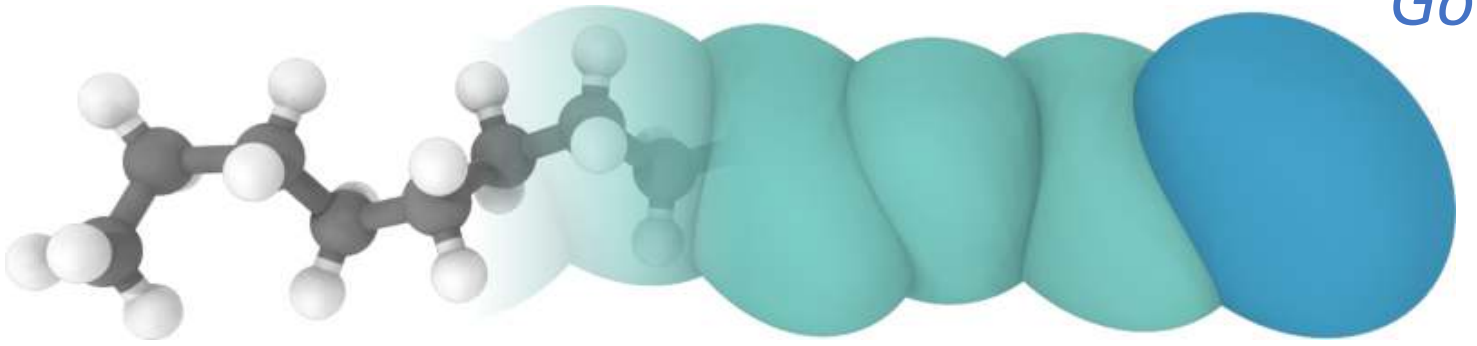
Managing the complexity of multiscale modelling with semantic technologies

Otello M. Roscioni

Goldbeck Consulting LTD



The OpenModel Project

www.open-model.eu

"Where have I seen this before?"

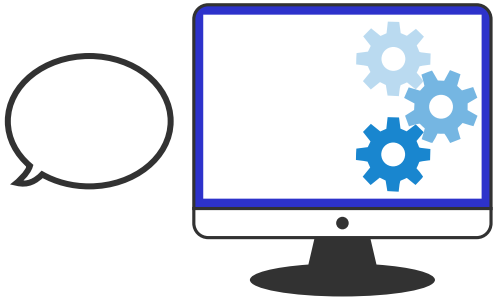
The OpenModel Project

www.open-model.eu

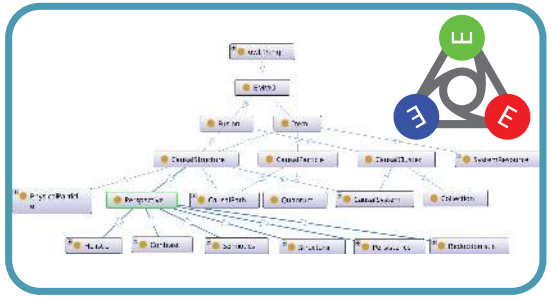
"Where have I seen this before?"

OpenModel is a **management system** for materials modelling workflows based on **semantic technologies**:

- Interoperability between physics-based models, solvers, third-party software, post-processors, and databases via a semantic layer.
- Automatic generation and execution of materials modelling workflows.
- FAIR data creation.



