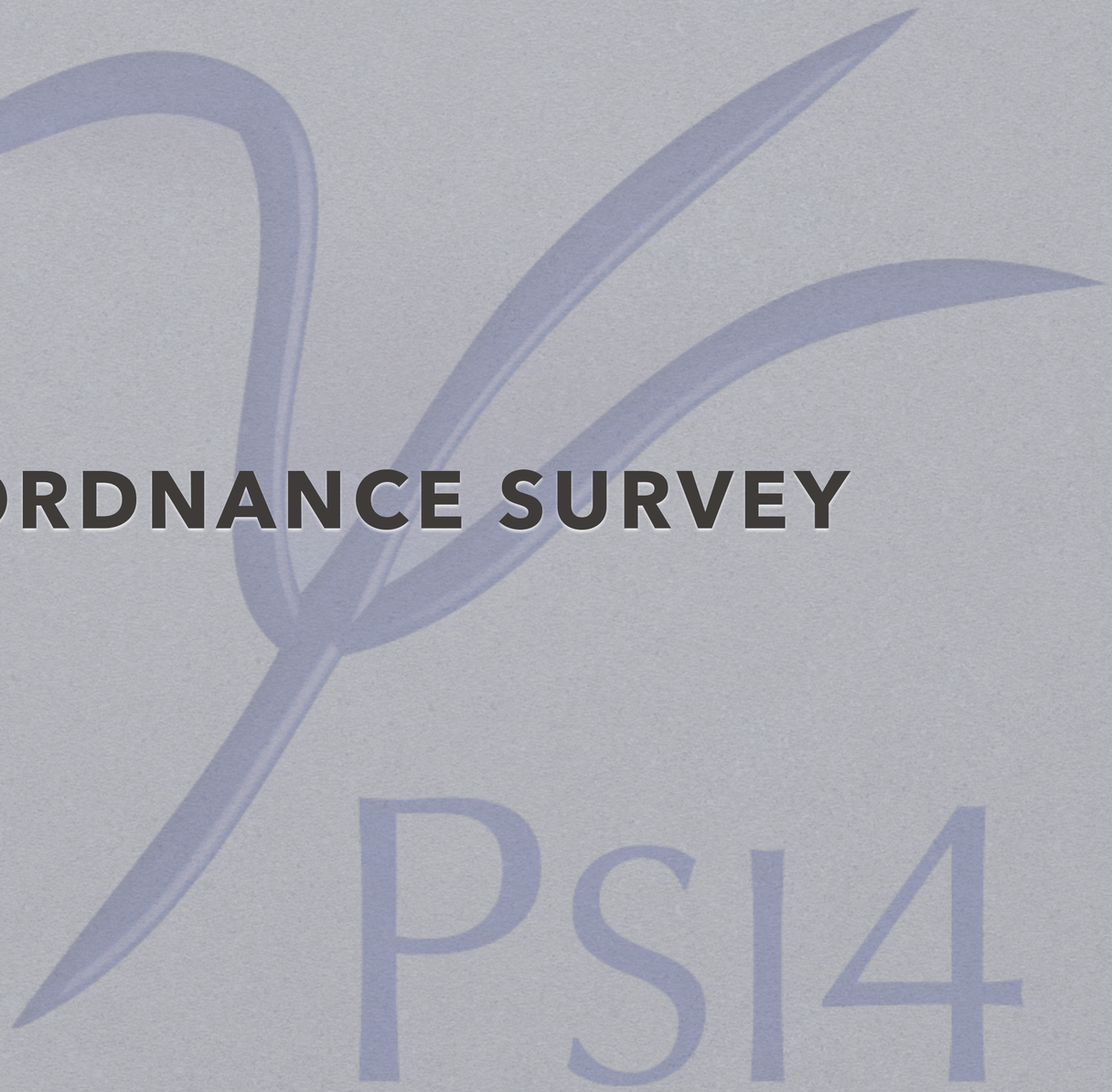
2017 Psi4 ORDNANCE SURVEY

LORI A. BURNS

Psi WWDC
3 NOV 2017



INPUT MODES

Psithon

```
molecule dimer {
O
H 1 1.0
H 1 1.0 2 90.0
--
H 1 R 2 135.0 3 0.0
O 4 1.0 1 135.0 3 0.0
H 5 1.0 4 90.0 90.0

R = 3.0
}

set basis aug-cc-pvdz

energy("sapt2+dmp2")
```

PsiAPI

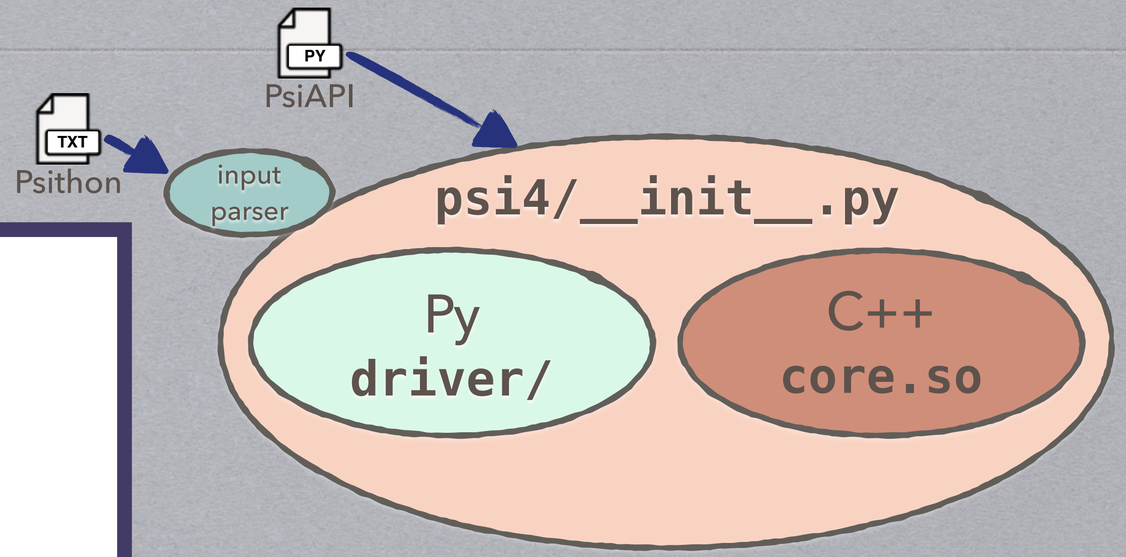
```
import psi4

psi4.geometry("""
O
H 1 1.0
H 1 1.0 2 90.0
--
H 1 R 2 135.0 3 0.0
O 4 1.0 1 135.0 3 0.0
H 5 1.0 4 90.0 90.0

R = 3.0
""")

psi4.set_options({
"basis": "aug-cc-pvdz"})

psi4.energy("sapt2+dmp2")
```

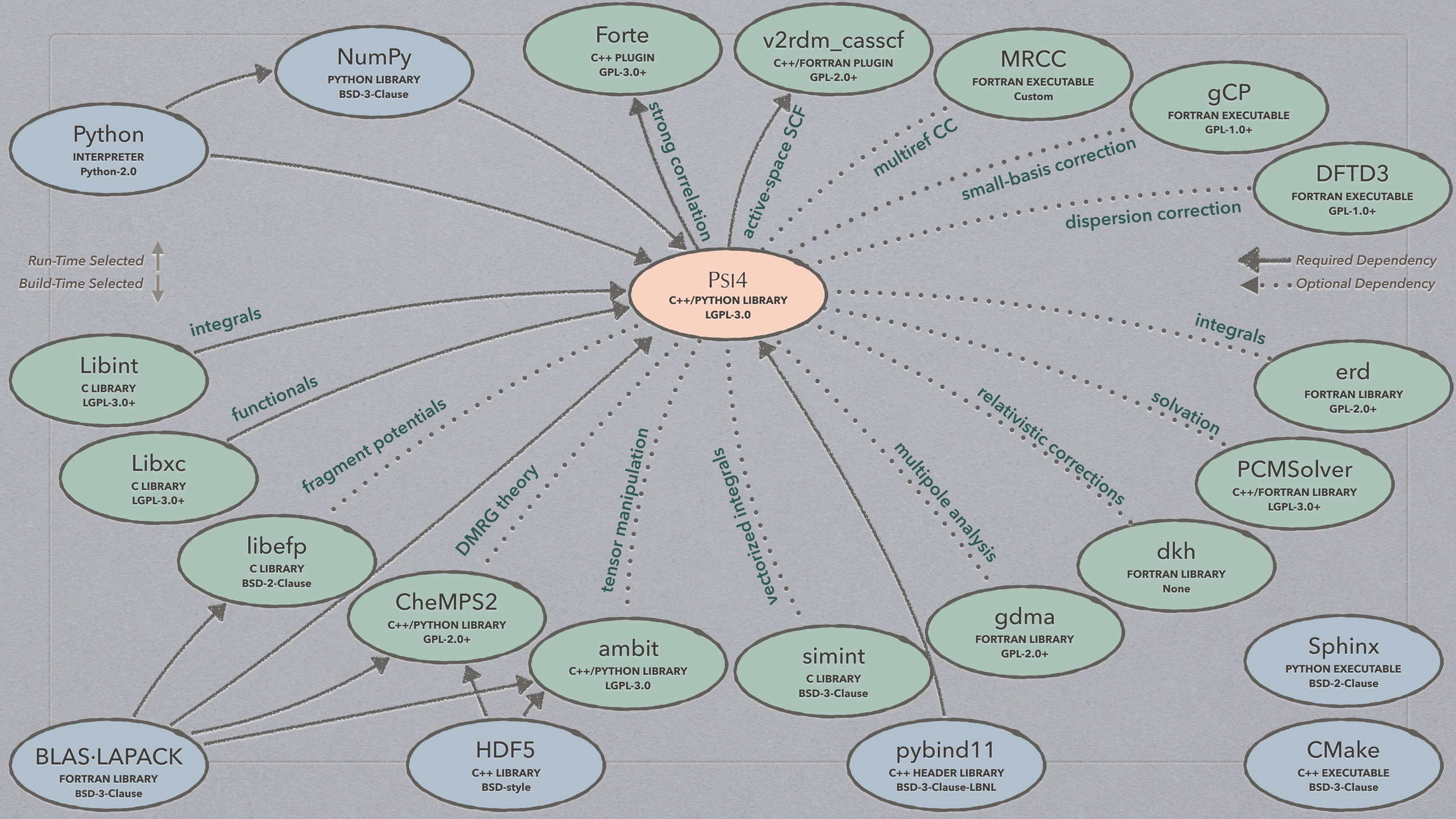


PYTHON MODULE W/C++ EXTENSION

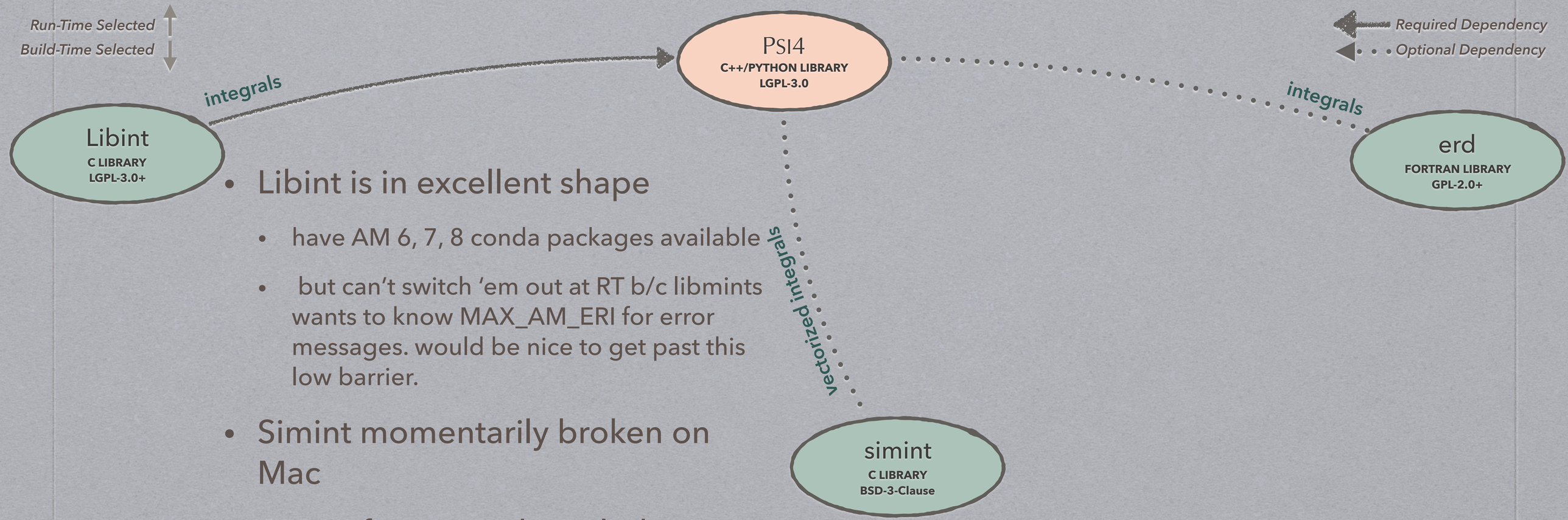
PSI4 c. 2017, 1.1

- Psithon (normal)
  - been around forever
  - would deprecate the term as domain-specific languages not come il faut but it's useful to name contrast to psiapi
  - sloppier namespacing
  - set PATH, run with **`psi4 input.in`**
- PsiAPI (new)
  - new mode, just as powerful
  - don't care what we call it, but a name is handy
  - set PYTHONPATH, run with **`python input.py`**

```
>>> psi4 --psiapi-path
export PATH=/Users/username/miniconda/envs/p4dev/bin:$PATH
export PYTHONPATH=/Users/username/gits/objdir/stage/usr/local/psi4/lib:$PYTHONPATH
>>> python -c "import psi4; print(psi4.__version__)"
```

