Geometry driver pattern in DFTB+

Bálint Aradi*

Bremen Center for Computational Materials Science

University of Bremen

Fortran-OOP discussions, 2021-11-16

* and the DFTB+ developers group

Pattern requirements

Intent

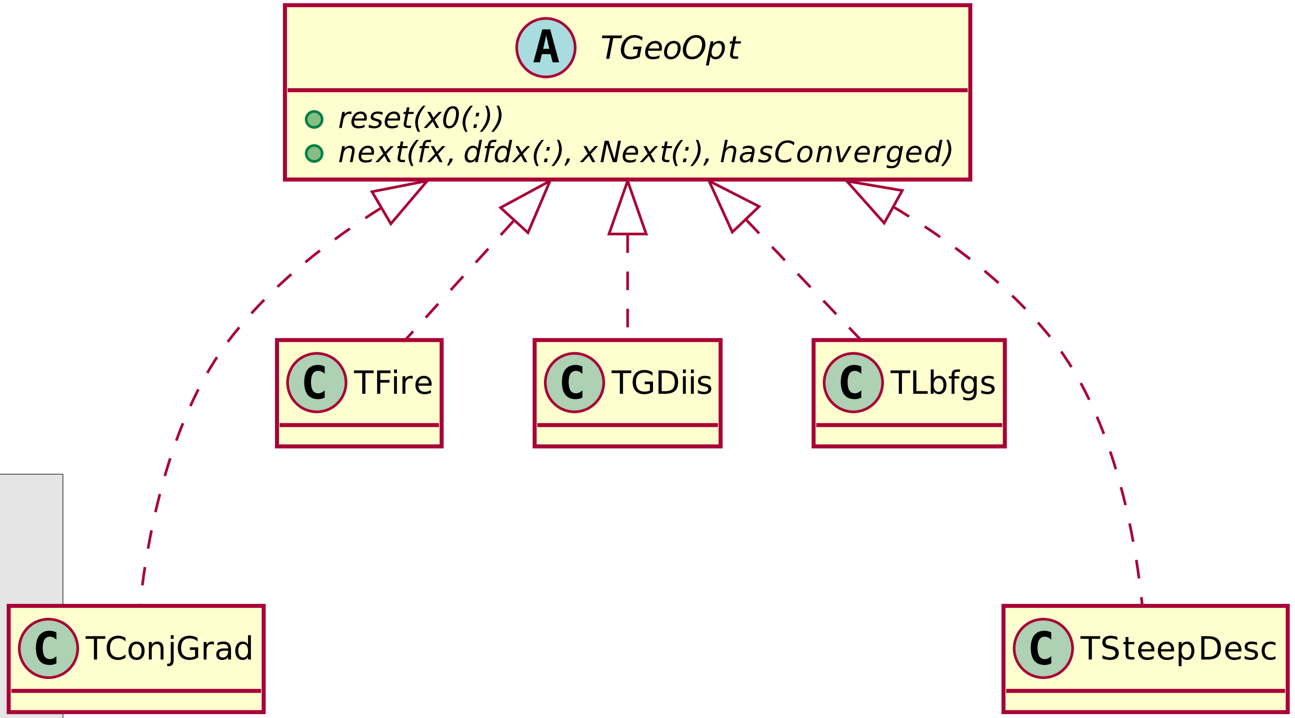
- Optimize geometry and update various geometry dependent components

Contraints

- Arbitrary geometry driver (conjugate gradient, steepest descent, etc.)
- Arbitrary geometry dependent components
- Driving a quantity should not require subclassing a special class (flexibility)

Geometry driver as iterator

- Geometry driver interface defined through abstract type
- Geometry driver is implemented as an **iterator**



