Making cabal-install non-destructive

Philipp Schuster, Andres Löh

September 12, 2012
Introduction

- My name is Philipp Schuster.
- I participated in Google Summer of Code 2012.
- My supervisor was Andres Löh.
- We wanted multiple instances of the same package version installed.
- Quite a few problems remain therefore nothing is merged yet.
Example Packages

King-1

Knight

Knight-1

Pawn

Pawn-1
Example Instances

- King-1-b22...
- Knight-1-f7a...
- Pawn-1-e34...
Listing the installed instances

$ ghc-pkg list --user -v
using cache: /home/pschuster/.ghc/i386-linux-7.6.0.20120815/package.conf.d/package.cache
using cache: /usr/local/lib/ghc-7.6.0.20120815/package.conf.d/package.cache
/home/pschuster/.ghc/i386-linux-7.6.0.20120815/package.conf.d
   King-1 (King-1-165729ba77dabdb7b827de2e721291b61-1020960593)
   Knight-1 (Knight-1-d1e1f57c04f2a3f462e6e364c4d6e-1040356745)
   Pawn-1 (Pawn-1-7a9672f4fce029cc4d72cc5957d45134-1022359486)
Queen-1 and Pawn-2 are added
Instances with Pawn-2 installed

- King-1-b22...
  - Knight-1-f7a...
  - Pawn-1-e34...
- Pawn-2-f89...
$ cd Pawn
$ cabal install
Resolving dependencies...
Configuring Pawn-2...
Building Pawn-2...
Preprocessing library Pawn-2...
[1 of 1] Compiling Pawn ( Pawn.hs, dist/build/Pawn.o )
In-place registering Pawn-2...
Installing library in /home/pschuster/.cabal/lib/Pawn-2-1181001620
Registering Pawn-2...
Installed Pawn-2
Instances with Queen installed

- King-1-b22...
- Knight-1-f7a...
- Pawn-1-e34...
- Queen-1-bc7...
- Knight-1-228...
- Pawn-2-f89...
There used to be a conflict
Trying to install another Knight

$ cd ../Knight
$ cabal install
Resolving dependencies...
In order, the following would be installed:
Knight-1 (reinstall) changes: Pawn-1 -> 2
cabal: The following packages are likely to be broken by the reinstalls:
King-1
Use --force-reinstalls if you want to install anyway.
Forcing to install another Knight

$ cabal install --force-reinstalls
Resolving dependencies...
Warning: The following packages are likely to be broken by the reinstalls:
King-1
Continuing even though the plan contains dangerous reinstalls.
Configuring Knight-1...
Building Knight-1...
Preprocessing library Knight-1...
[1 of 1] Compiling Knight ( Knight.hs, dist/build/Knight.o ) [Pawn changed]
In-place registering Knight-1...
Installing library in /home/pschuster/.cabal/lib/Knight-1-1213798927
Registering Knight-1...
Installed Knight-1
Knight got installed in a different location

$ ghc-pkg field Knight id,library-dirs
id: Knight-1-2a238a015dfde8866586869fc773edcf-1213798927
library-dirs: /home/pschuster/.cabal/lib/Knight-1-1213798927
id: Knight-1-d1e1f57c04f2a3f462e6c2ee364c4dfe-1040356745
library-dirs: /home/pschuster/.cabal/lib/Knight-1-1040356745
Instances with Queen installed

- King-1-b22...
- Knight-1-f7a...
- Pawn-1-e34...
- Queen-1-bc7...
- Knight-1-228...
- Pawn-2-f89...
Both instances of Knight are there

$ ghc-pkg field Knight id,depends
id: Knight-1-2a238a015dfde8866586869f7c773edcf-1213798927
depends: base-4.6.0.0-188a8a5ba06e0bf0503ba32ec2568ac7
  Pawn-2-824eda7296a96dd8a5eb9c8cbf3e2f24-1181001620
id: Knight-1-d1e1f57c04f2a3f462eecc2e364c4dbe-1040356745
depends: base-4.6.0.0-188a8a5ba06e0bf0503ba32ec2568ac7
  Pawn-1-7a9672f4fce029cc4d72cc5957d45134-1022359486
Installing another King

$ cd ../King
$ cabal install
Resolving dependencies...
In order, the following will be installed:
King-1 (reinstall)
Warning: Note that reinstalls are always dangerous. Continuing anyway...
Configuring King-1...
Building King-1...
Preprocessing library King-1...
[1 of 1] Compiling King ( King.hs, dist/build/King.o ) [Knight changed]
In-place registering King-1...
Installing library in /home/pschuster/.cabal/lib/King-1-1113590318
Registering King-1...
Installed King-1
King depends on the new Knight instance

$ ghc-pkg field King id,depends
id: King-1-3ec40c2c9564c1fd109479a358a82eef-1113590318
depends: base-4.6.0.0-188a8a5ba06e0bf0503ba32ec2568ac7
    Knight-1-2a238a015dfde8866586869fc773edcf-1213798927
id: King-1-165729ba77dabd7b827de2e2721291b61-1020960593
depends: base-4.6.0.0-188a8a5ba06e0bf0503ba32ec2568ac7
    Knight-1-d1e1f57c04f2a3f462eec2ee364c4dbe-1040356745
Instances with another King installed

- King-1-b22...
- Knight-1-f7a...
- Pawn-1-e34...