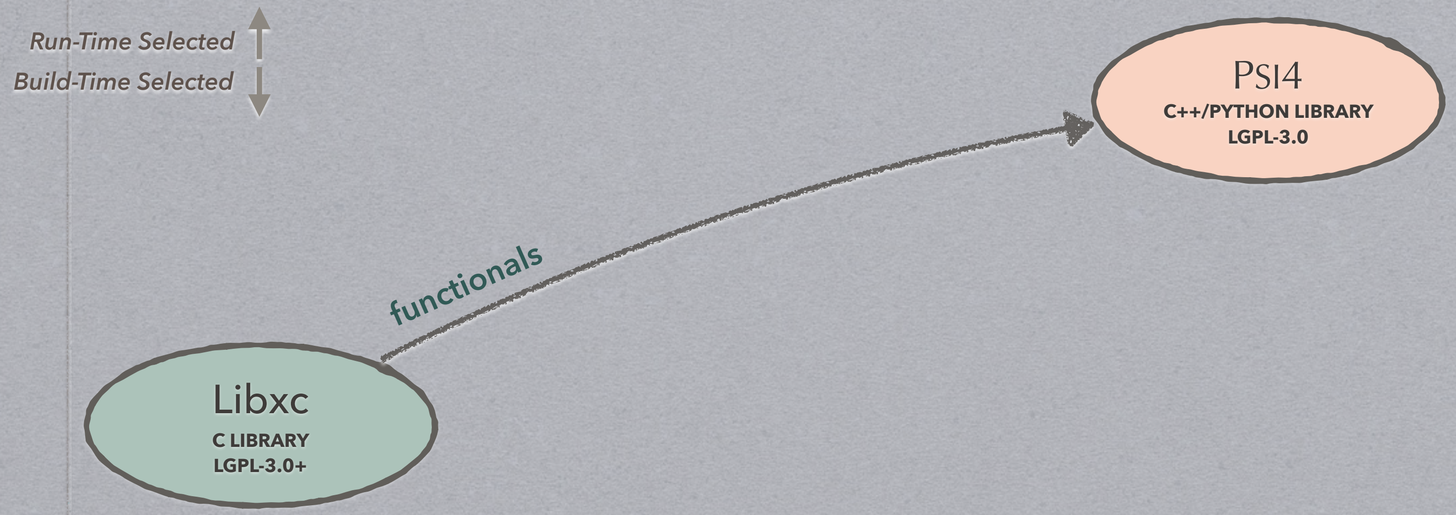

Forte
C++ PLUGIN
GPL-3.0+
v2rdm_casscf
C++/FORTRAN PLUGIN
GPL-2.0+
MRCC
FORTRAN EXECUTABLE
Custom
gCP
FORTRAN EXECUTABLE
GPL-1.0+
DFTD3
FORTRAN EXECUTABLE
GPL-1.0+
NumPy
PYTHON LIBRARY
BSD-3-Clause
Python
INTERPRETER
Python-2.0
strong correlation
active-space SCF
multiref CC
small-basis correction
dispersion correction
Run-Time Selected
Build-Time Selected
Required Dependency
Optional Dependency
PSI4
C++/PYTHON LIBRARY
LGPL-3.0
integrals
Libint
C LIBRARY
LGPL-3.0+
functionals
Libxc
C LIBRARY
LGPL-3.0+
fragment potentials
libefp
C LIBRARY
BSD-2-Clause
DMRG theory
CheMPS2
C++/PYTHON LIBRARY
GPL-2.0+
tensor manipulation
ambit
C++/PYTHON LIBRARY
LGPL-3.0
vectorized integrals
simint
C LIBRARY
BSD-3-Clause
multipole analysis
gdma
FORTRAN LIBRARY
GPL-2.0+
relativistic corrections
dkh
FORTRAN LIBRARY
None
integrals
erd
FORTRAN LIBRARY
GPL-2.0+
solvation
PCMSolver
C++/FORTRAN LIBRARY
LGPL-3.0+
Sphinx
PYTHON EXECUTABLE
BSD-2-Clause
BLAS·LAPACK
FORTRAN LIBRARY
BSD-3-Clause
HDF5
C++ LIBRARY
BSD-style
pybind11
C++ HEADER LIBRARY
BSD-3-Clause-LBNL
CMake
C++ EXECUTABLE
BSD-3-Clause

INTEGRALS



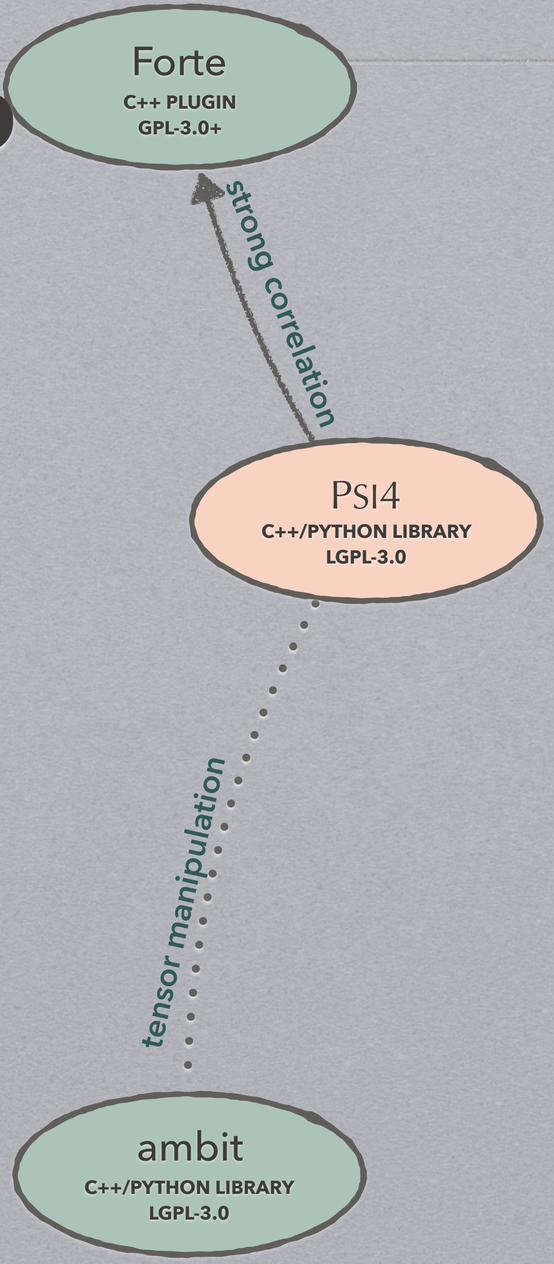
- Libint is in excellent shape
  - have AM 6, 7, 8 conda packages available
  - but can't switch 'em out at RT b/c libmints wants to know MAX_AM_ERI for error messages. would be nice to get past this low barrier.
- Simint momentarily broken on Mac
- ERD is fine to my knowledge

FUNCTIONALS



- Libxc was doing fine; we were prepared to switch to upstream when they indicated stability (e.g., RC1)
- Unfortunately, they simply declared 4.0
- New interface separates internal vs. external functional parameter setters in such a way as to greatly restrict RT-manipulation of fctls. Probably can't reproduce current Psi capabilities in 4.0.
- So currently still stuck at 3.0, pulling from GH:psi4/libxc, rather than GL:libxc/libxc.
  - joining PySCF & Horton stuck at Libxc 2.0.
  - we should communicate release dates and provide migration paths when we break API, lest others get irked at us

