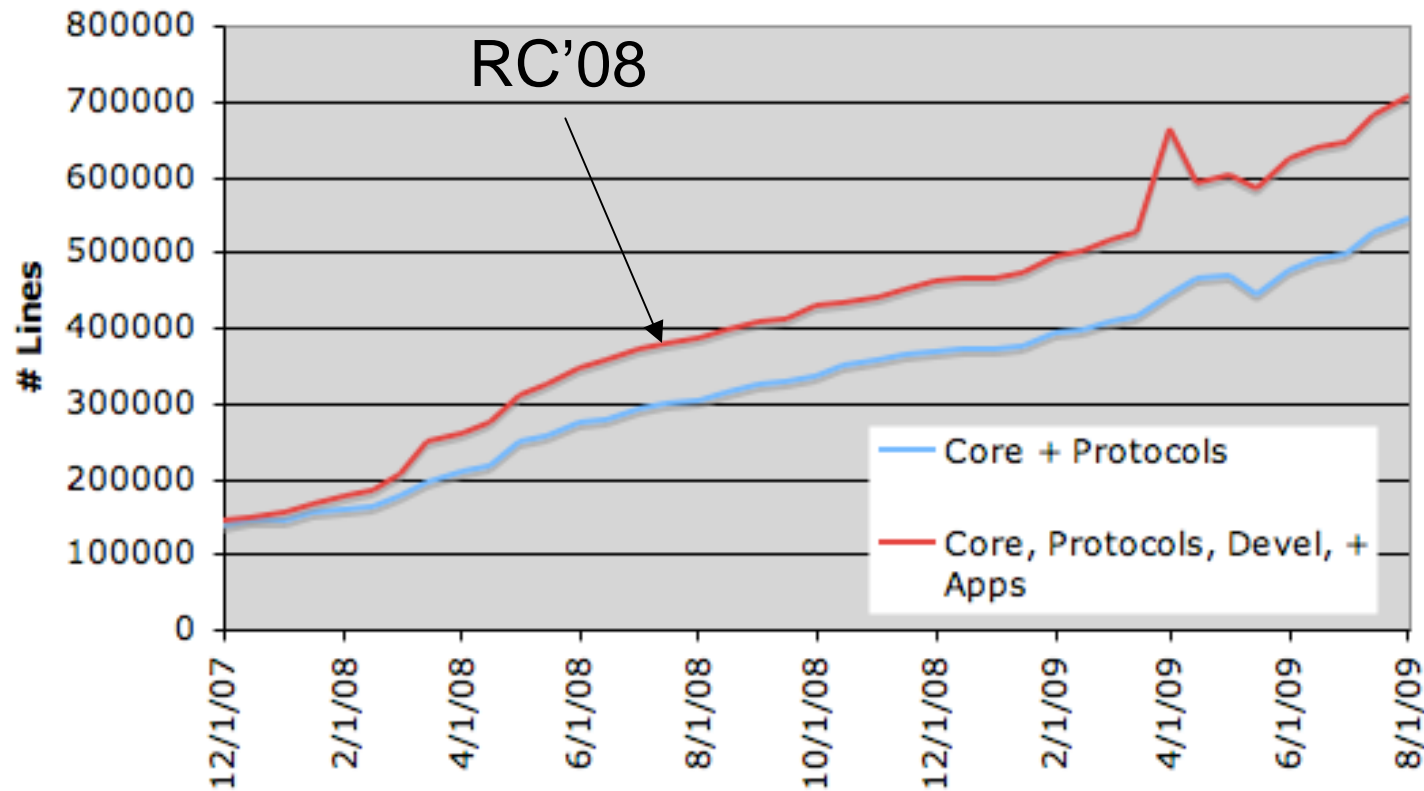


Outstanding Hurdles

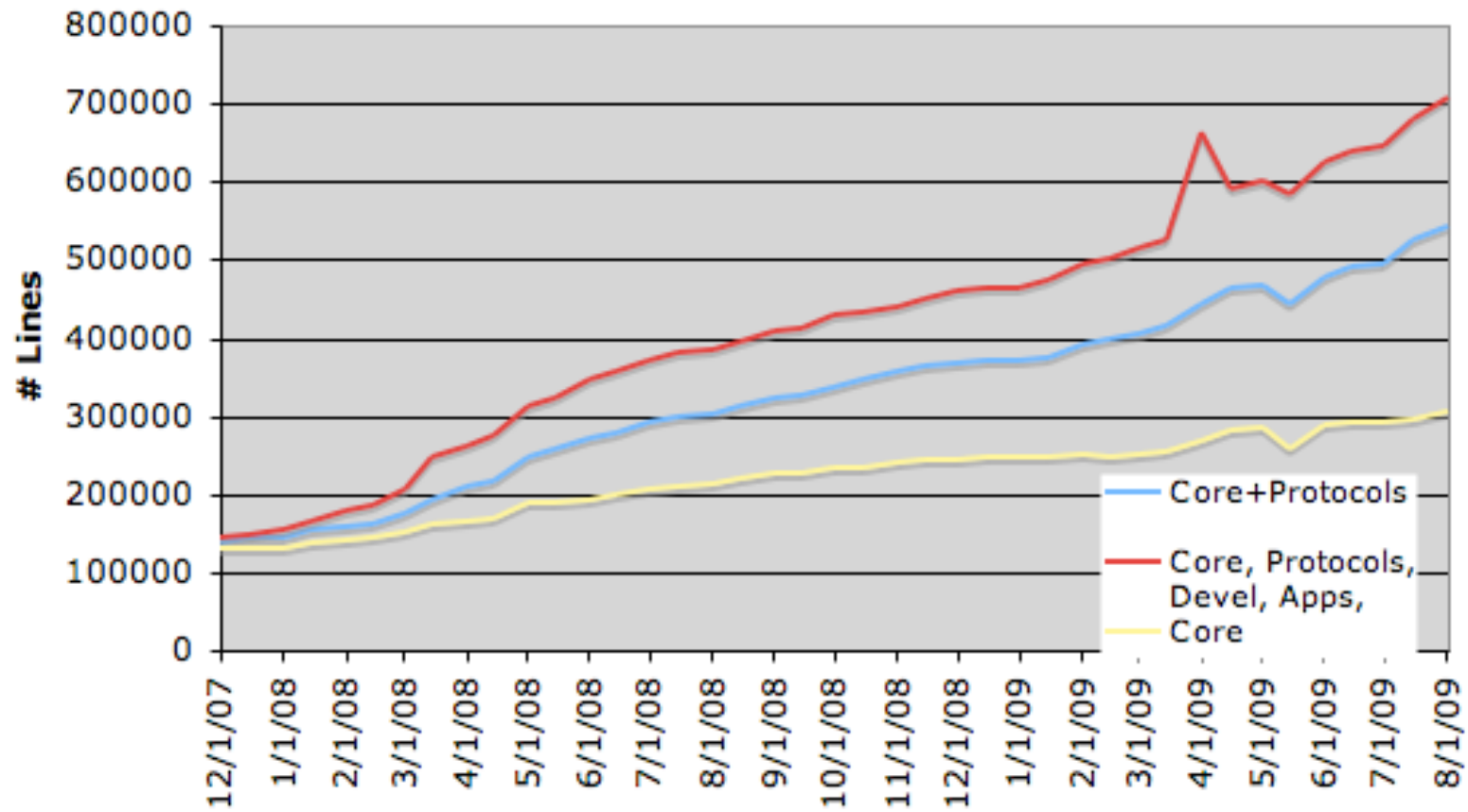
Mini's Growth



+110K lines
in utility,
numeric, &
ObjexxFCL

Andrew Leaver-Fay

Mini's Growth



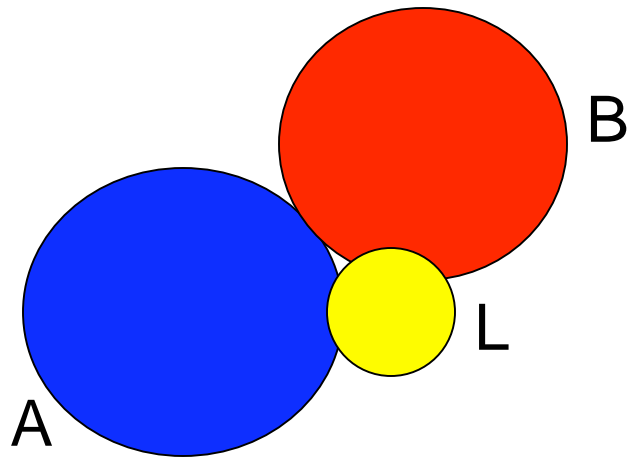
Outstanding Hurdles

- Missing R++ functionality
 - rtmin, wobble & chuck
 - Library subdivision
 - Allow EnergyMethods outside core/
 - Documentation
 - Selection Syntax
-
- Protocol Integration
 - Testing

