_irssi_dir"],
401 }
402
403 ssh_authorized_key { "${username}_key":
404     ensure => $ensure,
405     key    => $key,
406     type   => $type,
407     user   => $username,
408     require => User[$username],
409 }
410 }
411
412 File["sshd_config"] -> Package["openssh-server"]
413 Package["collectd-rrdtool"] -> File["collectd_swap"]
```

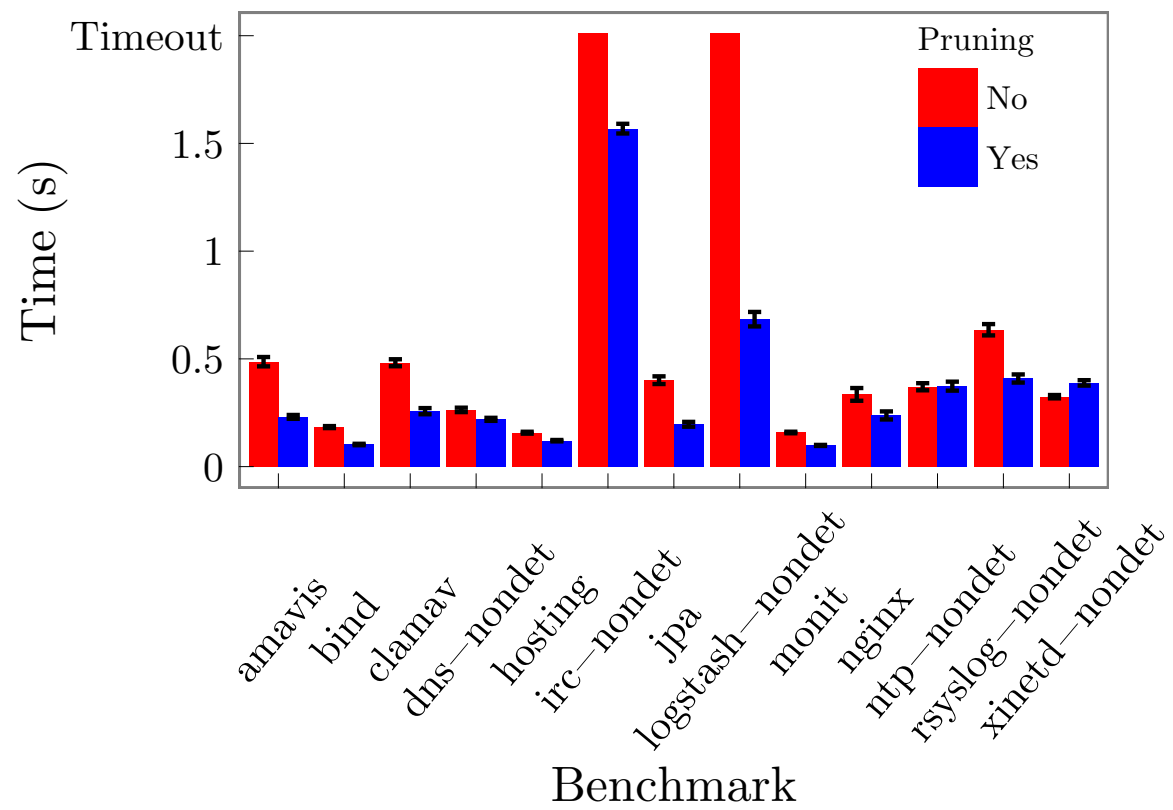