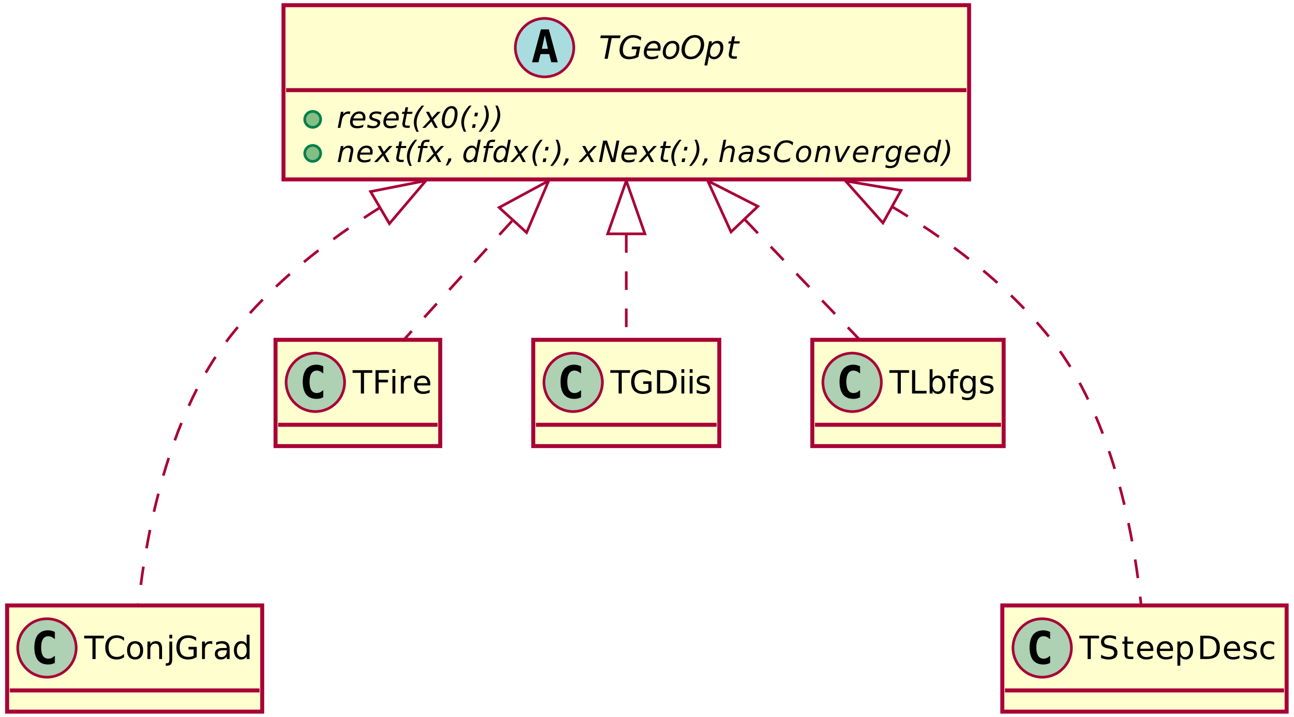
```
type, abstract :: TGeoOpt
contains
  procedure(reset), deferred :: reset
  procedure(next), deferred :: next
end type TGeoOpt
```

```
abstract interface
  subroutine reset(this, x0)
  subroutine next(this,fx, dfdx, xNext, hasConverged)
```

Geometry driver as iterator

- Actual drivers extend abstract type
- Drivers state kept between interations in private variables of the driver instance

```
type, extends(TGeoOpt) :: TConjGrad
  private
  integer :: state
  integer, allocatable :: gg(:)
contains
  procedure :: reset
  procedure :: next
end type TConjgGrad
```

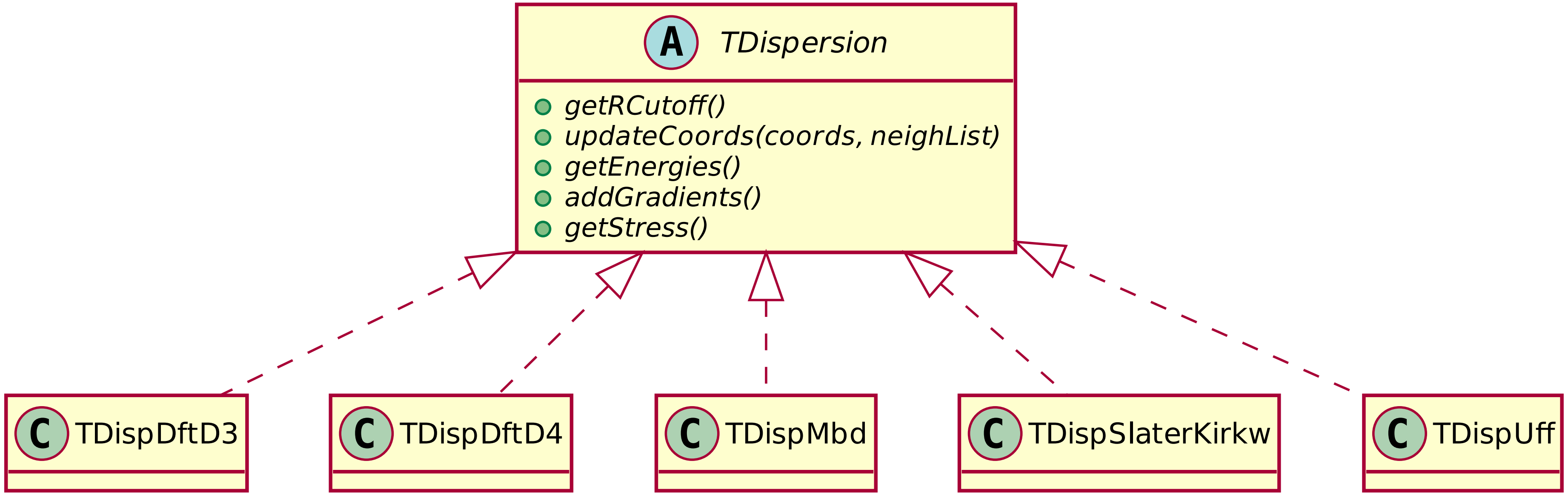


A TGeoOpt

- reset(x0(:))
- next(fx, dfdx(:), xNext(:), hasConverged)

```
subroutine reset(this, …)
  class(TConjGrad), intent(inout) :: this
:
subroutine next(this, …)
   class(TConjGrad), intent(inout) :: this
```

