Package managing for Coq

Current state

I would like to find a library for foo

Maybe it exists in the Coq contribs?

Let's google it then…

Cool, a tarball/svn repo/git repo/debian package

Does it work for my version of Coq?

Let's figure out how to install it

Problems

- No centralized repository of packages
- No unified build system
- Monolithic packages (e.g. ssreflect, CompCert, CoLoR, . . .)
- No easy way to have multiple Coq installs

symlink, make install, and so on

Plan A

Implement a package manager from scratch

Plan A

- Time consuming
- Error prone
- Not “science”

PLAN B

Forking an exisiting package manager: opam

- opam knows of (ocaml) 'compilers' and 'packages'
- search and replace:
  - opam with cpam
  - ocaml with coq
- What if the OCaml version of the system changes?
- What about dependencies between Coq packages (plugins) and OCaml packages?

- opam knows of (ocaml) 'compilers' and 'packages'
- search and replace:
  - opam with cpam
  - ocaml with coq
- What if the OCaml version of the system changes?
- What about dependencies between Coq packages (plugins) and OCaml packages?

- opam knows of (ocaml) 'compilers' and 'packages'
- search and replace:
  - opam with cpam
  - ocaml with coq
- What if the OCaml version of the system changes?
- What about dependencies between Coq packages (plugins) and OCaml packages?

Plan C

Modify an existing package manager: opam

Opam primer

- Each 'state' is stored in one `$opam/$switch`
- Each 'state' has its own OCaml compiler and set of packages
- `opam switch 4.01.0` will:
  - create a new switch if needed
  - install the ocaml compiler version 4.01.0
  - update the configuration env. if the switch already exists
- `opam install foobar` will:
  - lookup for the available versions of package `foobar`;
  - select the highest version available for the current compiler;
  - update other packages if needed

Powerful system to resolve dependencies between packages

Opam primer

- Each 'state' is stored in one `$opam/$switch`
- Each 'state' has its own OCaml compiler and set of packages
- `opam switch 4.01.0` will:
  - create a new switch if needed
  - install the ocaml compiler version 4.01.0
  - update the configuration env. if the switch already exists
- `opam install foobar` will:
  - lookup for the available versions of package `foobar`;
  - select the highest version available for the current compiler;
  - update other packages if needed

Powerful system to resolve dependencies between packages

Opam primer

- Each 'state' is stored in one `$opam/$switch`
- Each 'state' has its own OCaml compiler and set of packages
- `opam switch 4.01.0` will:
  - create a new switch if needed
  - install the ocaml compiler version 4.01.0
  - update the configuration env. if the switch already exists
- `opam install foobar` will:
  - lookup for the available versions of package `foobar`;
  - select the highest version available for the current compiler;
  - update other packages if needed

Powerful system to resolve dependencies between packages

Opam primer

- Each 'state' is stored in one `$opam/$switch`
- Each 'state' has its own OCaml compiler and set of packages
- `opam switch 4.01.0` will:
  - create a new switch if needed
  - install the ocaml compiler version 4.01.0
  - update the configuration env. if the switch already exists
- `opam install foobar` will:
  - lookup for the available versions of package `foobar`;
  - select the highest version available for the current compiler;
  - update other packages if needed

Powerful system to resolve dependencies between packages

Prototype

New commands for opam.

- `opam coq install coq.8.4.2` will:
  - create a new opam switch named `myocamlversion--coq.8.4.2`
  - install Coq 8.4pl2 in this switch (with its dependency on camlp5)
- `opam coq switch coq.HoTT` will:
  - browse the existing switches to find one with Coq HoTT;
  - if there is exactly one, will 'switch' to it;
  - if there is more than one, will tell the user to select one;
  - if there is none, will install one (with a fresh a OCaml compiler)
- `opam coq list` will:
  - show the available installs of Coq (similiar to `opam switch`)

Prototype

New commands for opam.

- `opam coq install coq.8.4.2` will:
  - create a new opam switch named `myocamlversion--coq.8.4.2`
  - install Coq 8.4pl2 in this switch (with its dependency on camlp5)
- `opam coq switch coq.HoTT` will:
  - browse the existing switches to find one with Coq HoTT;
  - if there is exactly one, will 'switch' to it;
  - if there is more than one, will tell the user to select one;
  - if there is none, will install one (with a fresh a OCaml compiler)
- `opam coq list` will:
  - show the available installs of Coq (similiar to `opam switch`)

Prototype

New commands for opam.