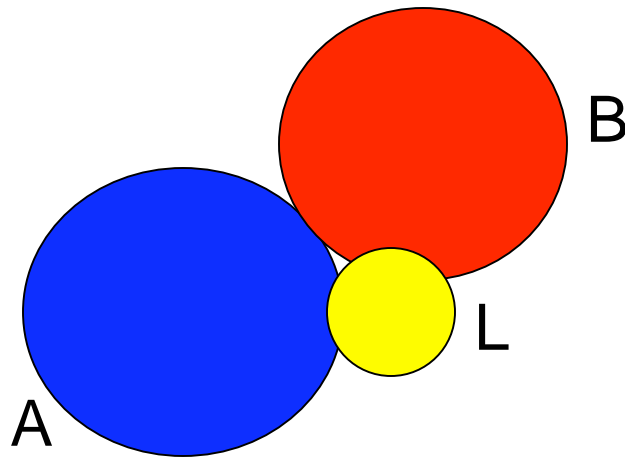
API Improvement

- Theorem:
The command line is a poor place to script
- Proof:
By Reductio ad absurdum.
Assume the command line is a good place to script,

Design a Protein/Protein Interface to bind a Ligand

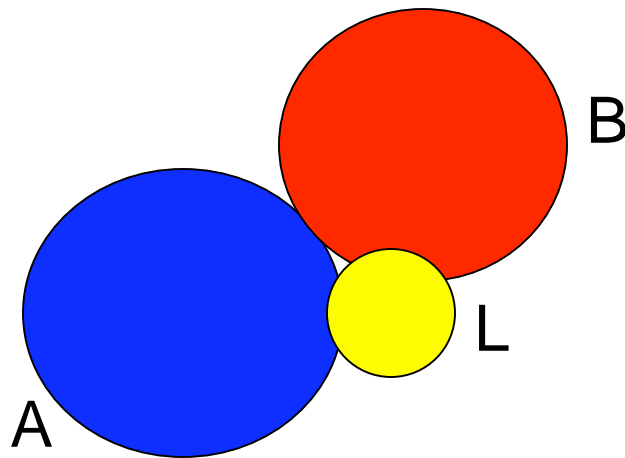


Design a Protein/Protein Interface to bind a Ligand



```
for i = 1:nstruct
  dock L to A
  for j = 1:10
    dock B to AL
    design ALB interface
    minimize ALB jumps + sc
```

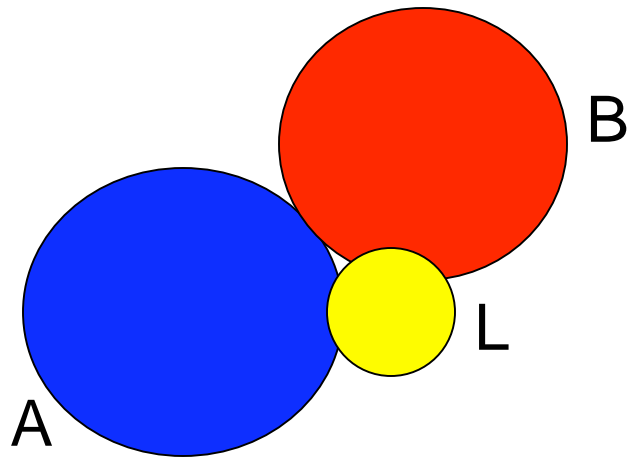
Design a Protein/Protein Interface to bind a Ligand



```
for i = 1:nstruct
  dock L to A
  for j = 1:10
    dock B to AL
    design ALB interface
    minimize ALB jumps + sc
```

Assumption: the command line is a good place to script.
Two different docking runs (AL, BAL) read the same flags
∴ The command line is a poor place to script