User front-end



Ontologies

AiiDA

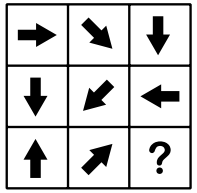
SimPhoNy

DLite

Workflows and data management

DB

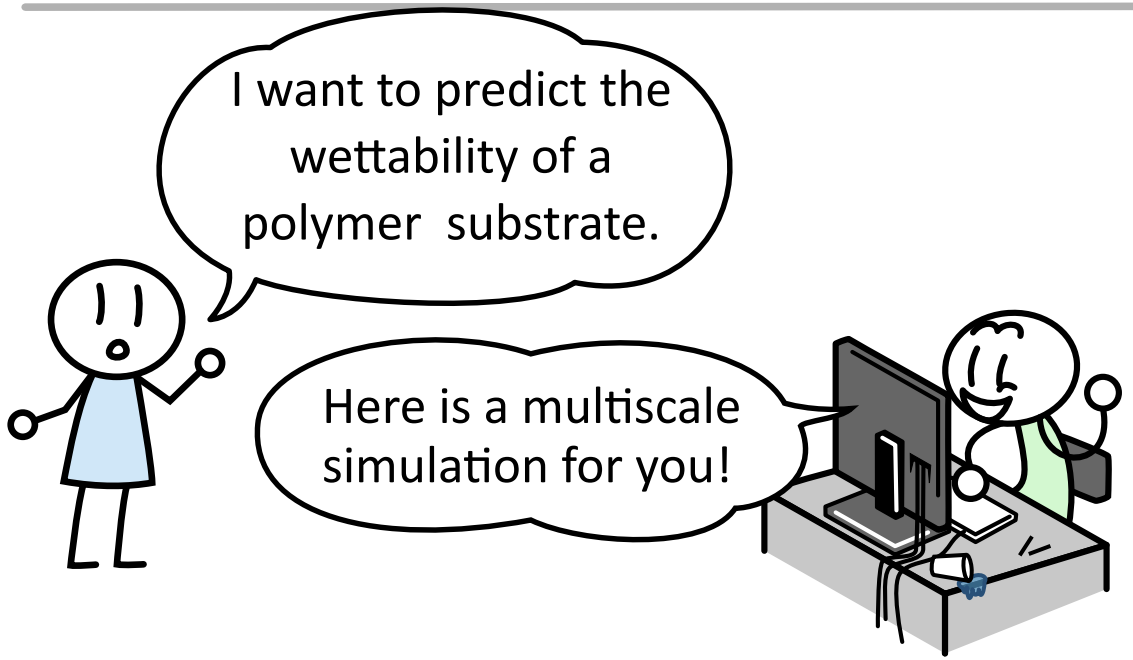
Software



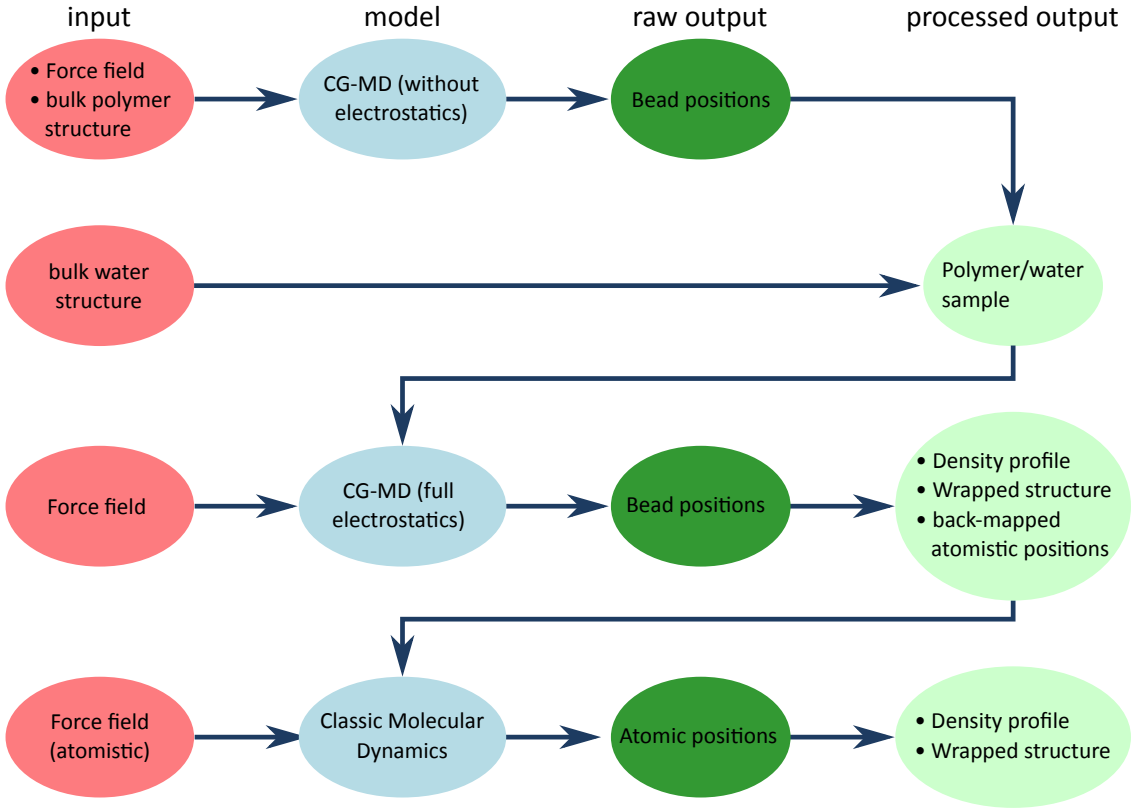
Reasoning

V&V Services

The use case: a hydrophilic polymer membrane



MODA workflow