Dispersion interaction as calculator



- Dispersion interaction acts as a **calculator**
  → variables (coords, neighbours)
  ← calculated quantities (energy, gradient, stress)

- Special calls notify the calculator about changes in variable values

- Special call queries the calculator for calculated quantities

- Actual dispersion models extend abstract type

```
type, extends(TDispersion) :: TDispDftD4
```

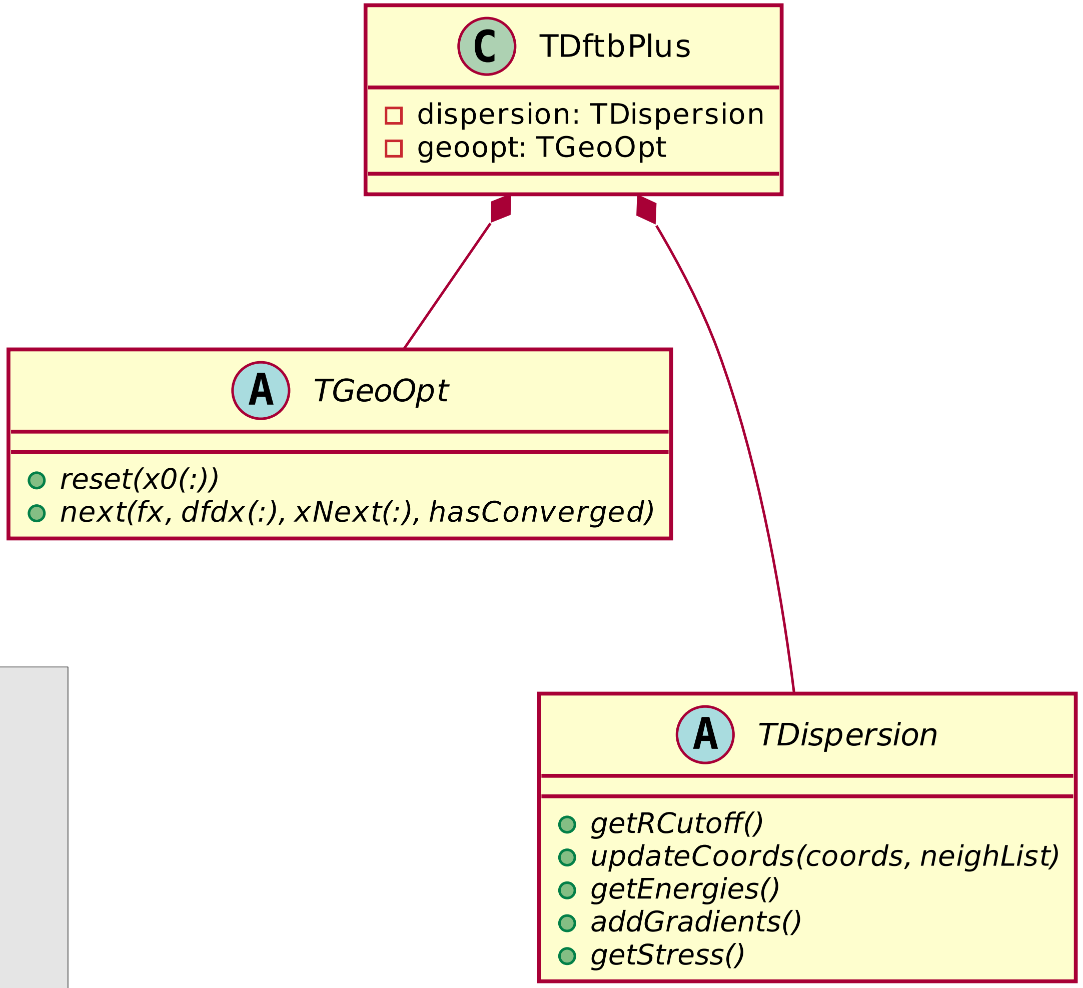
Main program with driver and calculator components

- Main program only deals with abstract class interfaces

Initialization

```
class(TGeoOpt), allocatable :: geoOpt
type(TConjGrad), allocatable :: conjGr
allocate(conjGr)
call TConjGrad_init(conjGr, …)
call move_alloc(conjGr, geoOpt)
```

```
class(TDispersion), allocatable :: disp
type(TDispDftD4), allocatable :: dispDftD4
allocate(dispDftD4)
call TDispDftD4_init(dispMbd, …)
call move_alloc(dispMbd, disp)
```