- King-1-c5d...
- Knight-1-228...

- Queen-1-bc7...
- Pawn-2-f89...
Calling the garbage collector

$ cabal remove --duplicates
"Would remove King-1-165729ba77dab7b827de2e721291b61-1020960593"
"Would remove Knight-1-d1e1f57c04f2a3f462e6c24664c4dbe-1040356745"
Instances that would be garbage collected

- King-1-b22...
- Knight-1-f7a...
- Pawn-1-e34...
- King-1-c5d...
- Knight-1-228...
- Queen-1-bc7...
- Pawn-2-f89...
Install location

- Customizable in `.cabal/config`. 
Install location

- Customizable in `.cabal/config`.
- Default `$libsubdir` was `$pkgid/$compiler` for example `repa-3.1.4.2/ghc-7.4.1`. Default should be `$pkgid-$unique` for example `repa-3.1.4.2-1079787003`. $unique is resolved to a big random number but only by `cabal-install` not by Cabal the library.
Install location

- Customizable in .cabal/config.
- Default $libsubdir was $pkgid/$compiler for example repa-3.1.4.2/ghc-7.4.1.
- Default should be $pkgid-$unique for example repa-3.1.4.2-1079787003.
Install location

- Customizable in .cabal/config.
- Default $libsubdir was $pkgid/$compiler for example repa-3.1.4.2/ghc-7.4.1.
- Default should be $pkgid-$unique for example repa-3.1.4.2-1079787003.
- $unique is resolved to a big random number but only by cabal-install not by Cabal the library.
Defaults for cabal-install and Cabal the library would be different.
Defaults for cabal-install and Cabal the library would be different.

Because of package_Paths.hs the install location has to be known at compile time.
InstalledPackageld

- Was Packageld-ABIhash for example
  base-4.6.0.0-188a8a5ba06e0bf0503ba32ec2568ac7.
InstalledPackageId

- Was PackageId-ABIhash for example base-4.6.0.0-188a8a5ba06e0bf0503ba32ec2568ac7.
- Is PackageId-ABIhash-BigRandom for example accelerate-0.12.1.0-c655a93ff75289c7bc2703bfd115c0a3-1248341437.
InstalledPackageId

- Was PackageId-ABIhash for example base-4.6.0.0-188a8a5ba06e0bf0503ba32ec2568ac7.
- Is PackageId-ABIhash-BigRandom for example accelerate-0.12.1.0-c655a93ff75289c7bc2703bfd115c0a3-1248341437.
- cabal-install determines the random number during configuration.
InstalledPackageId

- Was PackageId-ABIhash for example base-4.6.0.0-188a8a5ba06e0bf0503ba32ec2568ac7.
- Is PackageId-ABIhash-BigRandom for example accelerate-0.12.1.0-c655a93ff75289c7bc2703bfd115c0a3-1248341437.
- cabal-install determines the random number during configuration.
- Cabal the library only appends the given String.
InstalledPackageId

- Was PackageId-ABIhash for example `base-4.6.0.0-188a8a5ba06e0bf0503ba32ec2568ac7`.
- Is PackageId-ABIhash-BigRandom for example `accelerate-0.12.1.0-c655a93ff75289c7bc2703bfd115c0a3-1248341437`.
- `cabal-install` determines the random number during configuration.
- `Cabal` the library only appends the given String.
- `InstalledPackageId` can not be used as the install location because it contains the ABI hash.
Time-stamp

- A field time-stamp was added to InstalledPackageInfo.
Time-stamp

- A field time-stamp was added to InstalledPackageInfo.
- Used by cabal-install, Cabal and GHC to choose between instances.
Time-stamp

- A field time-stamp was added to InstalledPackageInfo.
- Used by cabal-install, Cabal and GHC to choose between instances.
- Not sure if shadowing in GHC still works.
ghc-pkg does not overwrite anymore

- When a new package is registered ghc-pkg used to remove all other instances with the same version.
ghc-pkg does not overwrite anymore

- When a new package is registered ghc-pkg used to remove all other instances with the same version.
- Now ghc-pkg never removes anything when registering.
ghc-pkg does not overwrite anymore

- When a new package is registered ghc-pkg used to remove all other instances with the same version.
- Now ghc-pkg never removes anything when registering.
- It should probably warn when inserting a package with an existing InstalledPackageld.
More of a proof of concept.
cabal remove --duplicates

- More of a proof of concept.
- Suggests all unnecessary packages for removal.
More of a proof of concept.

Suggests all unnecessary packages for removal.

A package is unnecessary if all packages that depend on it are unnecessary.

cabal remove –duplicates
More of a proof of concept.

Suggests all unnecessary packages for removal.

A package is unnecessary if all packages that depend on it are unnecessary

and it is not the latest instance of its version.
More of a proof of concept.