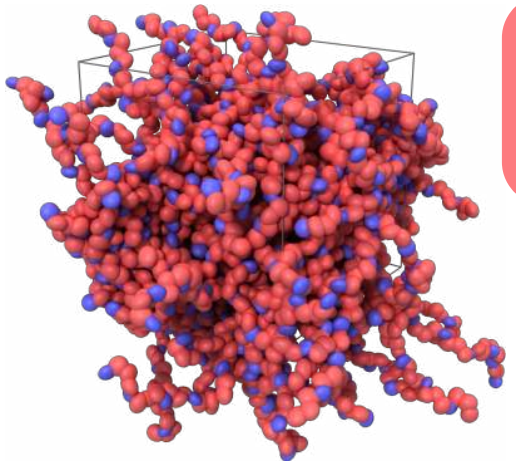


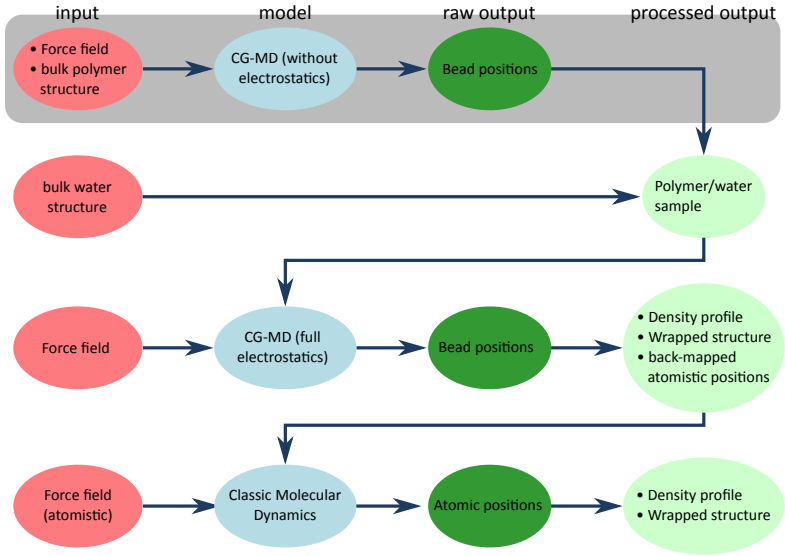
MODA is a human-friendly, high-level description of material models and workflows. **However**:

- ✗ The semantic of the arrows is undefined.
- ✗ The notation does not distinguish data from processes.
- ✗ Not machine executable.