C TDftbPlus

□ dispersion: TDispersion
□ geoopt: TGeoOpt

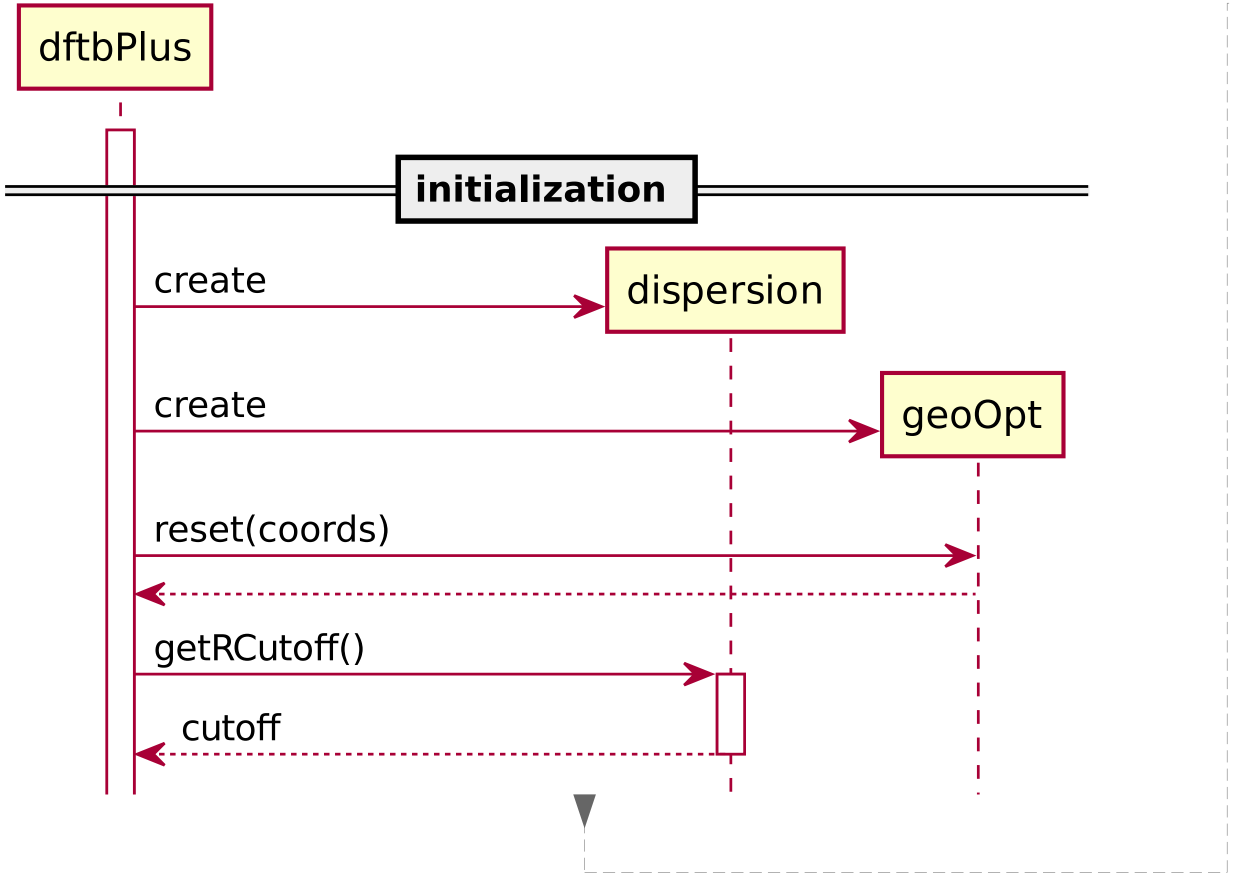
A *TGeoOpt*

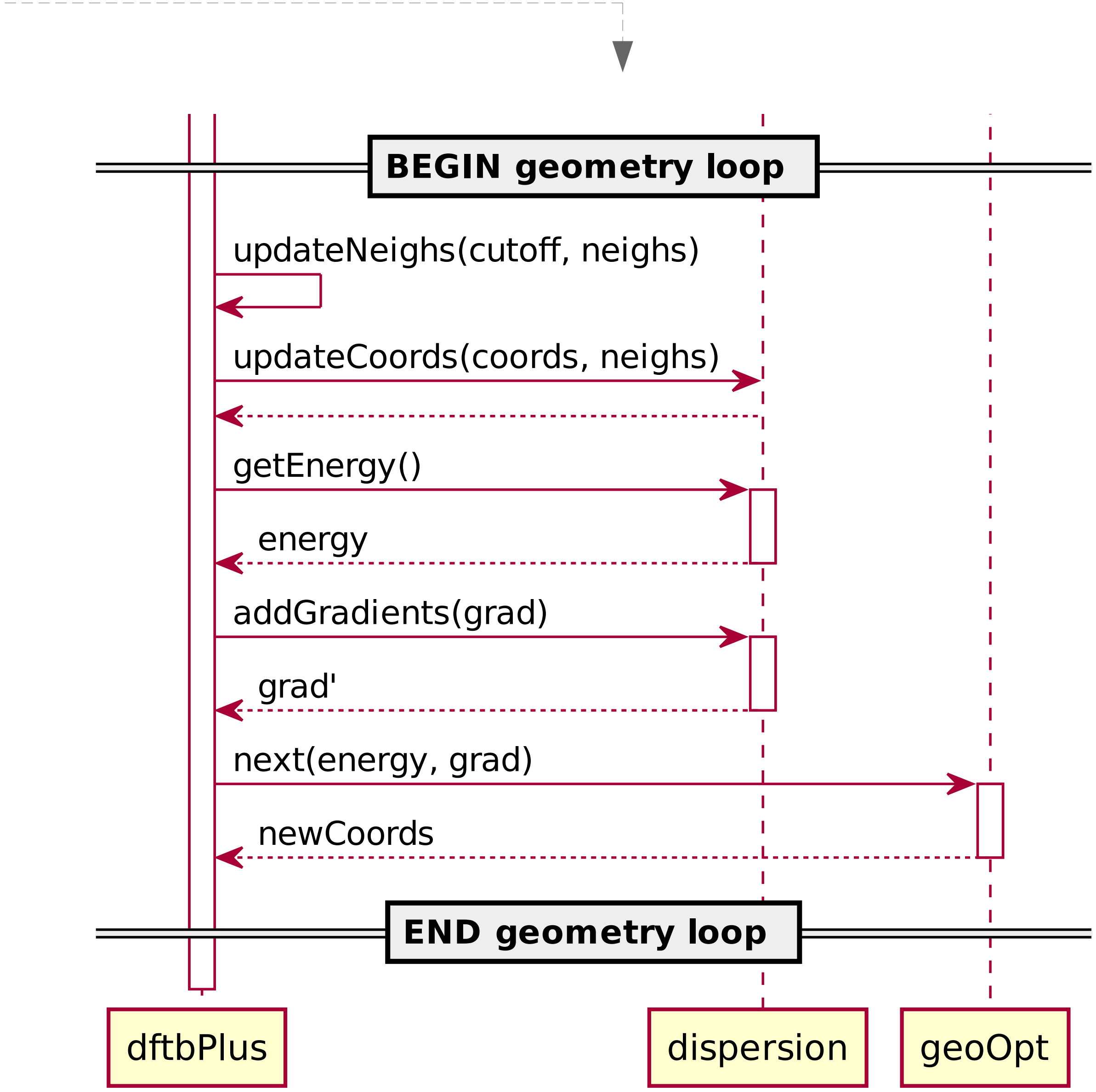
○ *reset(x0(:))*
○ *next(fx, dfdx(:), xNext(:), hasConverged)*

A *TDispersion*