- `opam coq install coq.8.4.2` will:
  - create a new opam switch named `myocamlversion--coq.8.4.2`
  - install Coq 8.4pl2 in this switch (with its dependency on camlp5)
- `opam coq switch coq.HoTT` will:
  - browse the existing switches to find one with Coq HoTT;
  - if there is exactly one, will 'switch' to it;
  - if there is more than one, will tell the user to select one;
  - if there is none, will install one (with a fresh a OCaml compiler)
- `opam coq list` will:
  - show the available installs of Coq (similiar to `opam switch`)

Prototype

New commands for opam.

- `opam coq install coq.8.4.2` will:
  - create a new opam switch named `myocamlversion--coq.8.4.2`
  - install Coq 8.4pl2 in this switch (with its dependency on camlp5)
- `opam coq switch coq.HoTT` will:
  - browse the existing switches to find one with Coq HoTT;
  - if there is exactly one, will 'switch' to it;
  - if there is more than one, will tell the user to select one;
  - if there is none, will install one (with a fresh a OCaml compiler)
- `opam coq list` will:
  - show the available installs of Coq (similiar to `opam switch`)

One word

- Coq is a regular opam package
- Coq packages are regular opam packages
- The front-end uses regular opam switches
- The front-end uses regular opam 'pins' (to pin the installed version of Coq)

Coq 8.4pl2

```
archive: "http://coq.inria.fr/distrib/V8.4pl2/files/coq-8.4pl2.tar.gz"
```

```
opam-version: "1.1"
patches: ["configure.patch"]
build: [
  [ "./configure"
    "-configdir" "%{lib}%/coq/config"
    "-mandir" "%{man}%"
    "-docdir" "%{doc}%"
    "--prefix" "%{prefix}%"
    "--usecamlp5"
    "--coqide" "no"
#   "-opt"
  ]
  [ "make" "-j4" "world"]
  [ "make" "install"]
]
depends: ["camlp5"]
```

- camlp5 by default
- opt compilers?
- set of patch

Coq 8.4pl2+mtac

```
opam-version: "1.1"
patches: ["configure.patch" "mtac-1.1.patch"]
build: [
  [ "./configure"
    "-configdir" "%{lib}%/coq/config"
    "-mandir" "%{man}%"
    "-docdir" "%{doc}%"
    "--prefix" "%{prefix}%"
    "--usecamlp5"
    "--coqide" "no"
#   "-opt"
  ]
  [ "make" "-j4" "world"]
  [ "make" "install"]
]
depends: ["camlp5"]
```

- One line of difference!

ssreflect

```
archive:"http://ssr.msr-inria.inria.fr/FTP/ssreflect-1.5rc1.tar.gz"
checksum: "c08130242ea2cfd1cb4ae8754fa411fe"
```

```
opam-version: "1.1"
maintainer: "thomas.braibant@gmail.com"
build: [
  [make]
  [make "install"]
]
depends: [ "coq" { >= "8.4.2"}]
```

- Version constraints

In practice, in Coq, all package constraints should use 'equals'!

ssreflect

```
archive:"http://ssr.msr-inria.inria.fr/FTP/ssreflect-1.5rc1.tar.gz"
checksum: "c08130242ea2cfd1cb4ae8754fa411fe"
```

```
opam-version: "1.1"
maintainer: "thomas.braibant@gmail.com"
build: [
  [make]
  [make "install"]
]
depends: [ "coq" { >= "8.4.2"}]
```

- Version constraints

In practice, in Coq, all package constraints should use 'equals'!

Current state

HoTT.stable
aac_tactics.0.4
compcert.2.0+ia32-linux
containers.2010
coq.8.4.2+mtac
coq.8.4.2
coq.8.4.3dev
coq.8.5dev
coq.HoTT+stable
counting.2010
cybele.1.2dev
ergo.2010
flocq.2.2.0
interval.0.16.1
mathcomp.1.5
ssreflect.1.5

Various versions of the same package.