https://github.com/hothello/openmodel

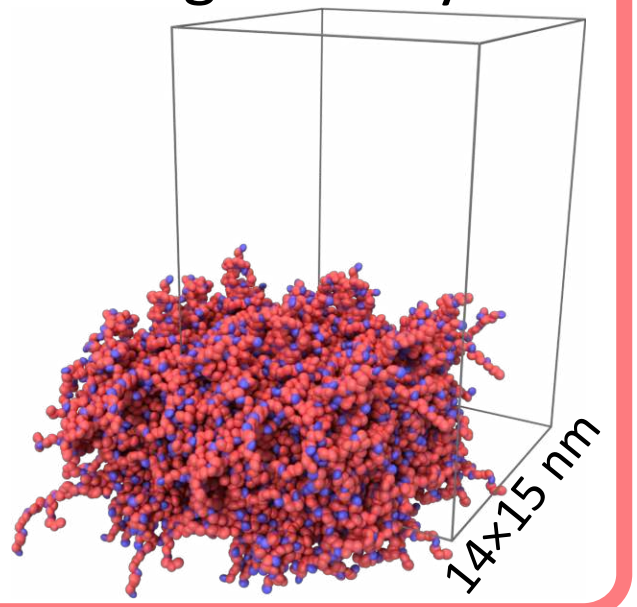
Step 1: create the slab





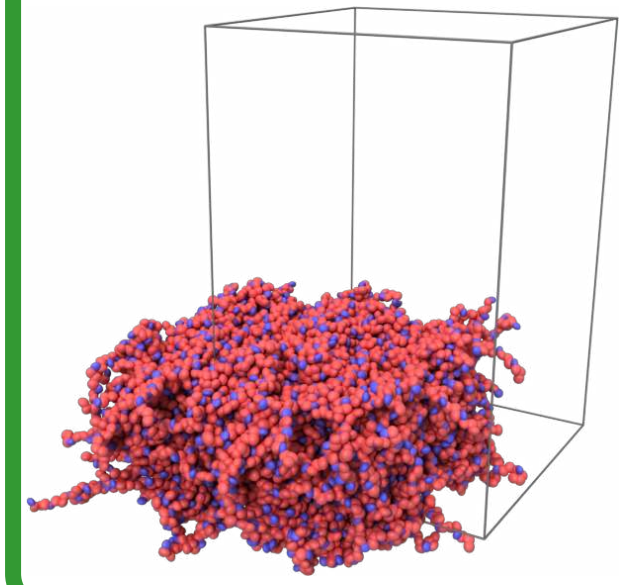
Materials: PLGA bulk structure, CG force field.

2×2×1 supercell, slab geometry.