Enter postcondition (optional)

CentOS 6

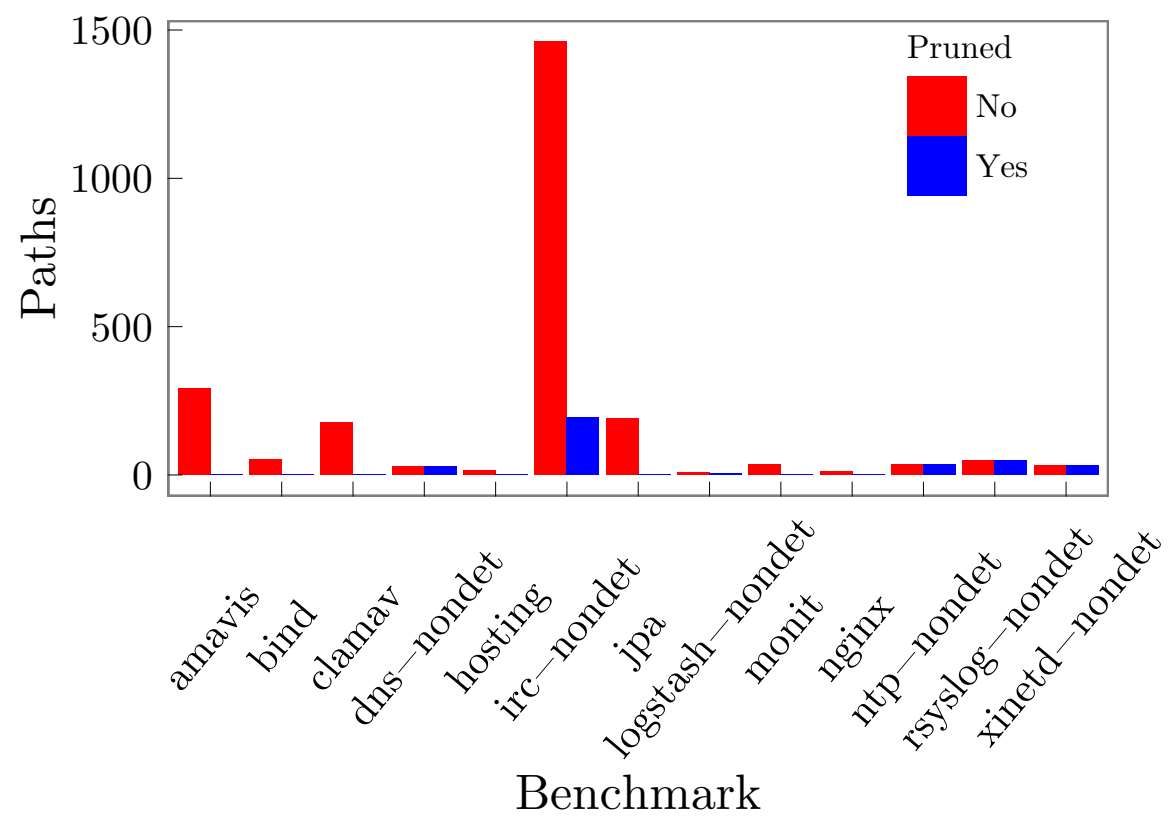
Run Rehearsal

Checking if manifest is deterministic ... no errors found.  
Checking if manifest is idempotent ... OK.

Display a menu

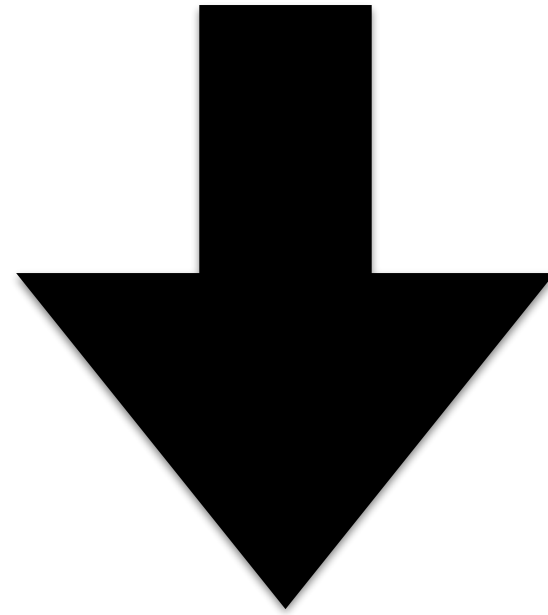


Optimizations are essential for checking determinism



Optimizations eliminate "uninteresting" paths

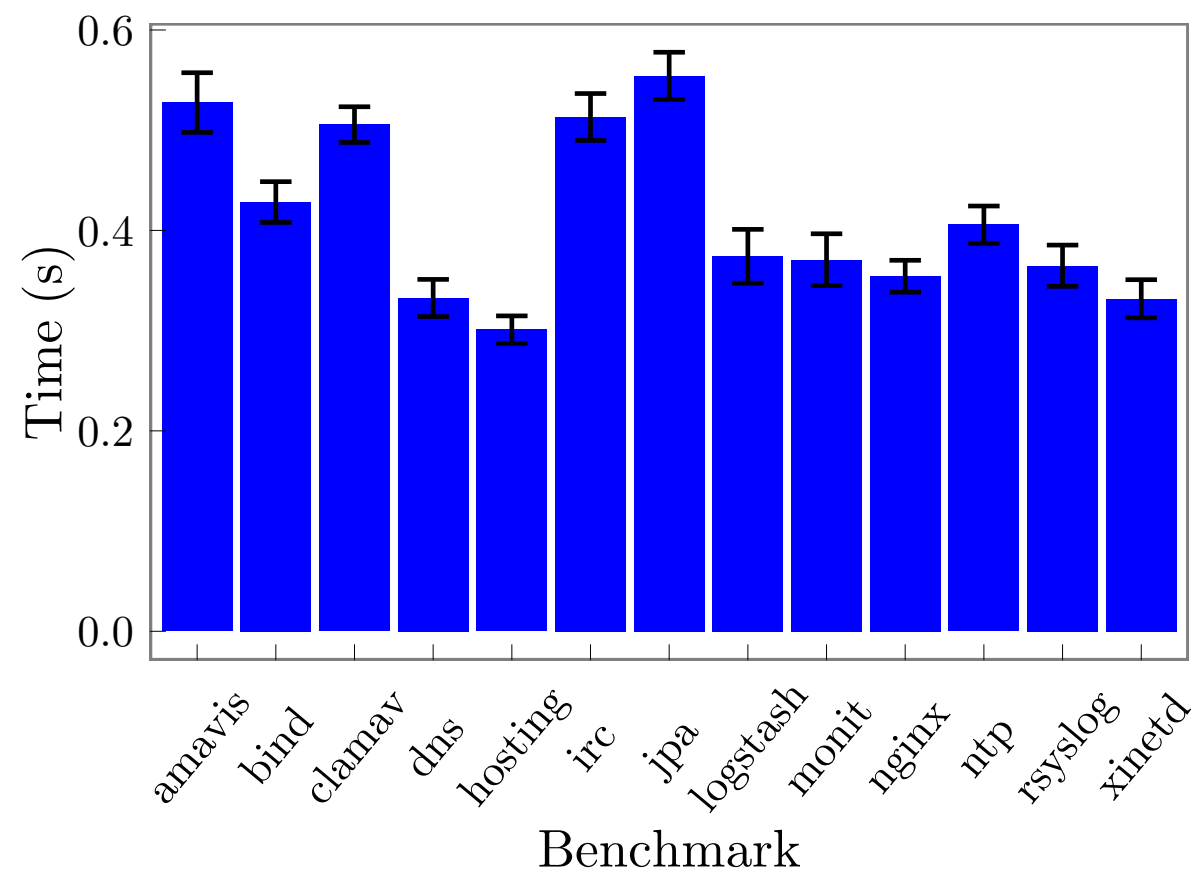