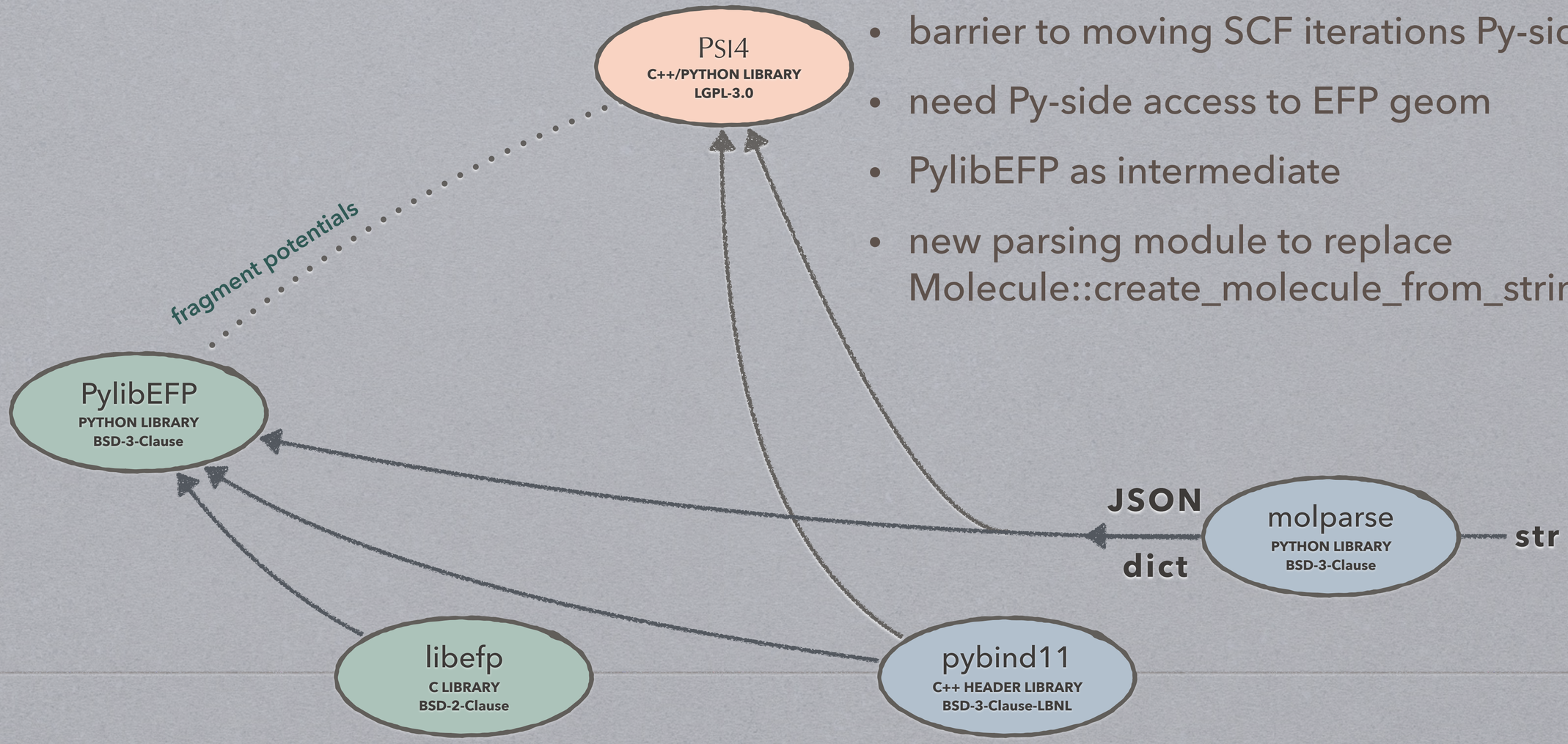
TENSOR CROWD

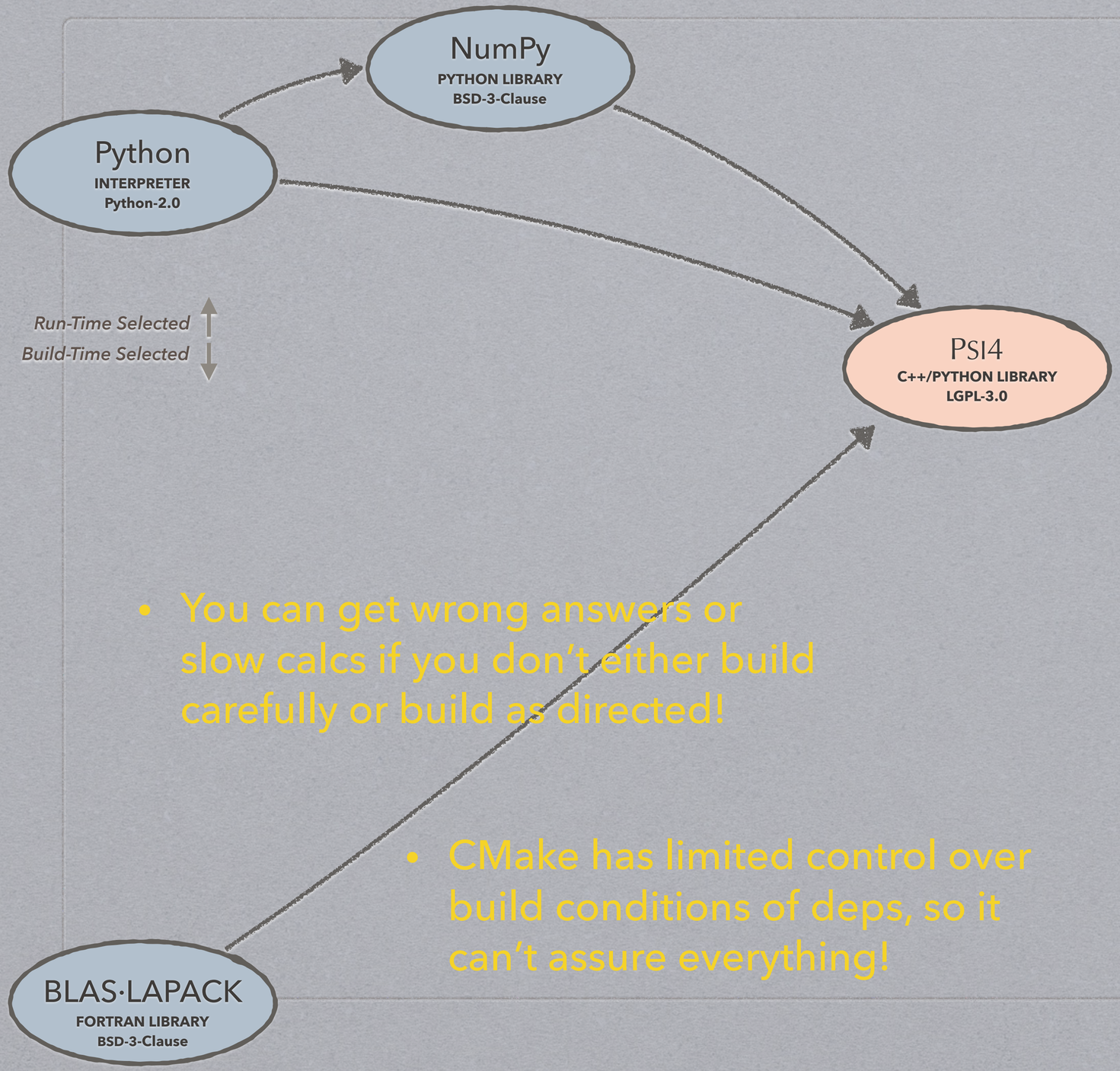


- Have ambit and Forte come to an agreement on the one true master branch?
- If we can add some internal versioning, are these ready to conda package?
- Forte, I believe, has pybound itself. More on potential complications later.
- Any other niftiness from Forte that should move back to Psi?

EFP COMPLICATIONS

- EFP presently octopus-like C-side
  - global EFP in P::e
  - parsing in libmints.Molecule
  - binary, library, if/else in SCF
- barrier to moving SCF iterations Py-side
- need Py-side access to EFP geom
- PylibEFP as intermediate
- new parsing module to replace Molecule::create_molecule_from_string





- You can get wrong answers or slow calcs if you don't either build carefully or build as directed!

- CMake has limited control over build conditions of deps, so it can't assure everything!

INSIDIOUS LAPACK

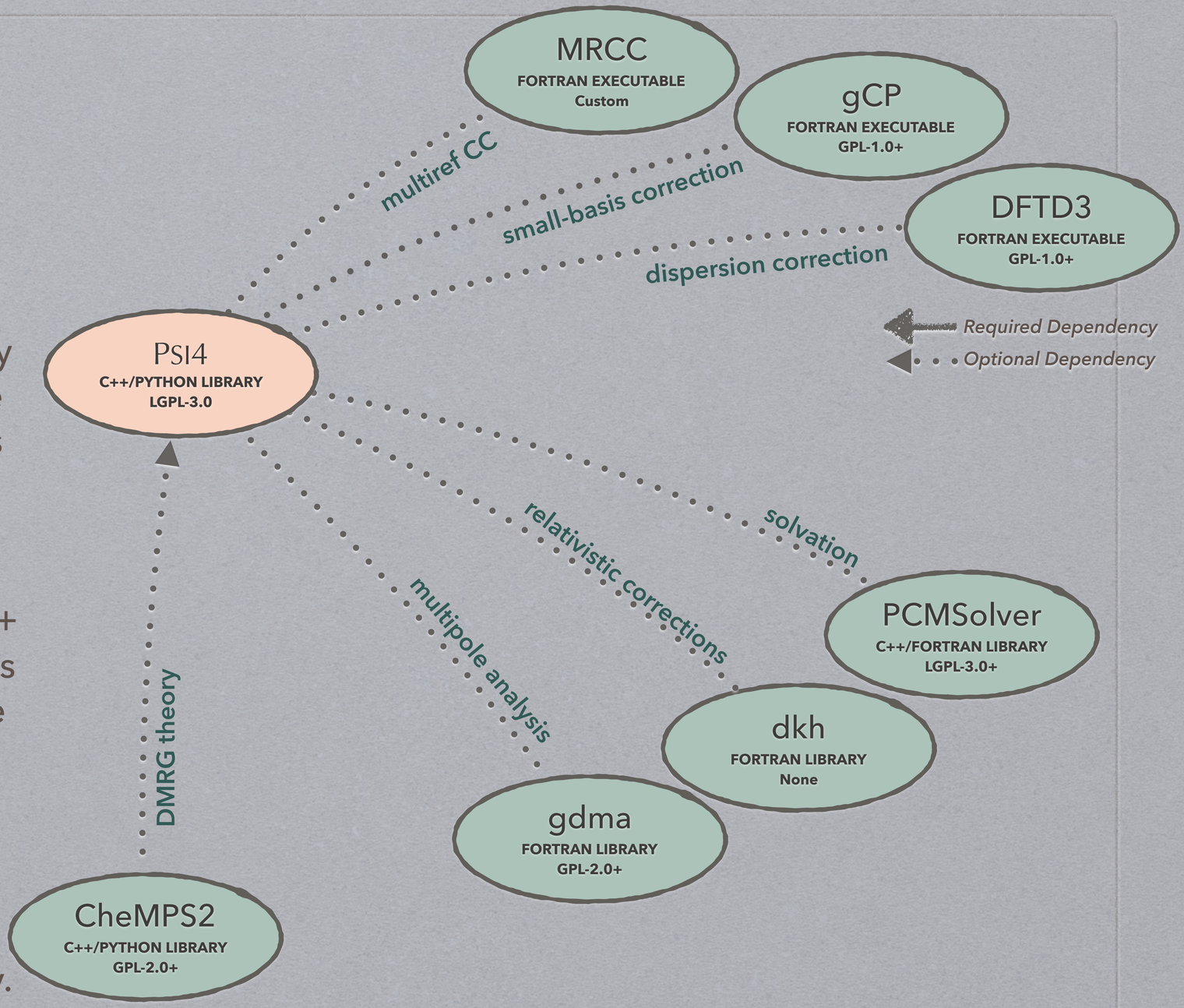
- Previously for conda packages (and you can still arrange your dev build this way), used Accelerate for Mac and statically linked MKL for Linux.
- Most ppl used NumPy from default channel linked to intel_thread/core.
- Initially noticed by RMcG, NumPy linalg errors show up with mixed LAPACK (and I mean wrong answers, not notification that env is wrong) and also threading performance can be suppressed if first lib loaded isn't multi architecture capable.
- Conda pkgs reworked to link to mkl_rt. **`bin/psi4-path-advisor`** sets this up for you, too.
- Numpy from default conda channel links to intel_thread/core, not rt, so import order starts to become important.
- Numpy from intel conda channel links to mkl_rt, so more robust wrt threading. Recommended. Now available for py27, 35, 36.
- Could avoid (but not eliminate) many user problems by importing numpy first in psithon, at cost of losing "how to get numpy" error msg.

OPENMP

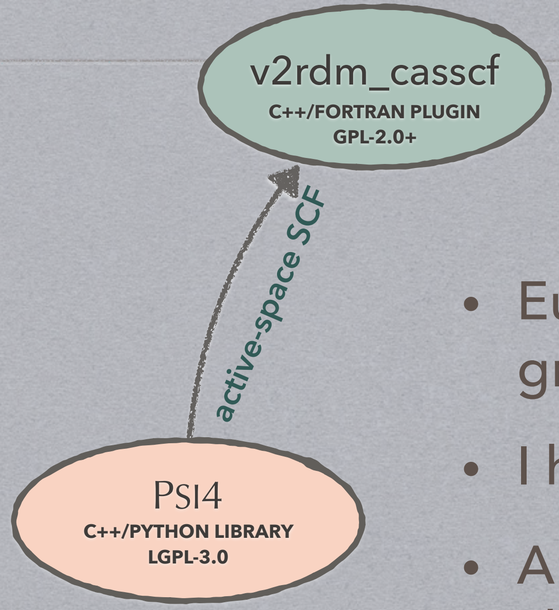
- On Mac, AppleClang has no OMP. *Can* build with GCC w/ OMP, but it's slow and painful to invest in meagerly supported software. For conda have to make separate sse41 packages for old hardware.
- On Linux, Intel compilers provide **`-msse2 -axCORE-AVX2,AVX`** flags that optimize code for multiple architectures. For conda, we presently rely on a old and custom gcc5.2 package.
- LAPACK & OMP are sadly intertwined, and current released FindOpenMP.cmake is not too helpful.
- Conda has a new compiler toolchain with very recent GCC 7 on Linux and Clang 4.0.1 on Mac (so new Intel 2018 can't handle its headers). Note that that's real Clang with OMP support (it works). All these are conda-installable, so no more Xcode Clang or system GCC.
- Once Intel Mac gets with the program, plan is to
  - drop Mac GCC and use Intel atop Clang with OMP & multiarch, thus building all-in-one pkgs like Linux
  - need legitimate Mac Intel license once path is validated
- Once I get time, Linux plan is to
  - drop current gcc5.2 and move to Intel atop GCC 7, continuing multiarch
- OpenMP detection in CMake is improving at >3.9. If it improves enough, will rework Math/OMP detection to use it and bump CMake min from 3.3 to at least 3.8

ADDONS

- CheMPS2 continues gamely, though I doubt it gets much use through Psi b/c parallelism.
- GDMA is great (except for that string length thing) but can't be updated trivially b/c bin –> lib conversion. Persuade Stone to separate into library and driver? Call as exe through Py instead?
- DKH, no complaints
- PCMSolver I suspect could get a piebald + C++ lib treatment, but other QC programs probably constraining it. Should its I/O be JSON-ified? Can we call it from Py-side when SCF iterations move thither?
- DFTD3 good for now, but the DFTD4 paper is out (haven't seen any code).
- gCP could use mods to improve flexibility.
- MRCC, there when you're desperate.