Design a Protein/Protein Interface to bind a Ligand



```
for i = 1:nstruct
  dock L to A
  for j = 1:10
    dock B to AL
    design ALB interface
    minimize ALB jumps + sc
```

Alt#1: B is fixed in sequence

Alt#3: L is designable

Alt#5: Favor native seq for A

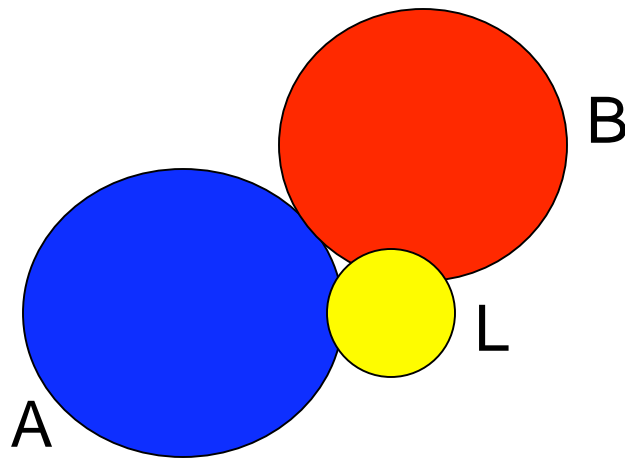
Alt#7: Filter each i, each j

Alt#2: A has flexible loops

Alt#4: A's loop lengths can vary

Alt#6: minimize N designs

Design a Protein/Protein Interface to bind a Ligand

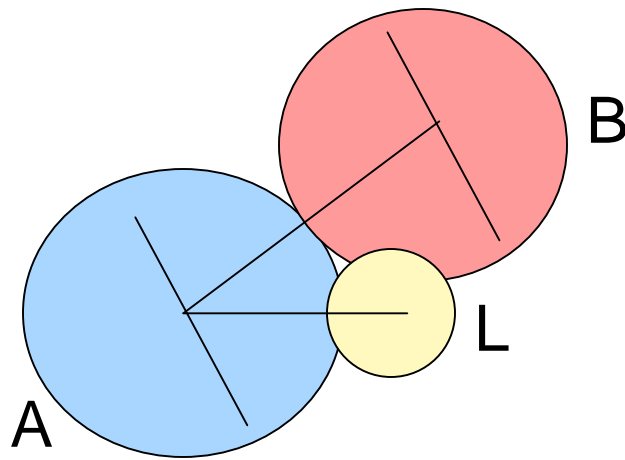


```
for i = 1:nstruct
  dock L to A
  for j = 1:10
    dock B to AL
    design ALB interface
    minimize ALB jumps + sc
```

Complications:

- Kinematic topology changes
- Sequence length might change
- Score function might change

Design a Protein/Protein Interface to bind a Ligand

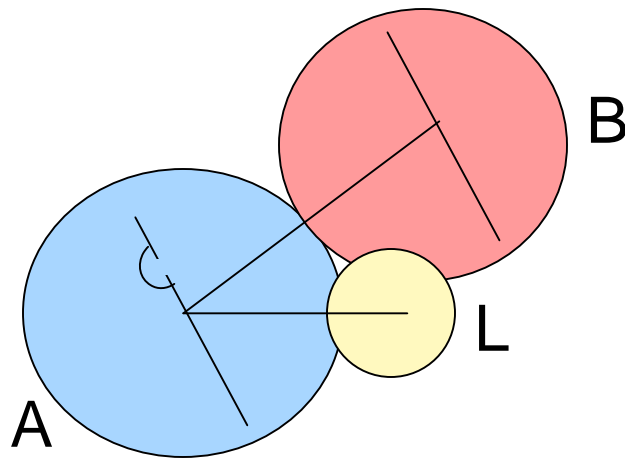


```
for i = 1:nstruct
  dock L to A
  for j = 1:10
    dock B to AL
    design ALB interface
    minimize ALB jumps + sc
```

Complications:

- Kinematic topology changes
- Sequence length might change
- Score function might change

Design a Protein/Protein Interface to bind a Ligand



```
for i = 1:nstruct
  dock L to A
  for j = 1:10
    dock B to AL
    design ALB interface
    minimize ALB jumps + sc
```