○ *getRCutoff()*
○ *updateCoords(coords, neighList)*
○ *getEnergies()*
○ *addGradients()*
○ *getStress()*

Interaction between driver and calculator





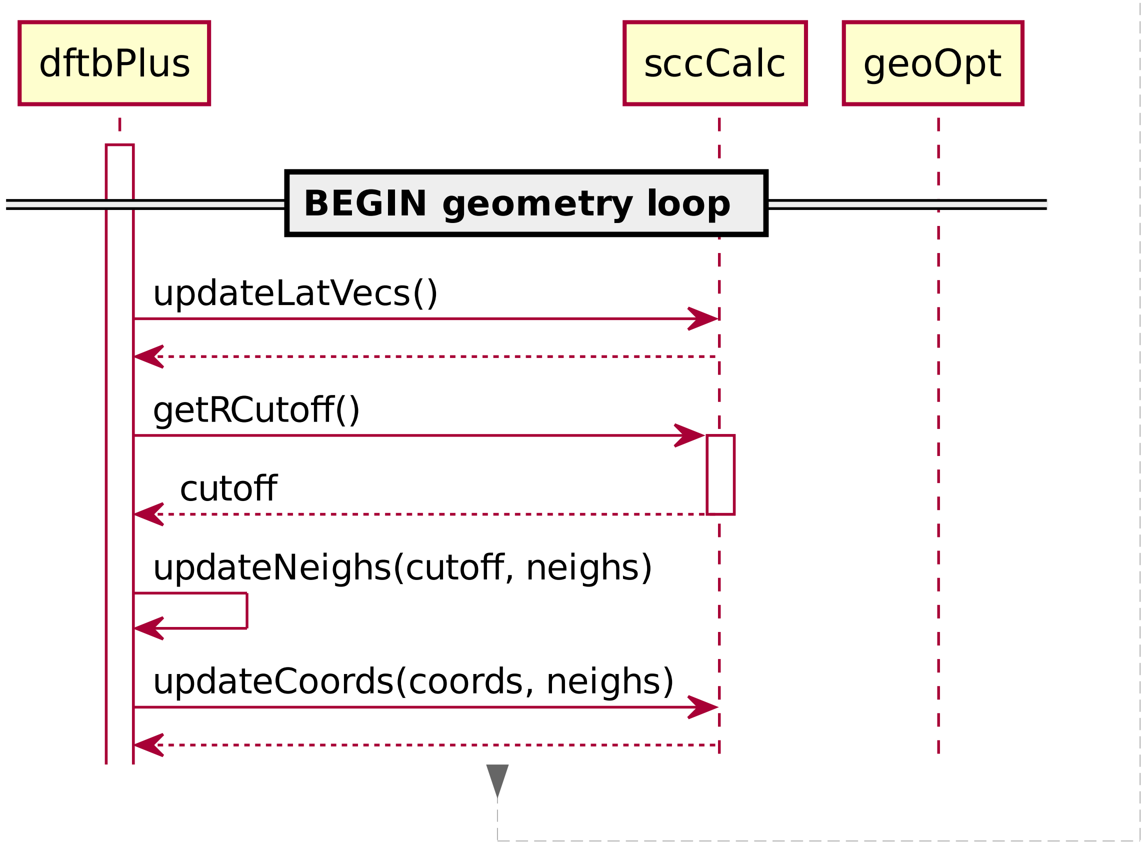
dftbPlus = **Pupeteer** [1]