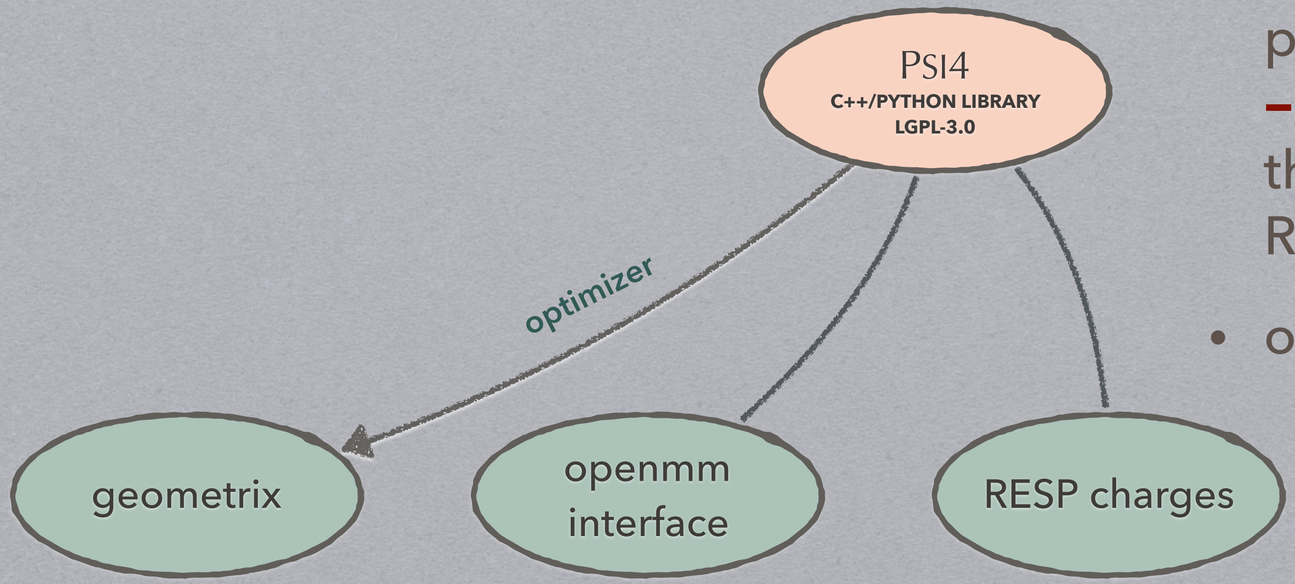


PLUGINS



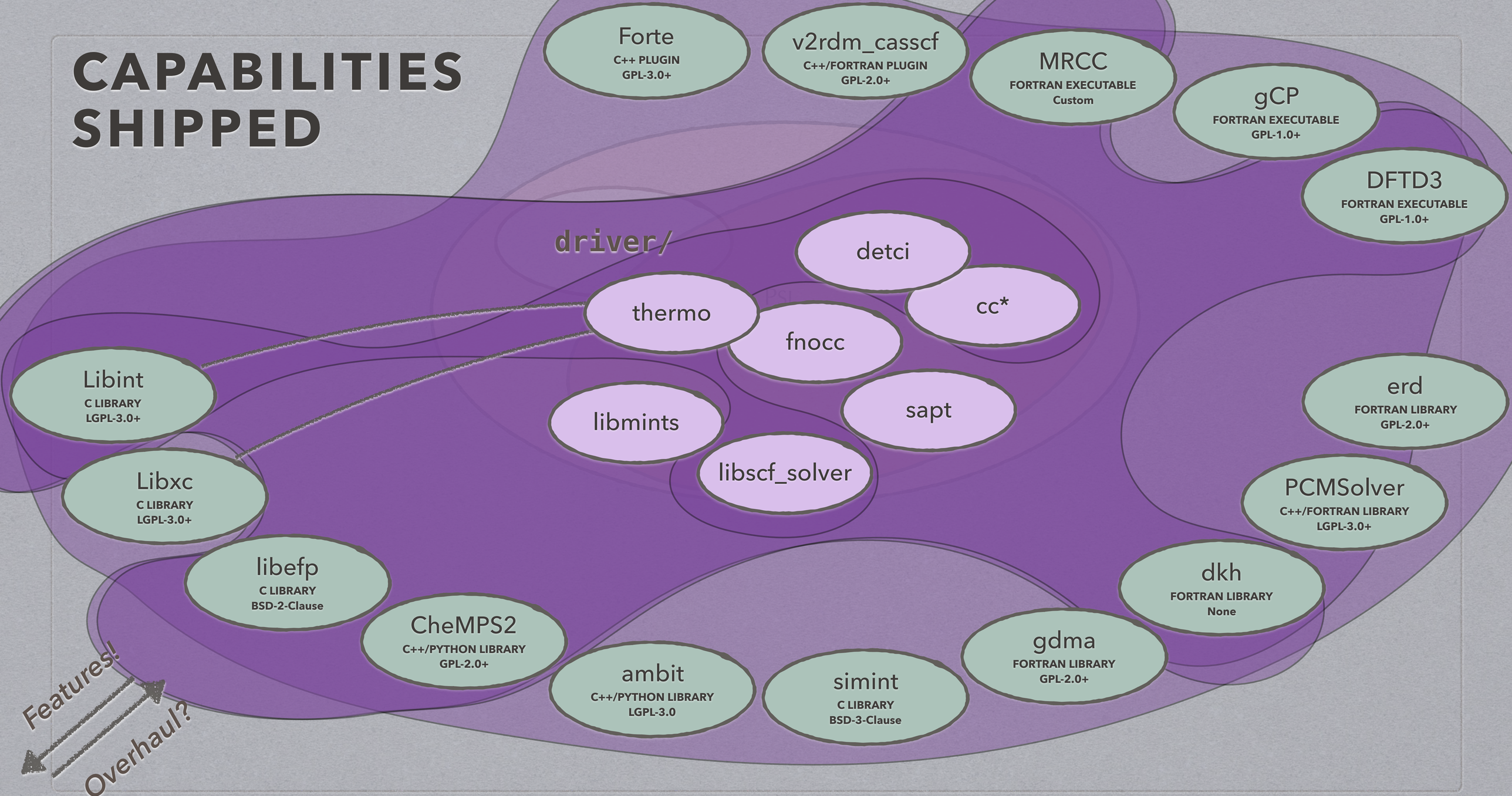
- Eugene newly updated with gradients
- I haven’t reintegrated into conda
- Among those suffering from ABI changes w/o version changes
- Pain that our plugins feel wrt ABI/API breakage is akin to what any upstream package feels

DESERVING PROJECTS

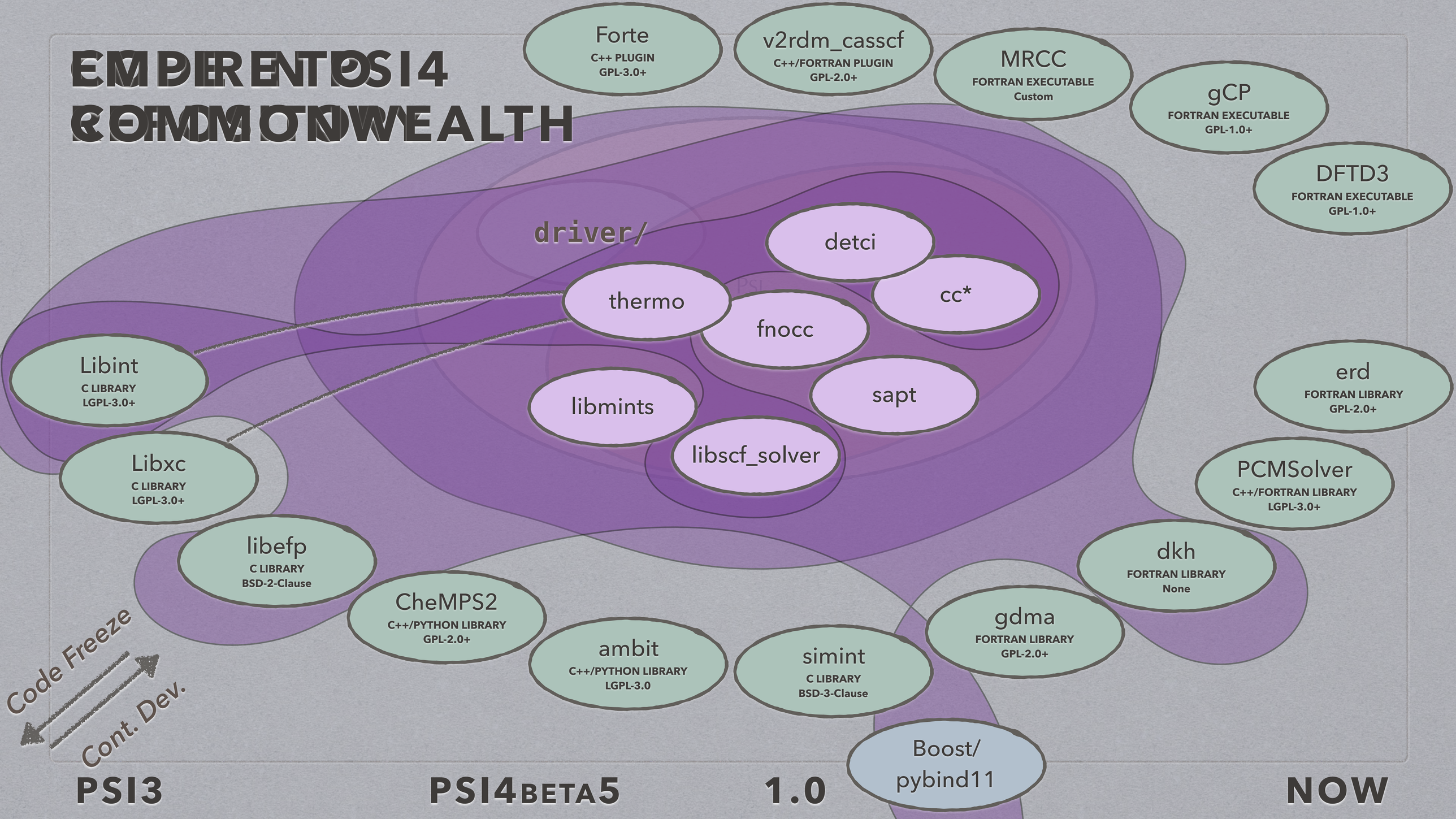


- I'd like to make even pure-Py projects build-on-demand (**`cmake -DENABLE_v2rdm_casscf`**), even though they're only interfaced at RT
- others, I'm sure