Complications:

- Kinematic topology changes
- Sequence length might change
- Score function might change

API Example: MatcherTask

- match.cc only reads the command line
- initialize_from_command_line()
 - Reads all matcher flags
 - Does not have to be called
 - Can be called and overwritten (not commutative)
- Every flag has a corresponding data member, accessor, and mutator
- Task is passed around to many different classes during Matcher initialization

Testing

- Unit Tests Rock
 - Will catch bugs integration tests miss
 - Guarantee code is working correctly
 - No wasted production runs with buggy code
 - Speed code development
 - Find bugs early
 - Pinpoints bugs!
- 180 core/ tests, 130 protocols/ tests, need 10x more

Testing

- Two challenges in writing unit tests
 1. Adding a new test to the test framework
 2. Deciding under what conditions your code is operating correctly

Testing

- Two challenges in writing unit tests
 1. Adding a new test to the test framework
 - Insultingly easy
 2. Deciding under what conditions your code is operating correctly
 - Hard, but essential

Testing

- E.g. test/protocols/match/BumpGrid.cxxtest.hh

```
class BumpGridTests : public CxxTest::TestSuite {
private:
    numeric::geometry::BoundingBox< core::Vector > bb_, bb2_;

public:
    void setUp() {
        /// next slide
    }

    void test_bool3D_or_overlap() {
        /// slide after next
    }
};
```



```
// Shared initialization goes here.  
void setUp() {  
    core_init();  
  
    lower_corner_ = core::Vector( 1.0, 1.0, 0.0 );  
    upper_corner_ = core::Vector( 10.0, 11.0, 12.0 );  
  
    bb_ = numeric::geometry::BoundingBox< core::Vector >( lower_corner_, upper_corner_ );  
  
    lower_corner2_ = core::Vector( 4.0, -1.0, 4.0 );  
    upper_corner2_ = core::Vector( 10.0, 15.0, 12.0 );  
  
    bb2_ = numeric::geometry::BoundingBox< core::Vector >( lower_corner2_, upper_corner2_ );  
}
```

```

void test_bool3D_or_overlap() {
    Bool3DGrid bool3d1, bool3d2;
    bool3d1.set_bounding_box( bb_ );
    bool3d2.set_bounding_box( bb2_ );

    TS_ASSERT( bool3d2.actual_bb().lower().x() == 4.0 );
    TS_ASSERT( bool3d2.actual_bb().lower().y() == -1.0 );
    TS_ASSERT( bool3d2.actual_bb().lower().z() == 4.0 );

    TS_ASSERT( bool3d2.actual_bb().upper().x() == 12.0 );
    TS_ASSERT( bool3d2.actual_bb().upper().y() == 15.0 );
    TS_ASSERT( bool3d2.actual_bb().upper().z() == 12.0 );

    core::Vector center( 7.5, 5.5, 8.5 );
    core::Vector contained( 8.5, 5.5, 8.5 );
    core::Vector edge_case( 9.5, 5.5, 8.5 );

    bool3d2.or_by_sphere_conservative( center, 2 );

    TS_ASSERT( bool3d2.occupied( center ) );
    TS_ASSERT( bool3d2.occupied( contained ) );
    TS_ASSERT( !bool3d2.occupied( edge_case ) );
    TS_ASSERT( !bool3d1.occupied( center ) );
    TS_ASSERT( !bool3d1.occupied( contained ) );
    TS_ASSERT( !bool3d1.occupied( edge_case ) );

    bool3d1.or_with( bool3d2 );
    TS_ASSERT( bool3d1.occupied( center ) );
    TS_ASSERT( bool3d1.occupied( contained ) );
    TS_ASSERT( !bool3d1.occupied( edge_case ) );
}

```

Testing

- Development workflow
 - Sketch out your classes
 - Starting with central classes
 - Start implementing class C
 - Write part of class C
 - Compile class C
 - Write a unit test, run, debug
 - Repeat
 - Run unit tests for all classes
- Development is faster & afterwards, tests persist!

Acknowledgements

- Sarel: Parser
- Andrew Ban: Observers
- Oliver Lange: TopologyBroker
- Ron Jacak, Sergey Lyskov: Unit Test Framework
- David Baker