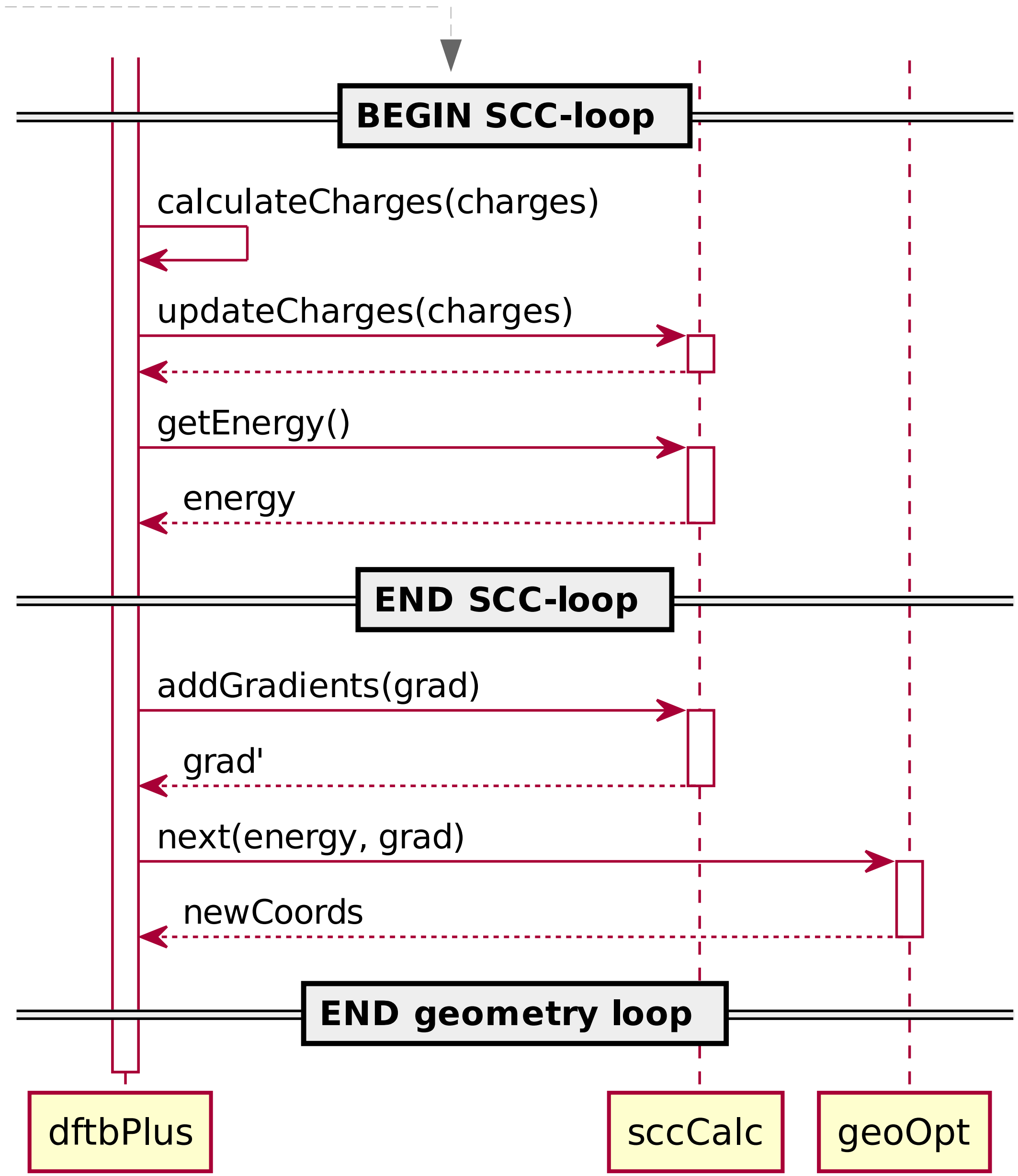
- Orchestrates the interaction between driver and calculator