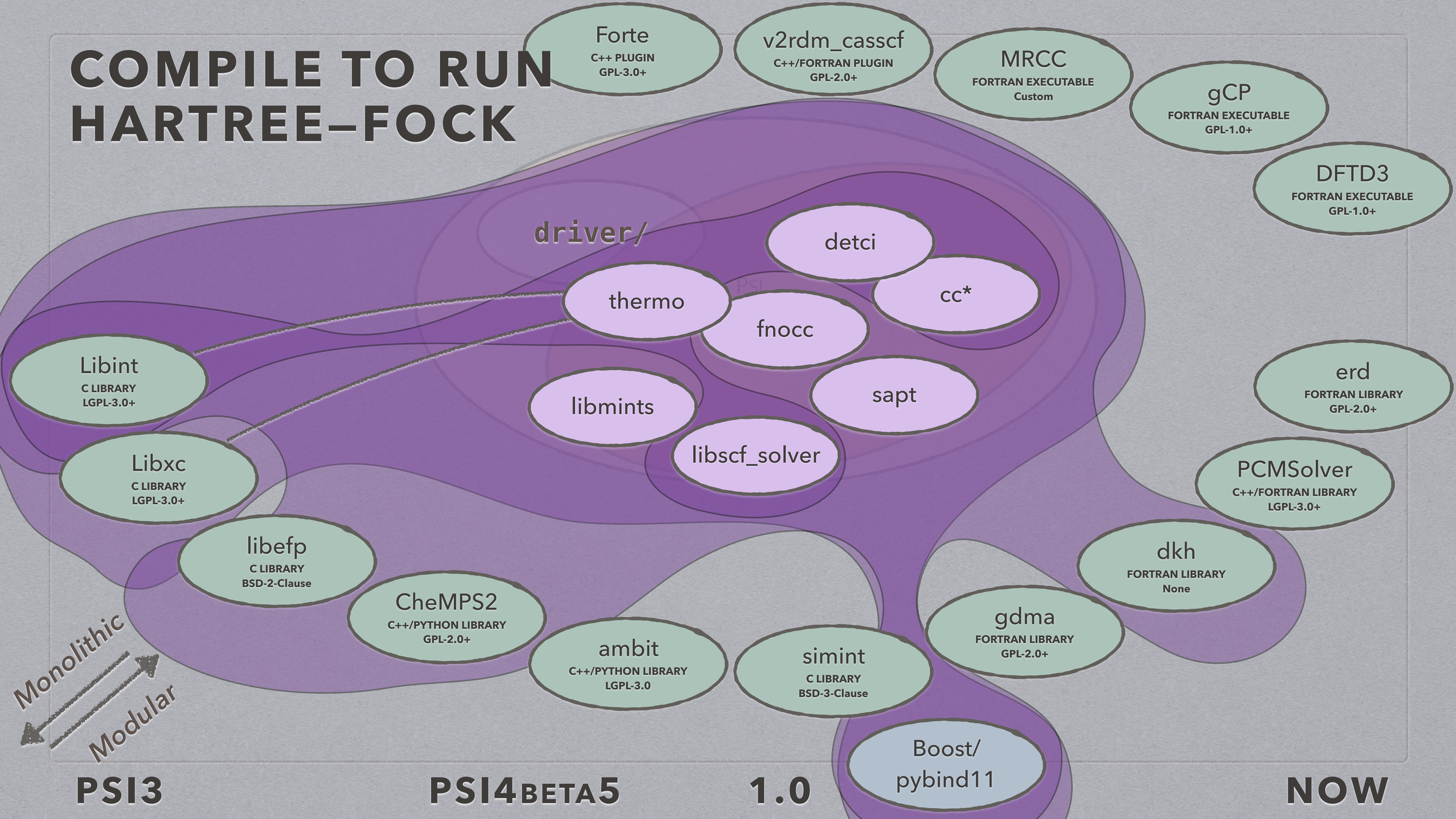
CAPABILITIES SHIPPED



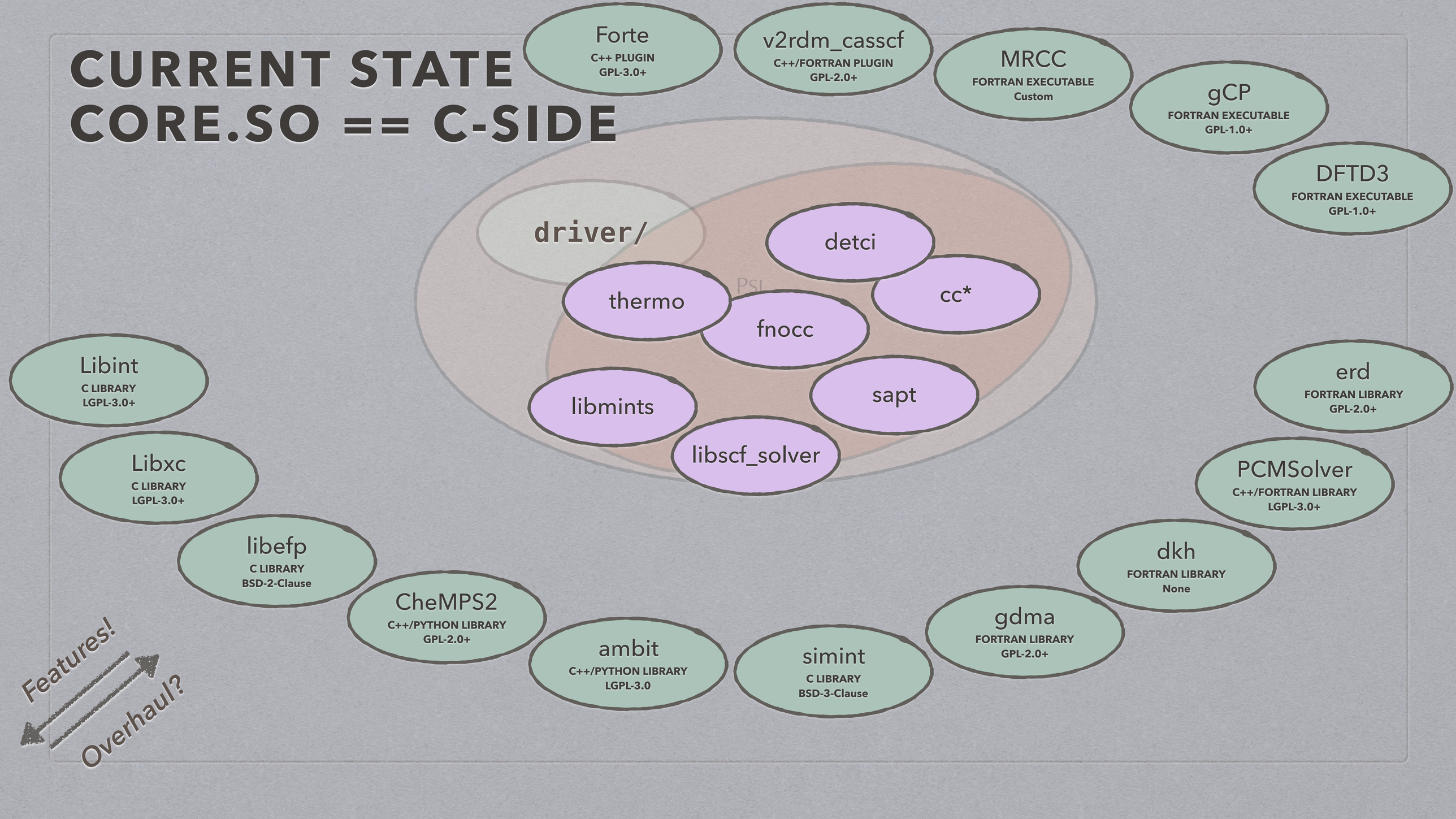
PSI3 PSI4BETA5 1.0 NOW


EMPIRE PSI4
COMMONWEALTH
Forte
C++ PLUGIN
GPL-3.0+
v2rdm_casscf
C++/FORTRAN PLUGIN
GPL-2.0+
MRCC
FORTRAN EXECUTABLE
Custom
gCP
FORTRAN EXECUTABLE
GPL-1.0+
DFTD3
FORTRAN EXECUTABLE
GPL-1.0+
driver/
detci
thermo
cc*
fnocc
sapt
libmints
libscf_solver
Libint
C LIBRARY
LGPL-3.0+
Libxc
C LIBRARY
LGPL-3.0+
erd
FORTRAN LIBRARY
GPL-2.0+
PCMSolver
C++/FORTRAN LIBRARY
LGPL-3.0+
libefp
C LIBRARY
BSD-2-Clause
dkh
FORTRAN LIBRARY
None
CheMPS2
C++/PYTHON LIBRARY
GPL-2.0+
gdma
FORTRAN LIBRARY
GPL-2.0+
ambit
C++/PYTHON LIBRARY
LGPL-3.0
simint
C LIBRARY
BSD-3-Clause
Boost/
pybind11
Code Freeze
Cont. Dev.
PSI3
PSI4BETA5
1.0
NOW

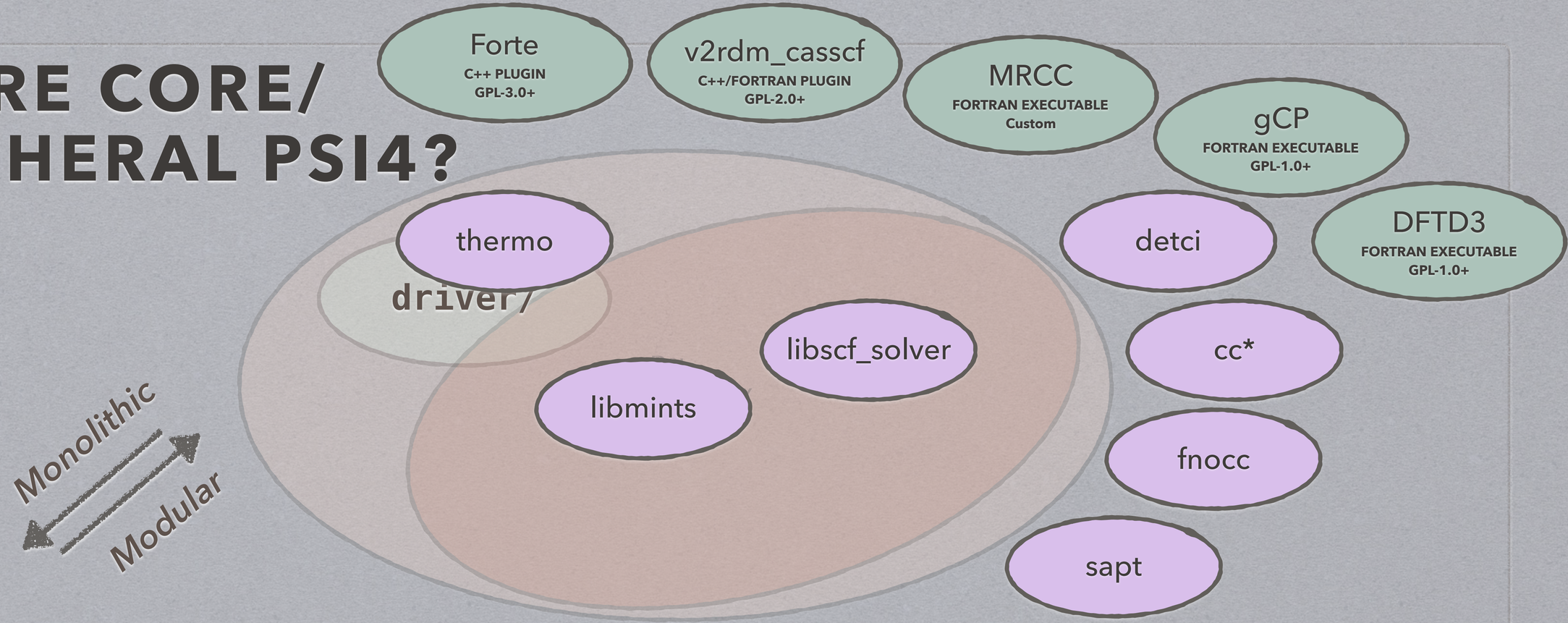
COMPILE TO RUN HARTREE–FOCK



CURRENT STATE
CORE.SO == C-SIDE



FUTURE CORE/ PERIPHERAL PSI4?

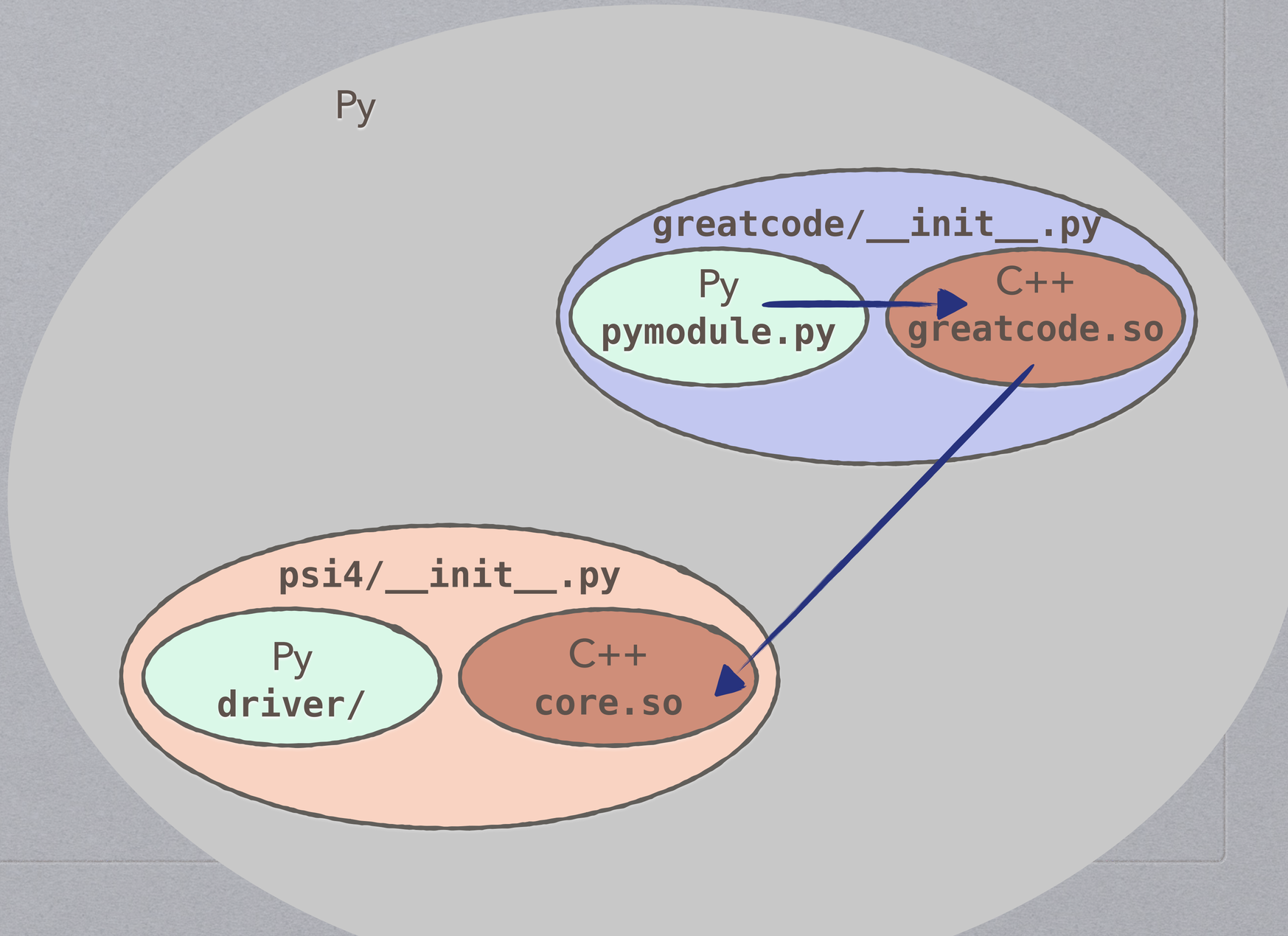


- Less distinction btwn methods *in* Psi and out of Psi in plugins, so new faculty can keep clear attribution of their research code.
- New contributors can grasp a finite level of commitment to working with psi4-core, rather than feeling uneasy obligation toward 1M loc.
- Existing projects can stay in-repo (if no longer under active dev) or move out to own repo (if you want dev branches and not compiling Psi a every CI).
- For users, everything gets distributed, so it looks just like before.

