Development of a Software Framework for Formalizing Forcefield Atom-Typing for Molecular Simulation

Janos Sallai, Tengyu Ma, Chris Iacovella, Christoph Klein

janos.sallai@vanderbilt.edu

WSSSPE4, Manchester, UK
Overview

Problem 1: specifying forcefield parameters in molecular simulations is a tedious, error prone task, particularly for large systems (lack of automation)

=> State-of-the-art software-engineering techniques to the rescue!
   - Model-integrated computing (MIC)
   - Domain Specific Languages (DSL)

Problem 2: forcefields are published in books, journal papers, websites of research groups, etc. (data management issue)

=> CI service does domain-specific forcefield verification, catalogs forcefields.
Background: forcefields

*Forcefields* describe interactions between particles in molecular simulations.

- Forcefield parameters depend not only on the particle type (e.g. Carbon or Oxygen atom)

<table>
<thead>
<tr>
<th>Interaction types</th>
<th>Force as a function of</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bonds</td>
<td>Bond distance</td>
</tr>
<tr>
<td>Angles</td>
<td>Angle defined by 3 particles</td>
</tr>
<tr>
<td>Torsion angles</td>
<td>Torsion angle defined by 4 particles</td>
</tr>
<tr>
<td>Non-bonded interactions</td>
<td>Distance of particles</td>
</tr>
</tbody>
</table>
**Background: forcefields**

*Forcefields* describe interactions between particles in molecular simulations

- Forcefield parameters depend not only on the particle type (e.g. Carbon or Oxygen atom)

- BUT also on the *chemical context* (e.g. a Carbon atoms at the end of an alkane chain different parameters from those in the backbone)
Background

Today, large molecular systems (inputs to simulators) are built from recurring blocks

- Blocks capture **structure** (particles, bonds, etc) and **forcefield** parameters
- Building a system for blocks is done in custom code (Python, C++, etc.)
Background

Today, large molecular systems (inputs to simulators) are built from recurring blocks

- Blocks capture structure (particles, bonds, etc) and forcefield parameters
- Building a system for blocks is done in custom code (Python, C++, etc.)

BUT

- **Stitching** blocks together is hard
  - There are bonds and other interactions across block boundaries
  - How these are handled depends on the type of the connecting block
- Blocks **cannot be reused** across projects
  - Even if structural building blocks are the same, forcefield used varies from project to project
Separation of structure and forcefield

mBuild creates complex molecular structures from reusable components

- Components are connected with *ports*
- Ports define geometric relations
- Supports *hierarchical composition*
- Provides *generative modeling* operations (tiling, space filling, grafting, masks)

How should we apply forcefield parameters?

https://github.com/iModels/mbuild
Applying forcefield to structures

We are building a tool to do this *automatically*.

**Foyer**: a tool to compute forcefield parameters of a structure

- Reads in structure and specification
- Outputs a structure with forcefield parameters that can be fed to a MD simulator

Foyer’s forcefield definition specifies *parameters and context* in which they apply

- Context is specified as *logic statement* over graph structures (rules)
- Set of rules can be statically checked to *verify* that
  - Forcefield parameters can be computed unambiguously
  - There are no contradictory rules

[https://github.com/iModels/foyer](https://github.com/iModels/foyer)
Forcefield specifications

A table with interaction types the corresponding parameters

<p>| | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>atom</td>
<td>791</td>
<td>13</td>
<td>CT</td>
<td>&quot;Perfluoroalkane CF3-&quot;</td>
<td>6</td>
<td>12.011</td>
<td>4</td>
</tr>
<tr>
<td>atom</td>
<td>792</td>
<td>13</td>
<td>CT</td>
<td>&quot;Perfluoroalkane -CF2-&quot;</td>
<td>6</td>
<td>12.011</td>
<td>4</td>
</tr>
<tr>
<td>vdw</td>
<td>791</td>
<td></td>
<td></td>
<td>3.5000</td>
<td>0.0660</td>
<td></td>
<td></td>
</tr>
<tr>
<td>vdw</td>
<td>792</td>
<td></td>
<td></td>
<td>3.5000</td>
<td>0.0660</td>
<td></td>
<td></td>
</tr>
<tr>
<td>charge</td>
<td>791</td>
<td></td>
<td></td>
<td>0.3600</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>charge</td>
<td>792</td>
<td></td>
<td></td>
<td>0.2400</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Where is it coming from?

- Published in books/journal/conference papers
- Some simulators include forcefield files (tend to be general-purpose)
- Research groups develop in-house forcefields that are designed for a particular class of chemical systems (e.g. proteins, nanolubrication, etc)

Documentation of applicability is often **lacking**.
Even if documentation exist, it is **not** **machine-readable**.
Forcefield specification

Foyer provides a formalism to describe applicability rules.

\[ C_{791} : \text{type=C} \land \text{count(bonded atoms(type=F))}=3 \land \text{count(bonded atoms(type=C))}=1 \]

\[ C_{792} : \text{type=C} \land \text{count(bonded atoms(type=F))}=2 \land \text{count(bonded atoms(type=C))}=2 \]

This has to be done by the developer of the forcefield.
Online forcefield repository

Goals: having an online forcefield repository that

- is searchable
- captures evolution of forcefields (version control)
- links back to external URLs/DOIs that use/reference a particular forcefield
- provides an API for tool integration

This goal is elusive. Hard to incentivize research groups to contribute.
Forcefield CI

Motivation:

Research groups store forcefield specifications in public repositories such as GitHub. GitHub’s webhooks allow for easy integration.

Validating an annotated forcefields requires test cases. A central database of structures and their known good parameterizations can be used to test multiple forcefields.

CI service can catalog the forcefields that are hosted on GitHub, can can create a searchable database.
Forcefield CI

[Diagram showing the Forcefield CI service with stages for verification of annotation logic and validation of annotated forcefield, connected to GitHub repository, Forcefield developer, Molecular libraries, and Forcefield users.]
Forcefield CI

We are building a Continuous Integration service that

- Integrates with GitHub
  - Webhooks on commits/pull request
- Verifies forcefield specs
  - Ensures unambiguity of rules
  - Reveals conflicting rules
- Validates forcefield specs
  - Runs Foyer on test cases: structures without parameterizations
  - Compares Foyer output with known parameterization
- Reports results
  - Forcefield CI web interface
  - Pass/Fail result to GitHub
Forcefield CI

Our team is building an online Continuous Integration service that

- Maintains a database of forcefield projects
  - Follows how the forcefield evolves
  - Maintains history
  - Provides an online, searchable interface
- Assigns permalinks to forcefield versions
  - Unique, stable URI (and URL) for each git commit that contains forcefield change
  - We consider assigning DOIs
- Maintains back references
  - Links from forcefield to projects where it is used
Status

**mBuild**: building complex molecular structures through component composition
- Production ready
- Available at [https://github.com/iModels/mbuild](https://github.com/iModels/mbuild)

**Foyer**: assigning forcefield parameters to untyped structures
- Prototype ready: [https://github.com/iModels/foyer](https://github.com/iModels/foyer)
- Small subset of OPLSaa forcefield works
- Declarative annotation syntax is being developed

**Forcefield CI**
- Web application, GitHub integration prototype is ready
- Verification, Validation, Forcefield catalogization is in planning phase
Questions

janos.sallai@vanderbilt.edu