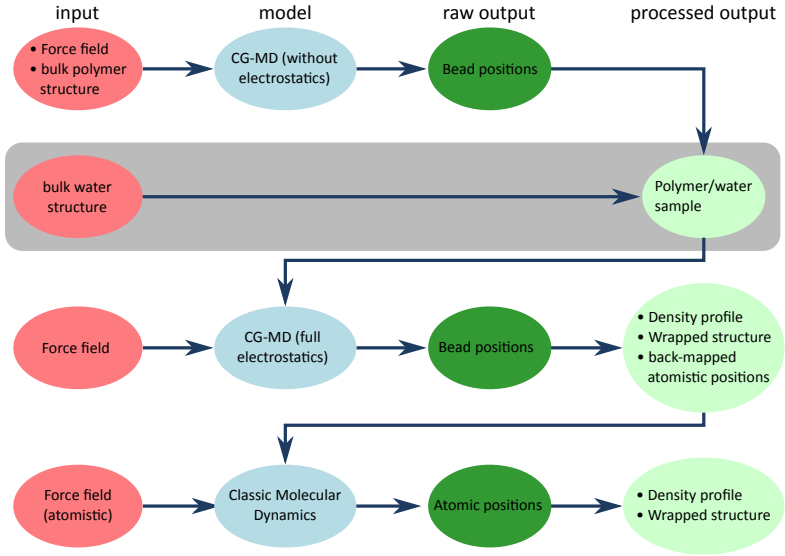


CG-MD

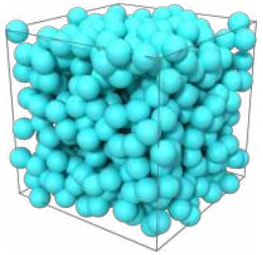
relaxed structure



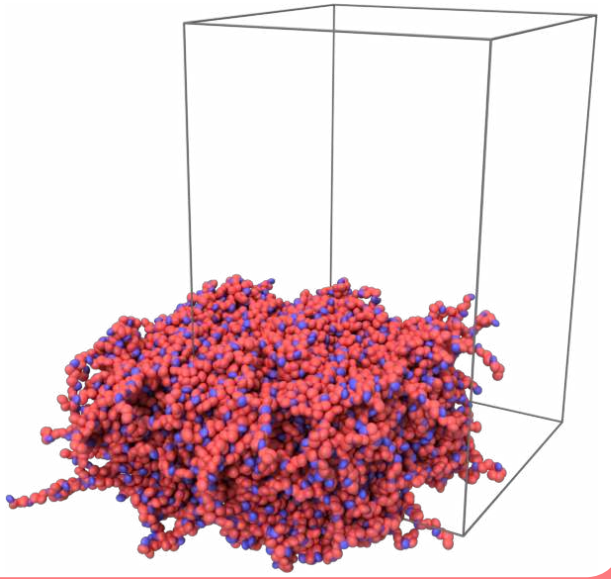
Step 2: add the solvent



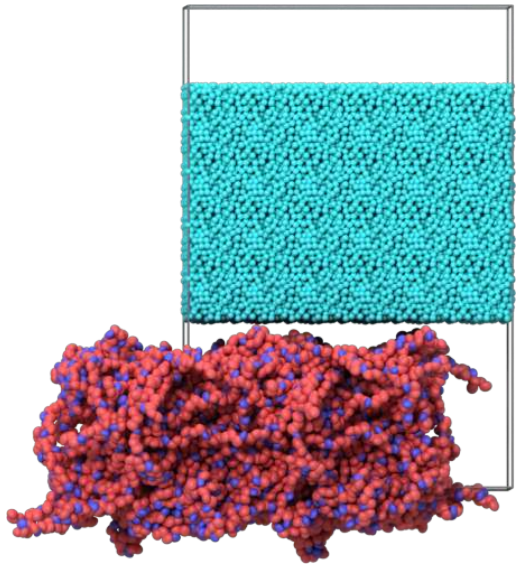
Materials: water