[1] D. Rouson, J. Xia, X. Xu, Scientific software design, 2011

Calculator with multiple input variables

- Different variable updates with different frequency

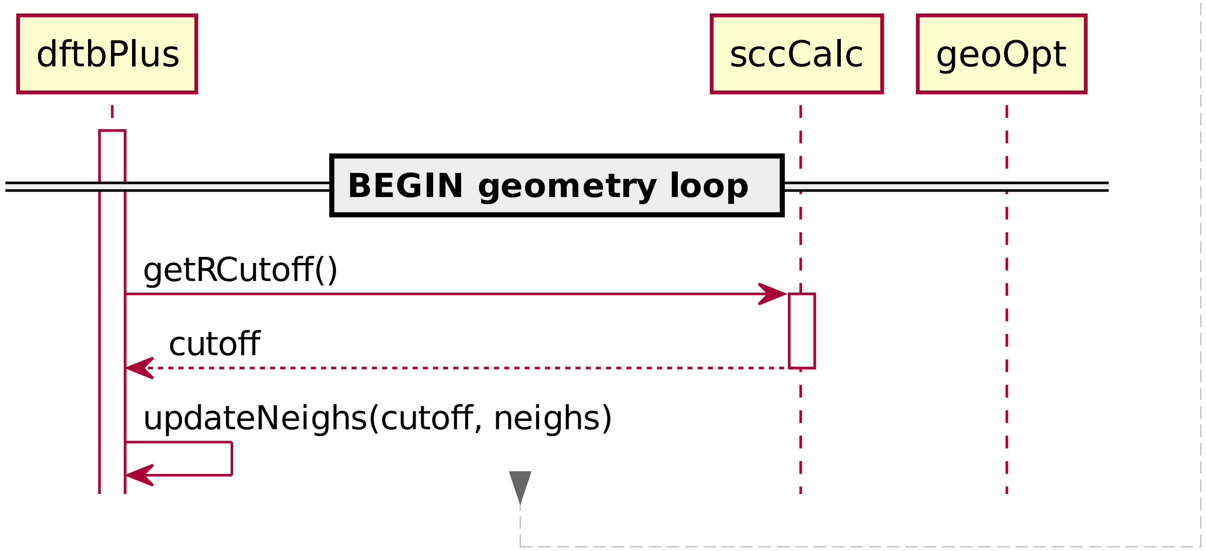




Problem: Calculator may have to cache lot of data (e.g. geometry needed when calculating energy arising due to updated charges)

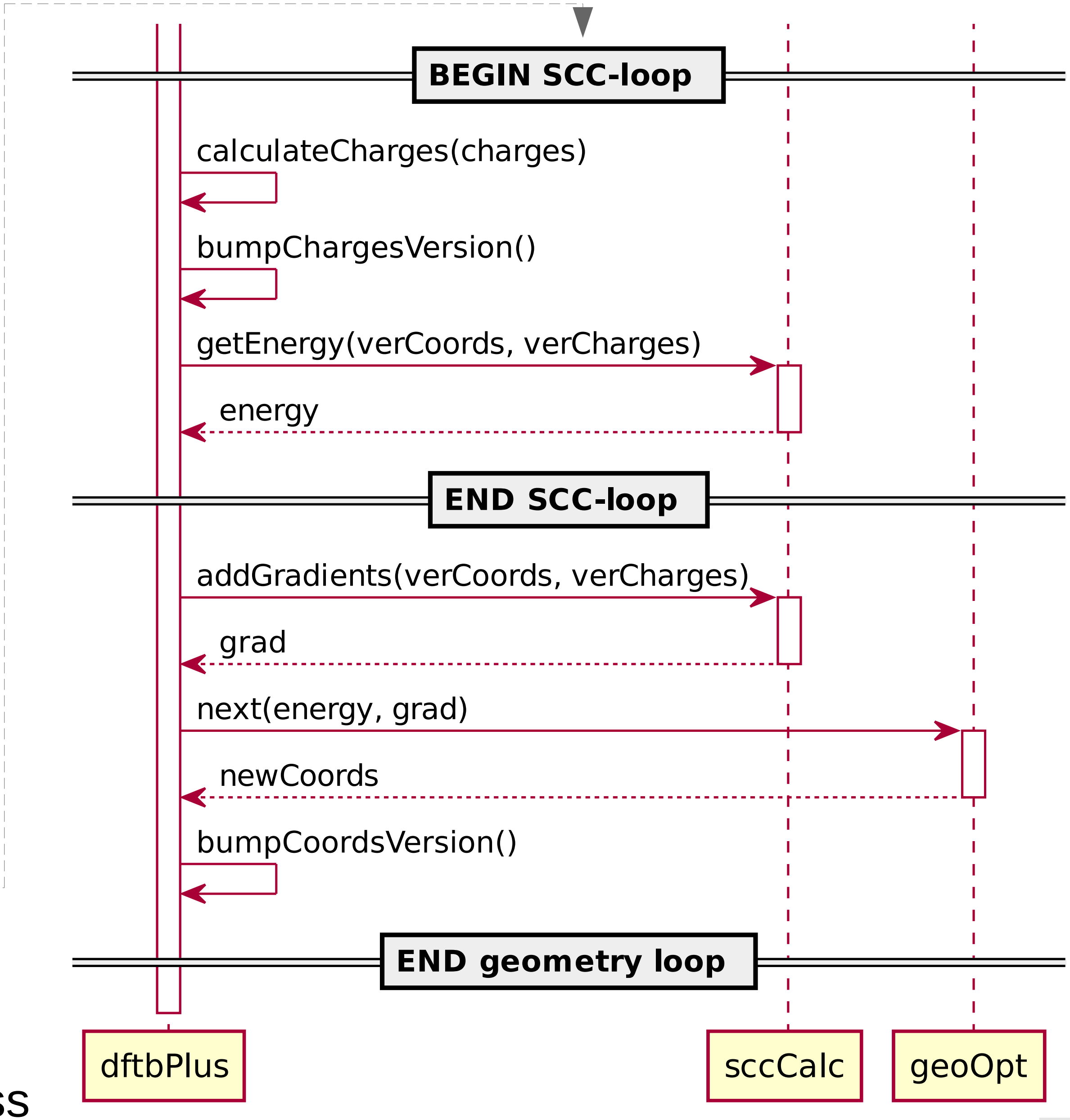
Improvement: Calculator with versioned input variables?