Questions

- Examples of unfinished packages
  - Ergo (from contribs) does not compile out of the box!
  - CoLoR has no install target (easy)
- Configuration options? (one of a kind problem for CompCert)
  - One package per compcert set of options?
- Semantic versioning? (`semver.org`)

  Given a version number MAJOR.MINOR.PATCH+tag, increment the:

  1. *MAJOR version when you make incompatible API changes;*
  2. *MINOR version when you add functionality in a backwards-compatible manner;*
  3. *PATCH version when you make backwards-compatible bug fixes.*

- How to deal with git branches as releases? (HoTT)
- camlp5/camlp4?
- coq_makefile, uninstall targets?
- Merge `user-contribs`, `theories`?
- Good uses of `-R`, `-I`?

Questions

- Examples of unfinished packages
  - Ergo (from contribs) does not compile out of the box!
  - CoLoR has no install target (easy)
- Configuration options? (one of a kind problem for CompCert)
  - One package per compcert set of options?
- Semantic versioning? (`semver.org`)

  Given a version number MAJOR.MINOR.PATCH+tag, increment the:

  1. *MAJOR version when you make incompatible API changes;*
  2. *MINOR version when you add functionality in a backwards-compatible manner;*
  3. *PATCH version when you make backwards-compatible bug fixes.*

- How to deal with git branches as releases? (HoTT)
- camlp5/camlp4?
- coq_makefile, uninstall targets?
- Merge `user-contribs`, `theories`?
- Good uses of `-R`, `-I`?

Questions

- Examples of unfinished packages
  - Ergo (from contribs) does not compile out of the box!
  - CoLoR has no install target (easy)
- Configuration options? (one of a kind problem for CompCert)
  - One package per compcert set of options?
- Semantic versioning? (`semver.org`)

  Given a version number MAJOR.MINOR.PATCH+tag, increment the:

  1. *MAJOR version when you make incompatible API changes;*
  2. *MINOR version when you add functionality in a backwards-compatible manner;*
  3. *PATCH version when you make backwards-compatible bug fixes.*

- How to deal with git branches as releases? (HoTT)
- camlp5/camlp4?
- coq_makefile, uninstall targets?
- Merge `user-contribs`, `theories`?
- Good uses of `-R`, `-I`?

Questions

- Examples of unfinished packages
  - Ergo (from contribs) does not compile out of the box!
  - CoLoR has no install target (easy)
- Configuration options? (one of a kind problem for CompCert)
  - One package per compcert set of options?
- Semantic versioning? (`semver.org`)

  Given a version number MAJOR.MINOR.PATCH+tag, increment the:

  1. *MAJOR version when you make incompatible API changes;*
  2. *MINOR version when you add functionality in a backwards-compatible manner;*
  3. *PATCH version when you make backwards-compatible bug fixes.*

- How to deal with git branches as releases? (HoTT)
- camlp5/camlp4?
- coq_makefile, uninstall targets?
- Merge `user-contribs`, `theories`?
- Good uses of `-R`, `-I`?

Perspectives

- Fine grained packages
- Stripped down Coq (libraries and plugins as separate packages)
- Download statistics (how much something is used)
- Opam2Web (to list packages)
- Ocamlot (continuous integration/test infrastructure)
- SDK for Windows, OS X, GNU/Linux (Vagrant?)
- Use libraries in Coq (pprint, framaC/unmarshal, ancient, persistent arrays, ...)
- Use cutting-edge OCaml (gadts, ephemerons, ...)
- Multiple repos of Coq packages

URLs

- https://github.com/braibant/opam
- https://github.com/braibant/opam-coq-repo

Pull-requests welcome

Move it to coq.inria.fr?

Plan D

Package Coq specific commands as an opam package

On my todo-list

TL; DR

OPAM
ONLY
DOES
EVERYTHING

thanks OCamlPro

One more thing

Survey about Coq

- Know the users (researchers, teachers, students, numbers)
- Know what they do (math, pl, production)
- Know how they work (ide, how much time, tactics)
- Know what library/plugin/contrib they use (are contribs used)
- Know what they find important (doc, perf, backward compatibility, new features)
- Guide development