CHALLENGES OF STATEHOOD

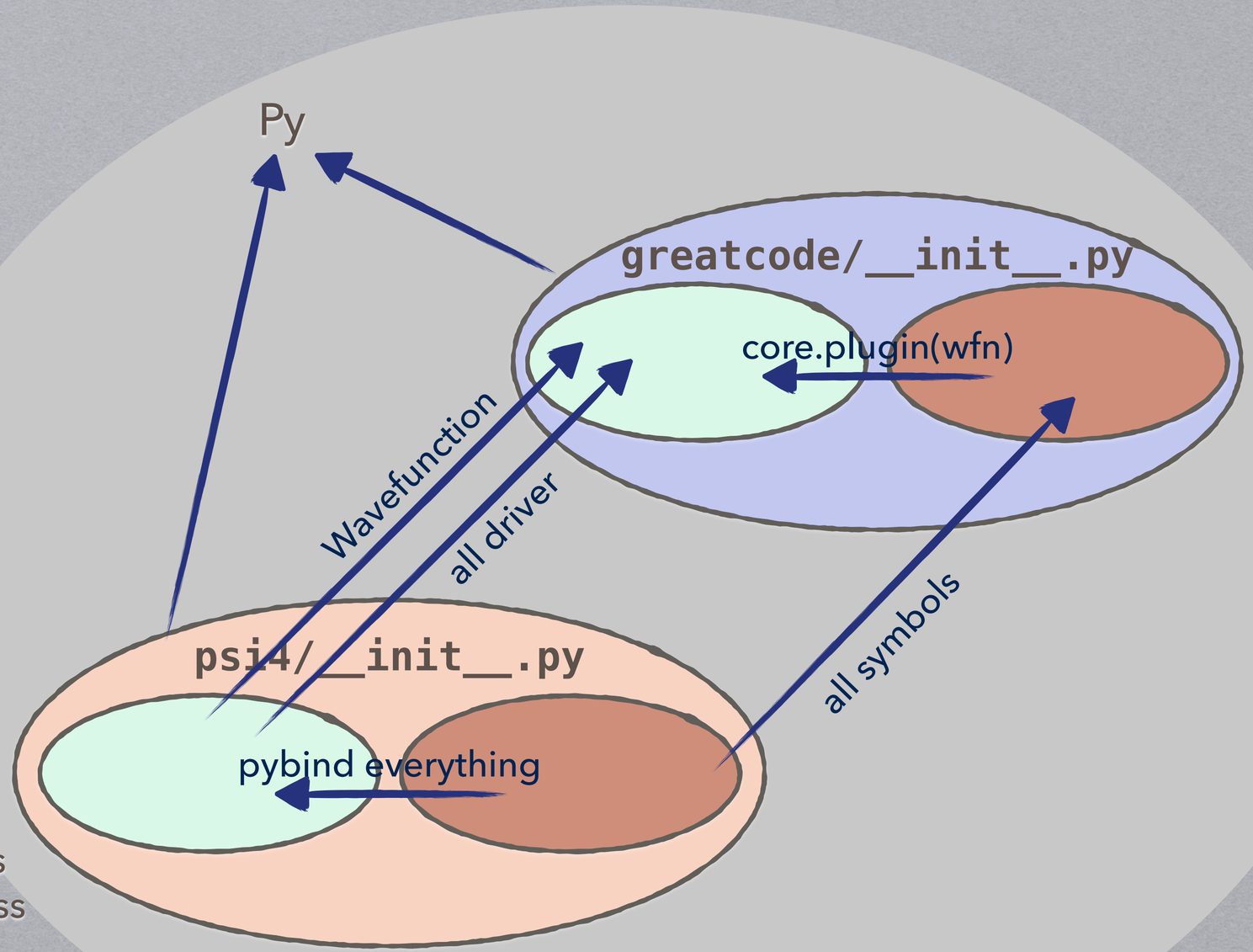
- Once liberated, Psi4 no longer providing framework for
  - build system, including dependencies
    - oddly, no one wants to manage a CMake system, much less a superbuild
  - documentation build
    - theme, linking, autodoc all a pain
    - all psi4's docs tools now available outside psi4 repo
  - versioning
    - if packages depend on you, I highly recommend the self-versioning scheme like psi4 has, so every commit is traceable
    - if a commit that's valuable downstream isn't released in short working order (couple weeks), then it needs to be otherwise addressable
  - options
    - uggh, but handled by current plugins (not nice for pure-Py to forcibly have to compile just to get options)
  - testing
    - a few inputs with Makefiles are better than nothing, but far better is a testing scheme that works for *installed* software. that is, pytest, not ctest. we can maintain current psithon test suite and run it through pytest. Just need to agree that pytest is authoritative.
    - downstream deps may not even build us, so no ctest and if they use a stable psi, won't even notice broken wrt us
- All of this calls for a more comprehensive plugin template, namely, a cookie cutter

PLUGIN FASHIONABILITY



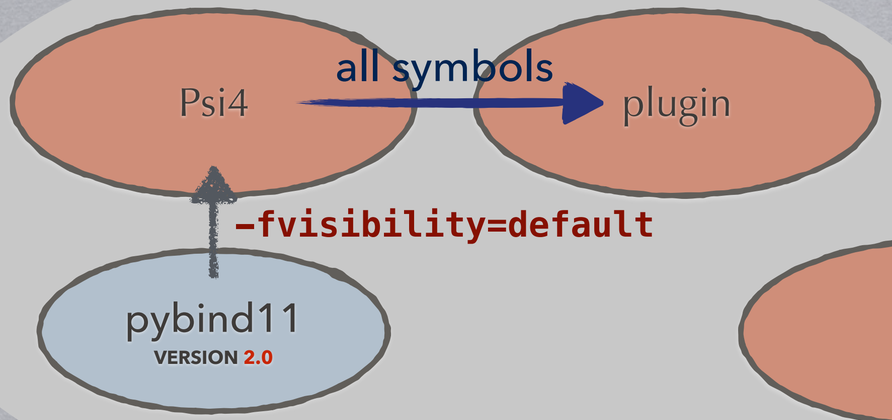
PLUGIN FASHIONABILITY

- a Psi method
  - C-side has access to all Psi C-libraries
  - Py-side has access to all Psi Py-/C-libraries
  - thus can shift Py/C boundary transparently to user as prototyping & efficiency demand
- a plugin method
  - C-side has access to all Psi C-libraries
  - Py-side has access to all Psi Py-/C-libraries
  - very limited Py/C interface w/i plugin
- plugins are excellent workspace for developing a Psi C-library (not that the compile time-penalty for in-tree lib dev is high anymore) but decidedly second-class for method prototyping or sustaining

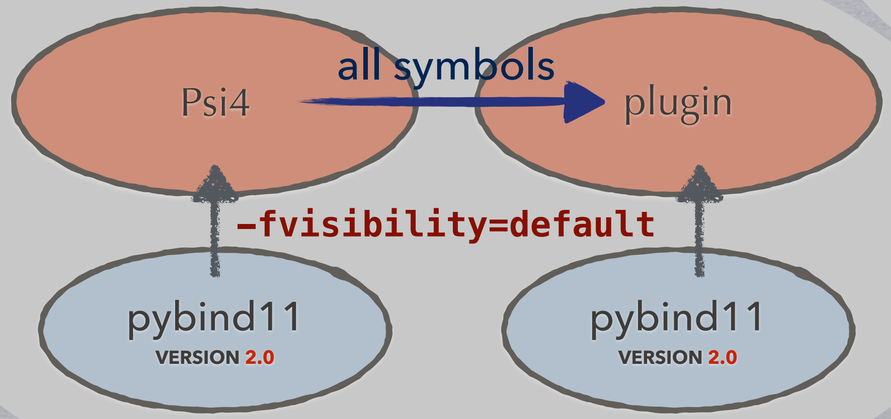


FVISIBILITY CRISIS

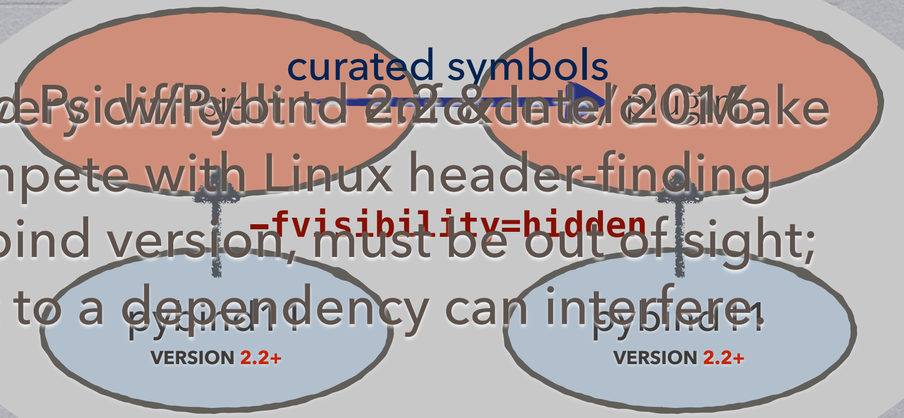
- Presently can't build Psi w/PyBind 2.2 & default CMake directives can't compete with Linux header-finding tech. To control pybind version, must be out of sight; even an install next to a dependency can interfere.



NOW

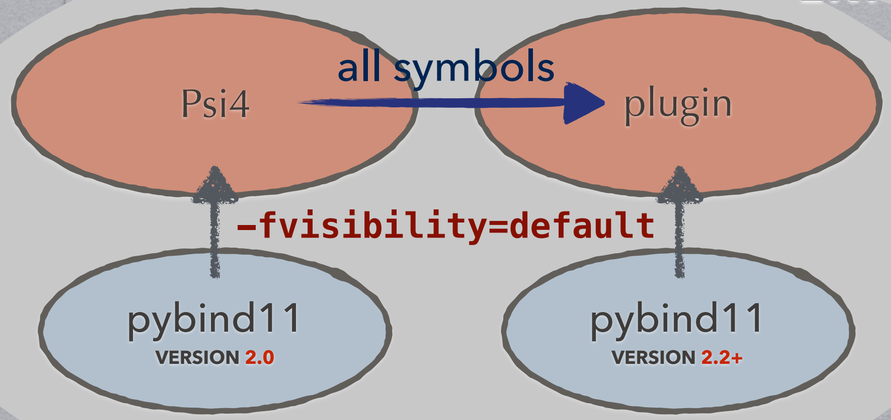


ENFORCE YEAR-OLD VERSION

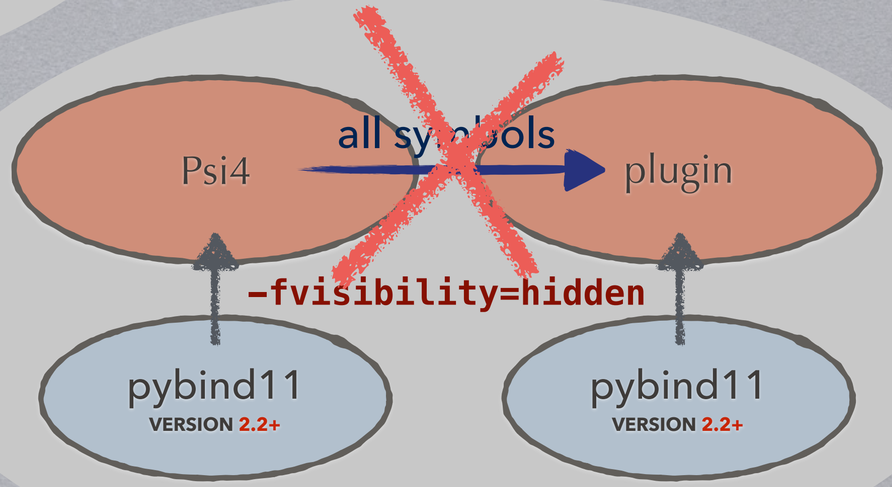


TACKLE SYMBOL CURATION

- so can pybind more deps
  - forte
  - pylibefp
- only psi4-core?