Suggests all unnecessary packages for removal.

A package is unnecessary if all packages that depend on it are unnecessary

and it is not the latest instance of its version.

It does not even unregister.
Why not hash the build inputs?

- The original idea was to hash all build inputs (compiler, tools, source, dependencies).
Why not hash the build inputs?

- The original idea was to hash all build inputs (compiler, tools, source, dependencies).
- Use this "cabal-hash" to identify an instance and to detect if an instance can be reused.
Why not hash the build inputs?

- The original idea was to hash all build inputs (compiler, tools, source, dependencies).
- Use this "cabal-hash" to identify an instance and to detect if an instance can be reused.
- Conflating all build information into a hash has a drawback:
Why not hash the build inputs?

- The original idea was to hash all build inputs (compiler, tools, source, dependencies).
- Use this "cabal-hash" to identify an instance and to detect if an instance can be reused.
- Conflating all build information into a hash has a drawback:
- Two packages might be usable together although their build inputs and therefore their hashes are not exactly the same.
Comparing hashes is an optimization

- Let’s consider two theoretically possible modes for dependency resolution in cabal-install:
Comparing hashes is an optimization

- Let’s consider two theoretically possible modes for dependency resolution in cabal-install:
  - Mode 1: Disregard all installed packages, come up with an install plan and if some of the necessary packages are already there use them.
Comparing hashes is an optimization

- Let’s consider two theoretically possible modes for dependency resolution in cabal-install:
  - Mode 1: Disregard all installed packages, come up with an install plan and if some of the necessary packages are already there use them.
  - Mode 2: Take into account the installed packages and try to prefer them when making the install plan.
Comparing hashes is an optimization

- Let’s consider two theoretically possible modes for dependency resolution in cabal-install:
  - Mode 1: Disregard all installed packages, come up with an install plan and if some of the necessary packages are already there use them.
  - Mode 2: Take into account the installed packages and try to prefer them when making the install plan.
- Using a hash makes Mode 2 impossible unless all the information is also available from InstalledPackageInfo.
Comparing hashes is an optimization

- Let’s consider two theoretically possible modes for dependency resolution in cabal-install:

  - Mode 1: Disregard all installed packages, come up with an install plan and if some of the necessary packages are already there use them.

  - Mode 2: Take into account the installed packages and try to prefer them when making the install plan.

- Using a hash makes Mode 2 impossible unless all the information is also available from InstalledPackageInfo.

- Using a hash is an optimization.
Compilation is not deterministic

- Just a "cabal-hash" is not enough for unique identification.
Compilation is not deterministic

- Just a "cabal-hash" is not enough for unique identification.
- Even compiling with the same build inputs is not guaranteed to yield the same instance.
Compilation is not deterministic

- Just a "cabal-hash" is not enough for unique identification.
- Even compiling with the same build inputs is not guaranteed to yield the same instance.
- Would not be a problem if there would only ever be one instance per build inputs per machine.
Compilation is not deterministic

- Just a "cabal-hash" is not enough for unique identification.
- Even compiling with the same build inputs is not guaranteed to yield the same instance.
- Would not be a problem if there would only ever be one instance per build inputs per machine.
- But we have a global and a user database so there might actually be two incompatible instances with the same build inputs.
Communicate the InstalledPackageId back to cabal-install

- cabal-install comes up with an InstallPlan containing to be installed packages.
Communicate the InstalledPackageId back to cabal-install

- cabal-install comes up with an InstallPlan containing to be installed packages.
- Those depend upon each other as well as on already installed packages.
Communicate the InstalledPackageId back to cabal-install

- cabal-install comes up with an InstallPlan containing to be installed packages.
- Those depend upon each other as well as on already installed packages.
- We want to specify all of those dependencies with an InstalledPackageId.
Communicate the InstalledPackageId back to cabal-install

- cabal-install comes up with an InstallPlan containing to be installed packages.
- Those depend upon each other as well as on already installed packages.
- We want to specify all of those dependencies with an InstalledPackageId.
- The InstalledPackageId is only known after installation.
Communicate the InstalledPackageId back to cabal-install

- cabal-install comes up with an InstallPlan containing to be installed packages.
- Those depend upon each other as well as on already installed packages.
- We want to specify all of those dependencies with an InstalledPackageId.
- The InstalledPackageId is only known after installation.
- It has to be communicated back to cabal-install.
Communicate the InstalledPackageId back to cabal-install

- cabal-install comes up with an InstallPlan containing to be installed packages.
- Those depend upon each other as well as on already installed packages.
- We want to specify all of those dependencies with an InstalledPackageId.
- The InstalledPackageId is only known after installation.
- It has to be communicated back to cabal-install.
- The current workaround is to only specify those instances that were already installed with an InstalledPackageId.
Future work

- More fine grained build inputs.
Future work

- More fine grained build inputs.
- Garbage collection that does something.
Future work

- More fine grained build inputs.
- Garbage collection that does something.
- Andres still wants a cabal hash.
Questions/Discussion