$\llbracket G \rrbracket: filesystem \rightarrow 2^{filesystem} + \text{error}$



If  $G$  is provably deterministic

$\llbracket G \rrbracket: filesystem \rightarrow filesystem + \text{error}$





Checking idempotence is fast  
(after checking determinism)



- ✓ Deterministic
- ✓ Idempotent

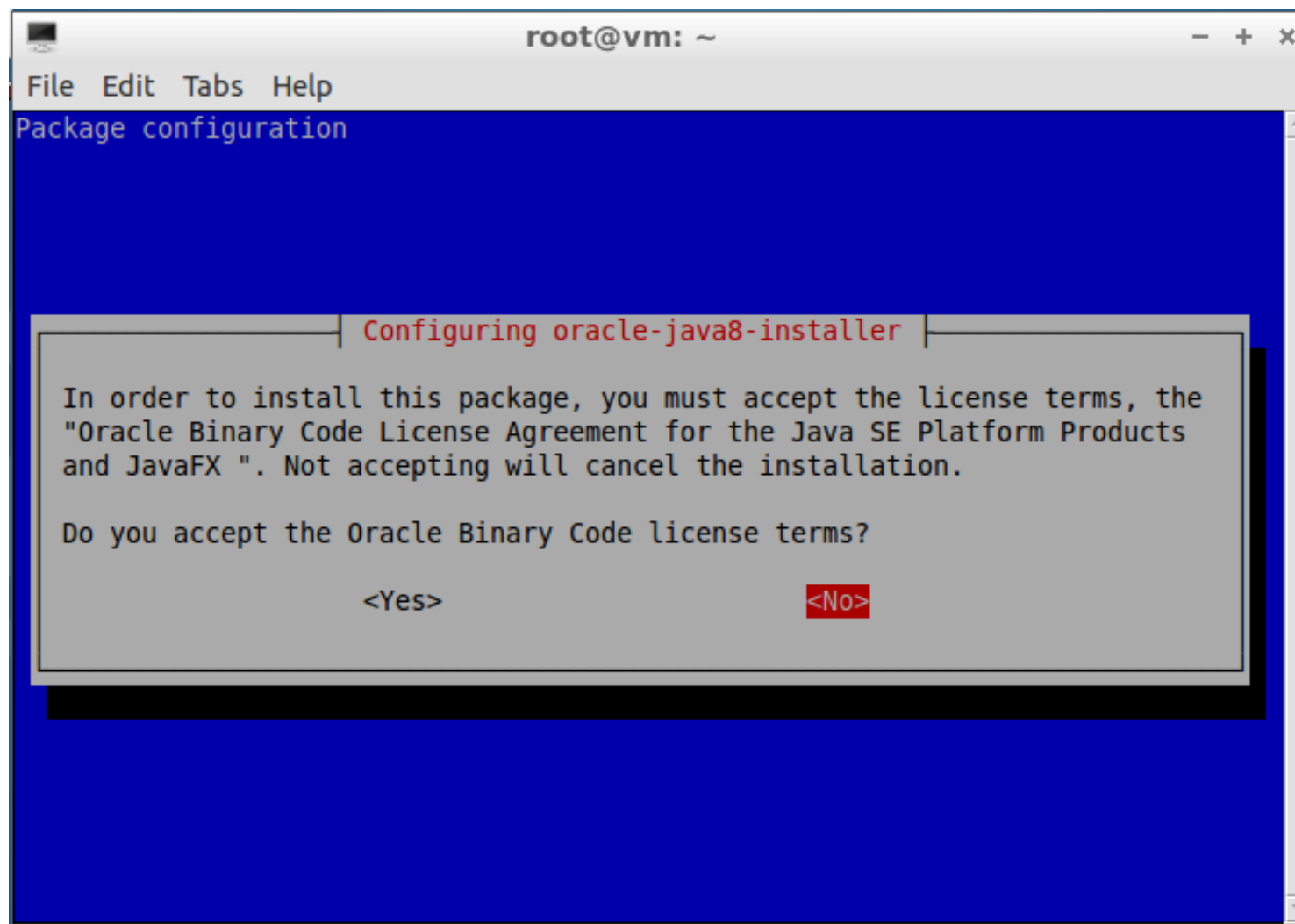


# Task: Install the Oracle JDK