MIGHT WORK; ENFORCE PSI SOLE 2.0

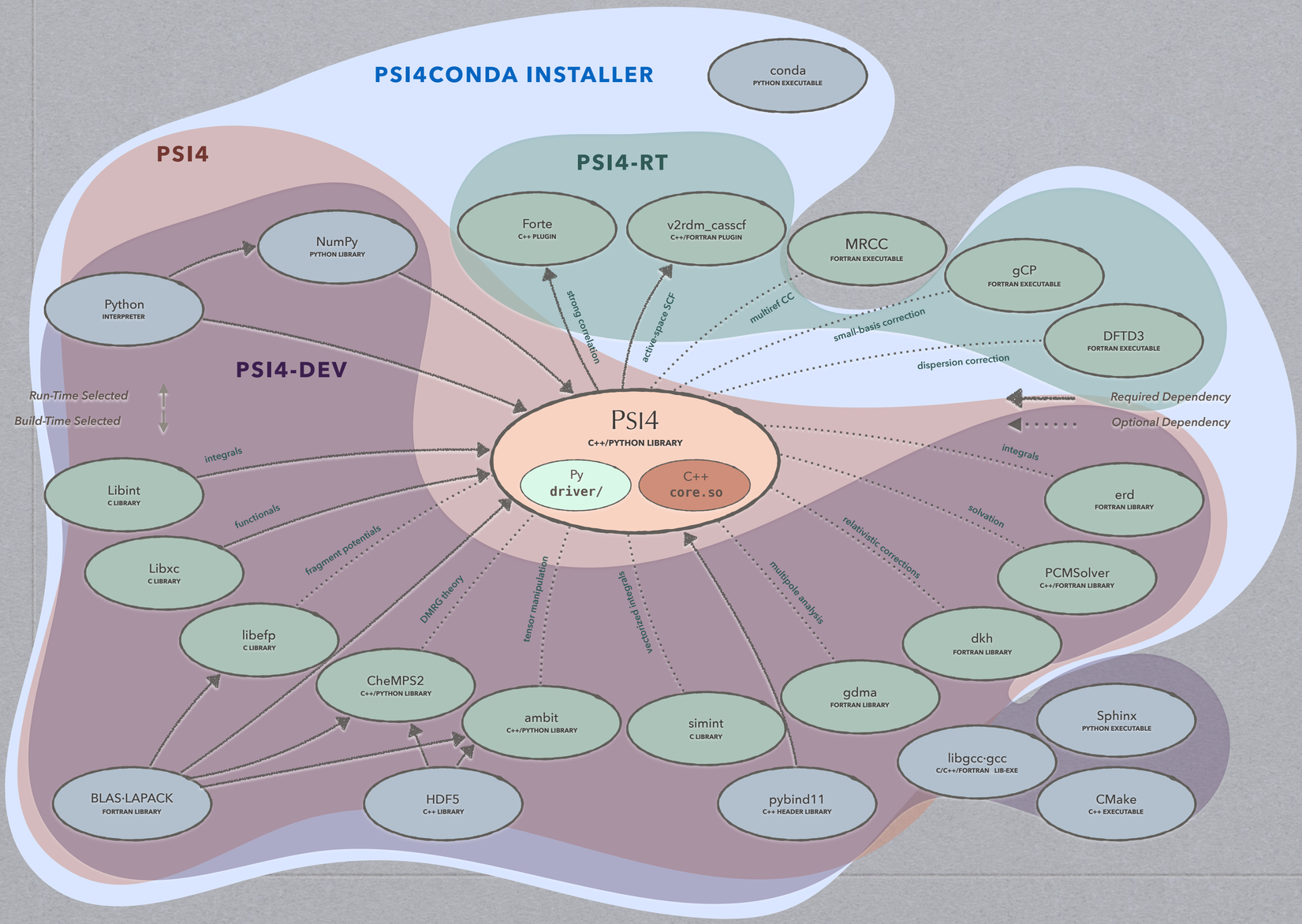


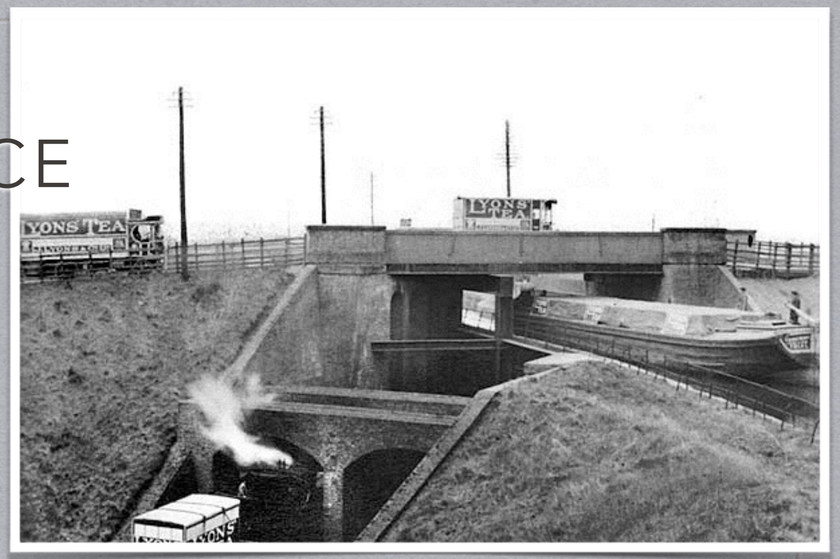
CAN'T ACCESS SYMBOLS

https://gcc.gnu.org/wiki/Visibility

CMAKE & CONDA FOR ALL

TRICKY OPTIMIZING FOR ALL USERS AT ONCE

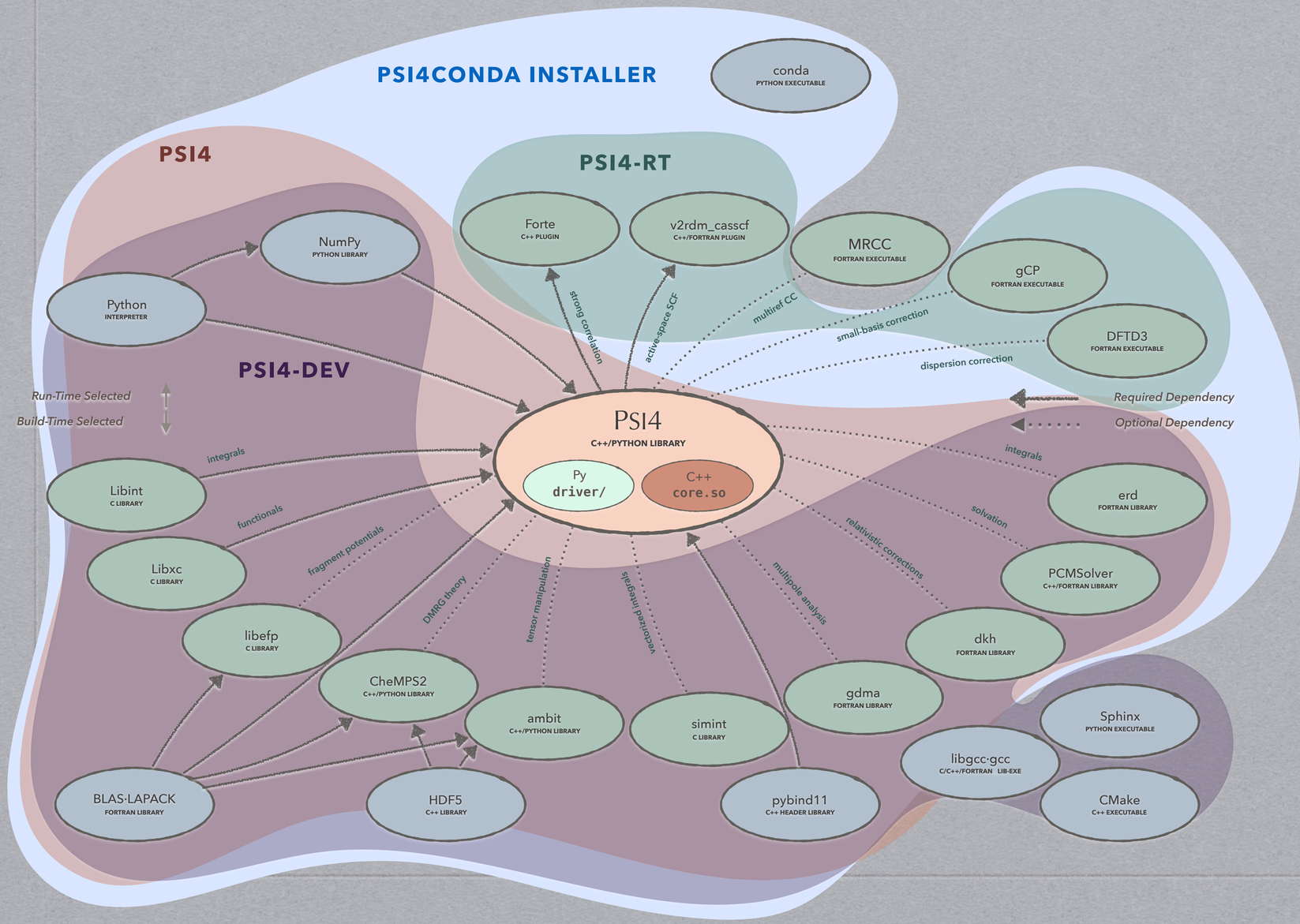




THREE BRIDGES, LONDON
(ROAD OVER CANAL OVER RAIL)

CMAKE & CONDA FOR ALL

TRICKY OPTIMIZING FOR ALL USERS AT ONCE



psi4conda

PY-WARY USERS: CONDA INSTALLER

BRINGS PSI4+DEPENDENCIES+ADD-ONS

PY-FRIENDLY USERS: CONDA PACKAGE

BRINGS PSI4+DEPENDENCIES+ADD-ONS

CORE DEVELOPERS: CONDA-ENABLED SUPERBUILD

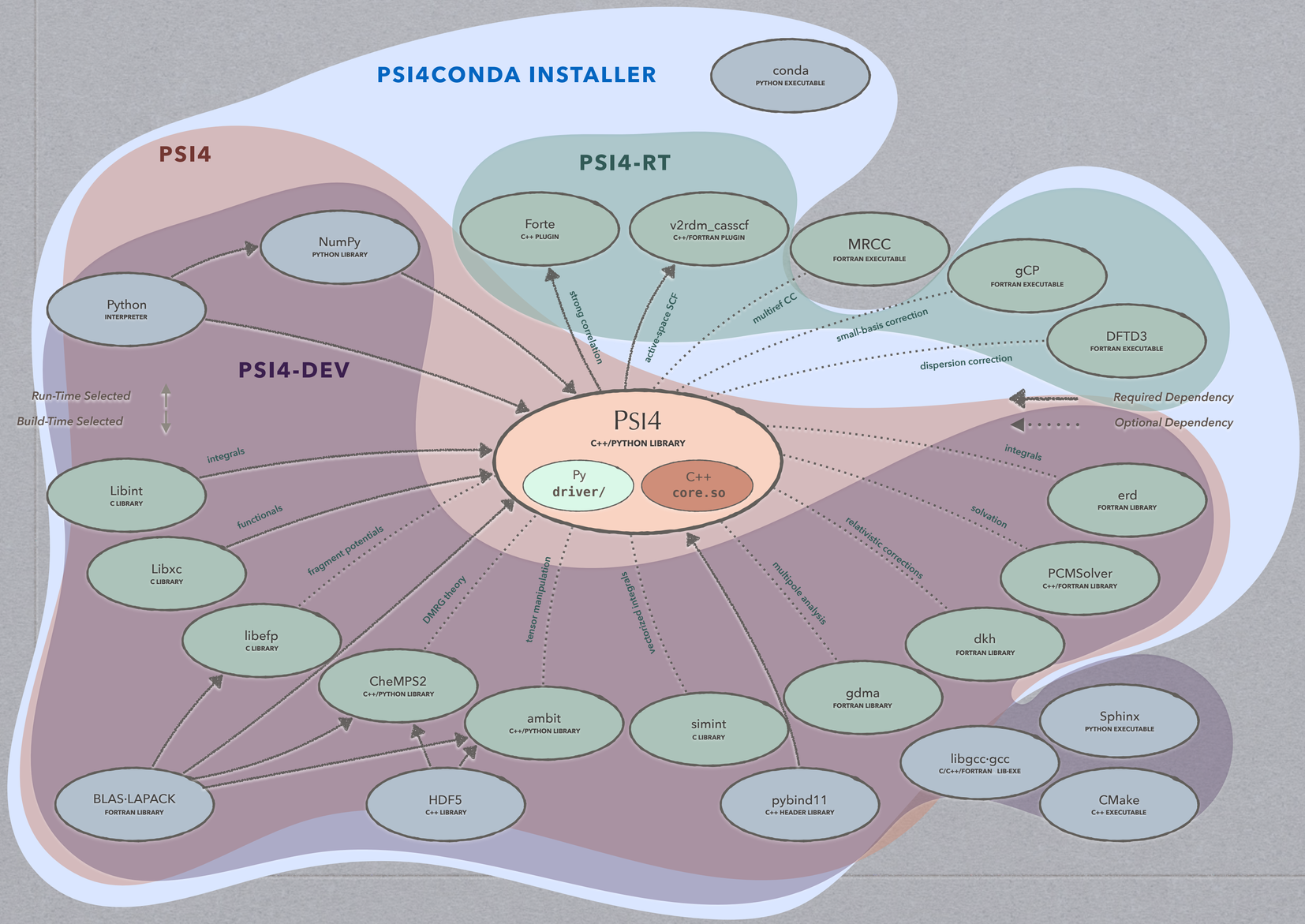
BRINGS DEPENDENCIES+ADD-ONS+CMAKE HINTS

PERIPHERAL DEVEL'S: CONDA-FACILITATED PLUGINS

BRINGS PSI4+DEVEL TOOLS+CMAKE HINTS

CMAKE & CONDA FOR ALL

TRICKY OPTIMIZING FOR ALL USERS AT ONCE



- Questions for you wrt psi4-dev:
  - I've copied over a headers pkg so you can get "mkl.h" via "-c psi4", but I'd rather not do too much of that.
  - Can you handle "conda create -n p4dev -c intel -c psi4/label/dev -c psi4" so you get mkl and numpy from Intel?
  - How much docs stuff in psi4-dev? Psi4 will tell you the long path with several packages & channels to get the docs deps. Do you really want them in psi4-dev? If so, can you handle "-c conda-forge -c astropy"?
- Any problems with psi4-path-advisor? Remember its "-h"

WHAT, PRAY, SHOULD I CONDA INSTALL?