- No special input variable update calls
- When querying calculator, all necessary input variables are passed
- Input variables have version identifiers
  → calculator can recognize changes



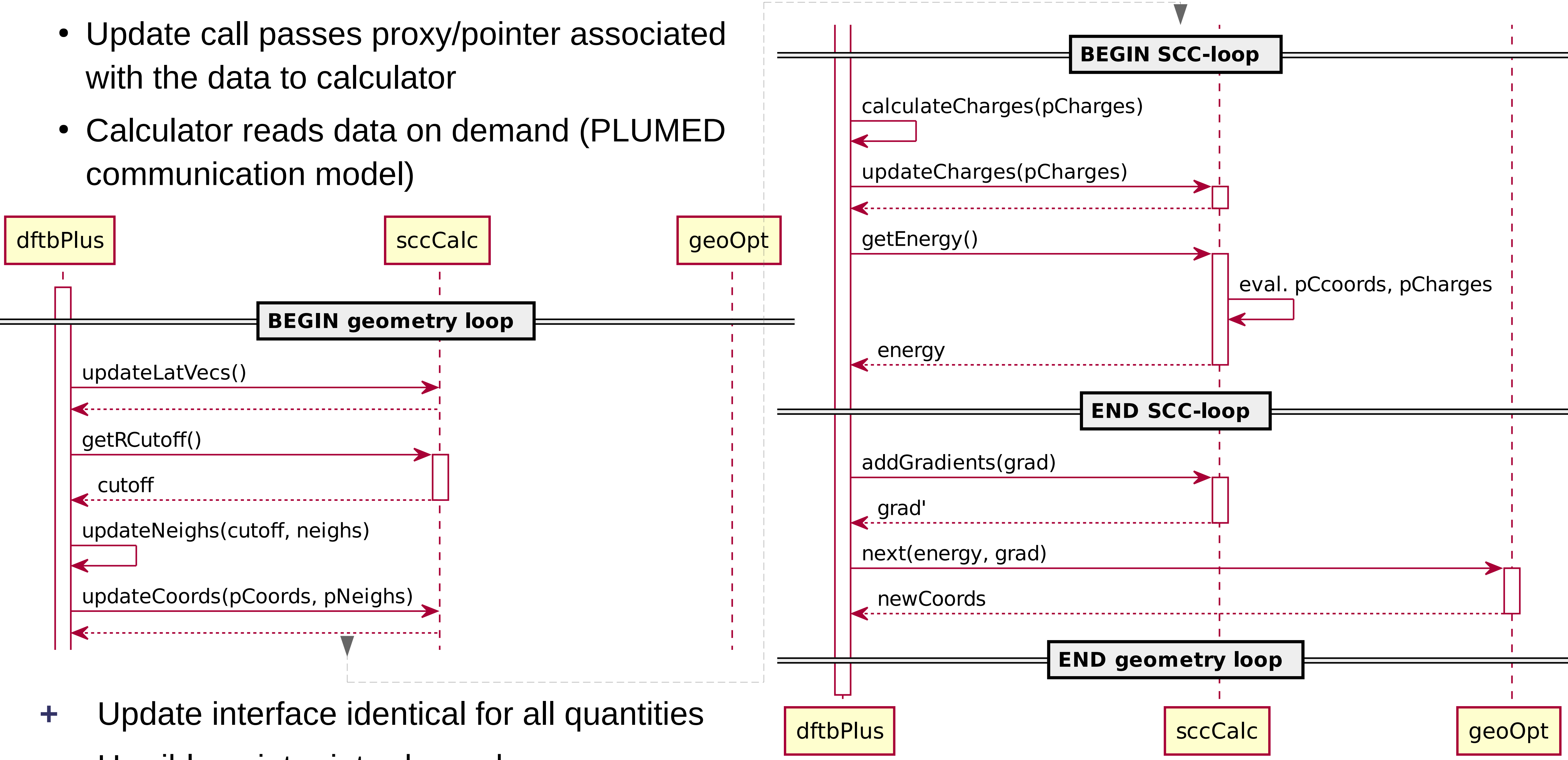
Problem

No generic interface for updating a given quantity, special query call needed for every (abstract) class



Improvement: Calculator with input value pointers/proxies?

- Update call passes proxy/pointer associated with the data to calculator
- Calculator reads data on demand (PLUMED communication model)



+ Update interface identical for all quantities

- Horrible pointer interdependency mess