```
add-apt-repository ppa:webupd8team/java  
apt-get update  
apt-get install oracle-java7-installer
```

```
apt::ppa{"ppa:webupd8team/java": }
```

```
package{"oracle-java7-installer":  
    require => apt::ppa["ppa:webupd8team/java"]  
}
```



Stack Overflow is a community of 4.7 million programmers, just like you, helping each other. Join them, it only takes a minute:

## Auto yes to the License Agreement on sudo apt-get -y install oracle-java7-installer

Publish your app to Amazon and receive up to **\$700** in bundled Services\*

amazon appstore

Terms and Conditions apply

asked 2 years ago  
viewed 5447 times  
active 7 months ago

▲  
17  
▼  
★  
11

The Oracle Java package for Ubuntu interactively asks about the License Agreement. So I have to say 'OK' and then 'yes' every time, but I'd like to automate it. What I do is this:

```
sudo add-apt-repository -y ppa:webupd8team/java  
sudo apt-get update  
sudo apt-get -y install oracle-java7-installer
```

Is there a simple way to automate the agreement process without using expect?

ubuntu automation apt

share improve this question

edited Oct 10 '13 at 13:04

asked Oct 9 '13 at 15:09

stackoverflow

Have you joined



try this out:

44

```
sudo add-apt-repository -y ppa:webupd8team/java
sudo apt-get update
echo debconf shared/accepted-oracle-license-v1-1 select true | sudo debconf-set-selections
echo debconf shared/accepted-oracle-license-v1-1 seen true | sudo debconf-set-selections
sudo apt-get -y install oracle-java7-installer
```

running 3rd and 4th command on my debian 7.1 helps, so I think the same can help on ubuntu as well

share improve this answer

answered Oct 15 '13 at 21:04

 **Maxym**  
8,370 ● 2 ● 24 ● 40

It works fine. Thanks a lot! – [kjtanaka](#) Oct 20 '13 at 6:26

Is there a simple way to automate the agreement process without using expect?

ubuntu automation apt

share improve this question

edited Oct 10 '13 at 13:04

asked Oct 9 '13 at 15:09

```
apt::ppa{"ppa:webupd8team/java": }
```



```
exec{"accept-license":
```

```
  command => "echo debconf shared/accepted-oracle-license-v1-1 select true | \  
              sudo debconf-set-selections"
```

```
}
```

```
package{"oracle-java7-installer":
```

```
  require => [apt::ppa["ppa:webupd8team/java"], Exec["accept-license"]]
```

```
}
```

**Current Work:**  
Learning a finite-state  
model of shell scripts  
from I/O examples

```
apt::ppa{"ppa:webupd8team/java": }
```



```
exec{"accept-license":
```

```
  command => "echo debconf shared/accepted-oracle-license-v1-1 select true | \  
              sudo debconf-set-selections"
```

```
}
```

```
package{"oracle-java7-installer":
```

```
  require => [apt::ppa["ppa:webupd8team/java"], Exec["accept-license"]]
```

```
}
```



# Thank You

[www.plasma.cs.umass.edu/rehearsal](http://www.plasma.cs.umass.edu/rehearsal)



- ✓ Determinism checking
- ✓ Idempotence checking
- ✓ Pre- and post- conditions

Rian Shambaugh  
Aaron Weiss  
Arjun Guha