1

```
conda create -n p4env
```

What do you want to do?

run PSI4	build PSI4 plugin	build PSI4	run PSI4 in Travis CI
`+= psi4 psi4-rt`	`+= psi4 psi4-dev`	`+= psi4-dev`	`+= ci-psi4 psi4`

2

Which Python version would you like to use (irrespective of any version installed)?

2.7	3.5	3.6
`+= python=2.7`	`+= python=3.5`	`+= python-3.6`

3

What computing platform are you targeting?

4aM

Mac

Execute the following:

```
sysctl machdep.cpu | grep features | grep -o -e SSE4.1 -e SSE4.2 -e AVX1.0 -e AVX2
```

(no output)	`SSE4.1`	`SSE4.2` (among others)	`AVX2` (among others)
Sorry, ancient hardware. You will not get a PSI4 conda package today.	You'll have to use the special **sse41** version, but PSI4 conda packages will work for you.	You'll have to use the special **sse41** version, but PSI4 conda packages will work for you.	Good, PSI4 conda packages will work for you.
	You CAN'T use Intel Distribution for Python conda packages.	You can use Intel Distribution for Python conda packages.	You can use Intel Distribution for Python conda packages.
	`+= sse41`	`+= sse41 -c intel`	`+= -c intel`

4aW

Windows

Do you have Bash on Ubuntu on Windows (https://msdn.microsoft.com/en-us/commandline/wsl/about)?

4bLW

Linux

Execute the following:

```
cat /proc/cpuinfo | grep flags | grep -o -e sse2 -e sse4_2 -e avx -e avx2
```

No output	Output includes: `sse2`	Output includes: `sse4_2`	Output includes: `avx` or `avx2`
Sorry, you will not get a PSI4 conda package today.	Good, PSI4 conda packages will work for you.	Good, PSI4 conda packages will work for you. Also, you can use Intel Distribution for Python conda packages.	Good, PSI4 conda packages are optimized for your instruction set. Also, you can use Intel Distribution for Python conda packages.
		`+= -c intel`	`+= -c intel`

Do you get the following:

```
OMP: Error #100: Fatal system error detected.
```

Execute the following:

```
export KMP_AFFINITY=disabled
```

4cLW

Execute the following:

```
arch
```

`x86_64`	Anything else
Good, PSI4 conda packages will work for you.	Sorry, 32-bit. You will not get a PSI4 conda package today.

4dLW

What does the following return?

```
ldd --version
```

< `2.7` (*e.g.*, 2.6, 2.2.4, 1.09)	≥ `2.7` (*e.g.*, 2.7, 2.14, 2.26)
Sorry, c. 2007 fundamental library. You will not get a PSI4 conda package today.	Good, PSI4 conda packages will work for you.

5

Do you want the last stable/ tagged release or a development/nightly build?

latest release	nightly build
`+= -c psi4`	`+= -c psi4/label/dev -c psi4`

6

Do you want to build the docs?

yes (secondary cmd below)

```
conda install sphinx-psi-theme cloud_sptheme astropy-helpers graphviz python-graphviz -c conda-forge -c astropy -c psi4
```

DOWN WITH GLOBALS

PSI4BETA5

[US] | https://github.com/psi4/psi4archive/blob/4.0b5/include/psi4-dec.h

Psi4 Docs | Comcast | BFDb | PDFdb | txu-oclc-5570528-... | Pau

```
public:
    class Environment
    {
        std::map<std::string, std::string> environment_;
        unsigned long int memory_;
        int nthread_;

        boost::shared_ptr<Molecule> molecule_;
        SharedMatrix gradient_;
        boost::shared_ptr<Vector> frequencies_;
        boost::shared_ptr<Wavefunction> wavefunction_;
        boost::shared_ptr<PointGroup> parent_symmetry_;
```

PSI4 1.0

. [US] | https://github.com/psi4/psi4archive/blob/1.0.x/include/process.h

Psi4 Docs | Comcast | BFDb | PDFdb | txu-oclc-5570528-... | P

```
public:
    class Environment
    {
        std::map<std::string, std::string> environment_;
        unsigned long int memory_;
        int nthread_;

        boost::shared_ptr<Molecule> molecule_;
        SharedMatrix gradient_;
        boost::shared_ptr<efp::EFP> efp_;
        SharedMatrix efp_torque_;
        boost::shared_ptr<Vector> frequencies_;
        boost::shared_ptr<PointGroup> parent_symmetry_;

        boost::shared_ptr<Molecule> legacy_molecule_;
        boost::shared_ptr<Wavefunction> legacy_wavefunction_;
```

NOW

] | https://github.com/psi4/psi4/blob/master/psi4/src/psi4/libpsi4util/process.h

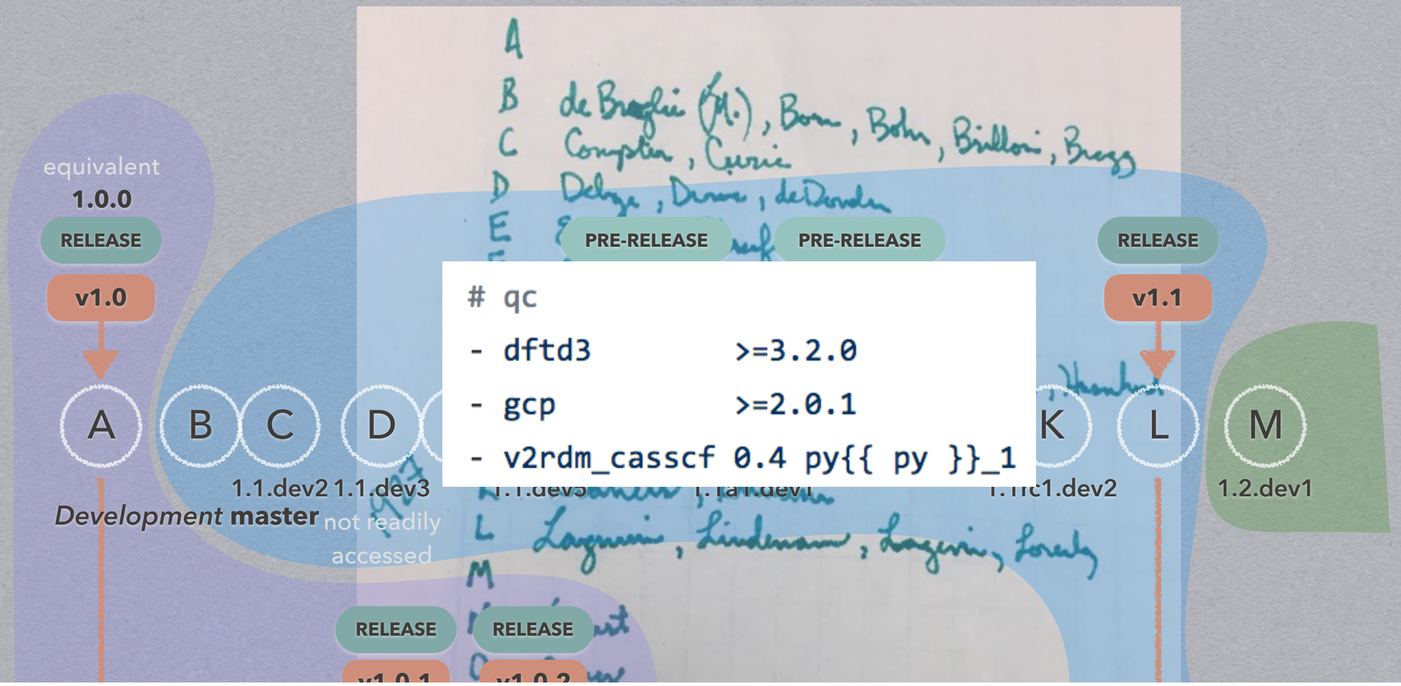
Docs | Comcast | BFDb | PDFdb | txu-oclc-5570528-... | Paul Thomas

```
class Environment {
    std::map<std::string, std::string> environment_;
    size_t memory_;
    int nthread_;
    std::string datadir_;

    std::shared_ptr<Molecule> molecule_;
    SharedMatrix gradient_;
    std::shared_ptr<efp::EFP> efp_;
    SharedMatrix efp_torque_;
    std::shared_ptr<Vector> frequencies_;
    std::shared_ptr<PointGroup> parent_symmetry_;

    std::shared_ptr<Molecule> legacy_molecule_;
    std::shared_ptr<Wavefunction> legacy_wavefunction_;
```

VERSIONING
PRAY I DON'T ALTER IT ANY FURTHER




```
# qc
- dftd3          >=3.2.0
- gcp            >=2.0.1
- v2rdm_casscf 0.4 py{{ py }}_1
```

- Last year's scheme is working great! …
  - needs a checklist and a calm mind to bump versions, but that's ok
- … for computers
- … for advertisers (1.1 hoopla)
- … not so much for people or downstream devs
  - have to keep rebuilding for ABI changes
  - have to remember how many commits past tag their feature of interest is

vertising
from API-
Psi4 1.13)

```
using_psi4_libxc = pytest.mark.skipif(is_psi4_new_enough("1.2a1.dev100") is False,
                                      reason="Psi4 does not include DFT rewrite to use Libxc. Update to development head")

using_psi4_efpmints = pytest.mark.skipif(is_psi4_new_enough("1.2a1.dev507") is False,
                                         reason="Psi4 does not include EFP integrals in mints. Update to development head")

using_psi4_sadpy = pytest.mark.skipif(is_psi4_new_enough("1.2a1.dev600") is False,
                                      reason="Psi4 does not include SADGuess bindings. Update to development head")
```

SCIPY 2018

- Austin, TX
- July 11–13
- MolSSI & others organizing a Chemistry & Materials mini symposium
- talk slots allocated proportionately to abstracts submitted, so think about what python aspects of your science are useful to others
- meet developers of your favorite software tools

Psi4

OPEN-SOURCE QUANTUM CHEMISTRY

JK object

DF-CC

SAPT(CCD)

OMPn

A/F/O-SAPT

ADC(2)

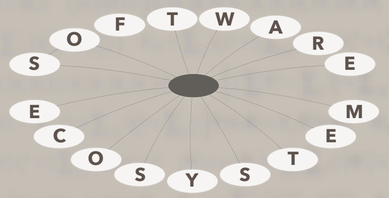
X2C

PCM

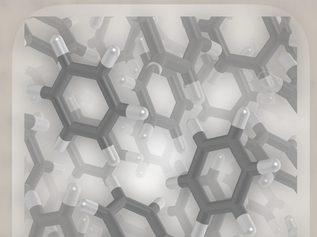
VMFC

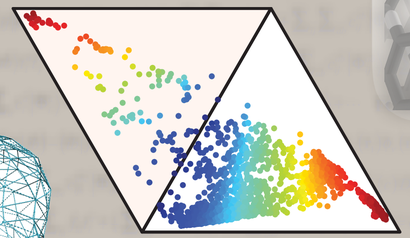
VMFC

DF-MCSCF



Psi4NumPy





Psithon
PsiAPI

Psi4Education