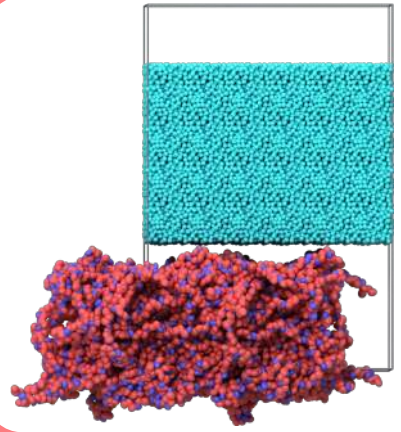
PLGA slab



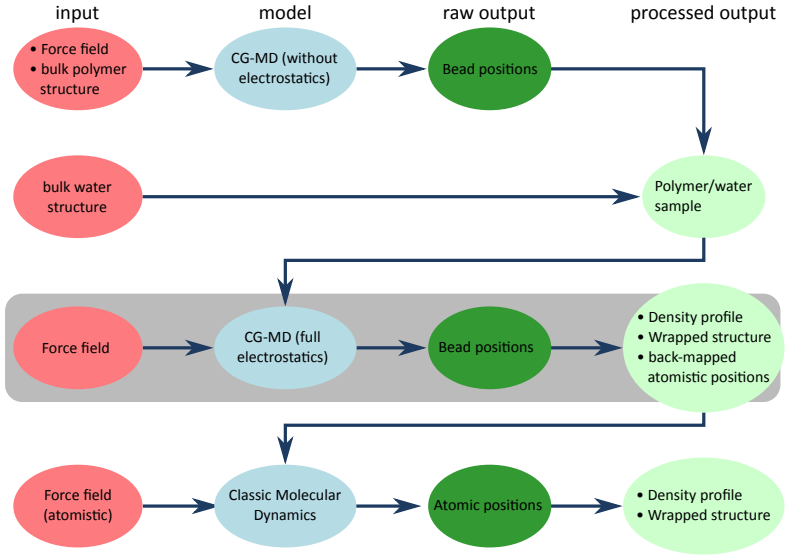
initial structure



Step 3: equilibrate again

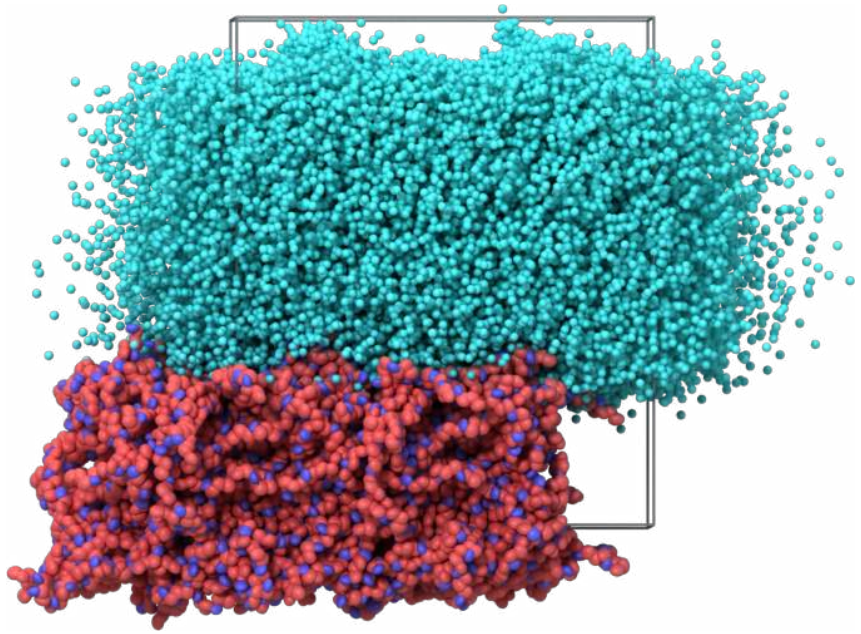


Input from previous step, CG force field.

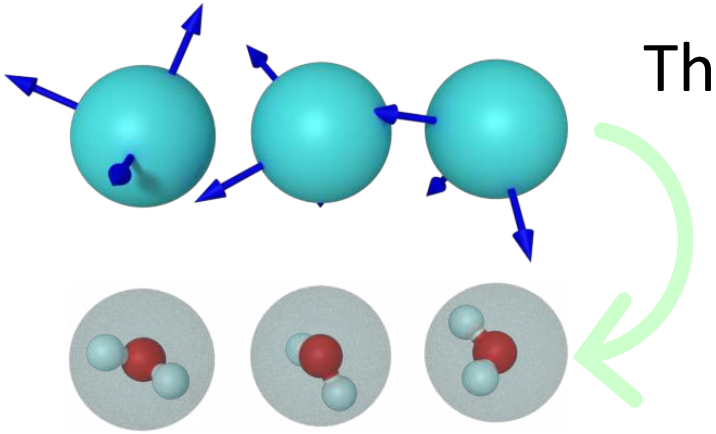


CG-MD

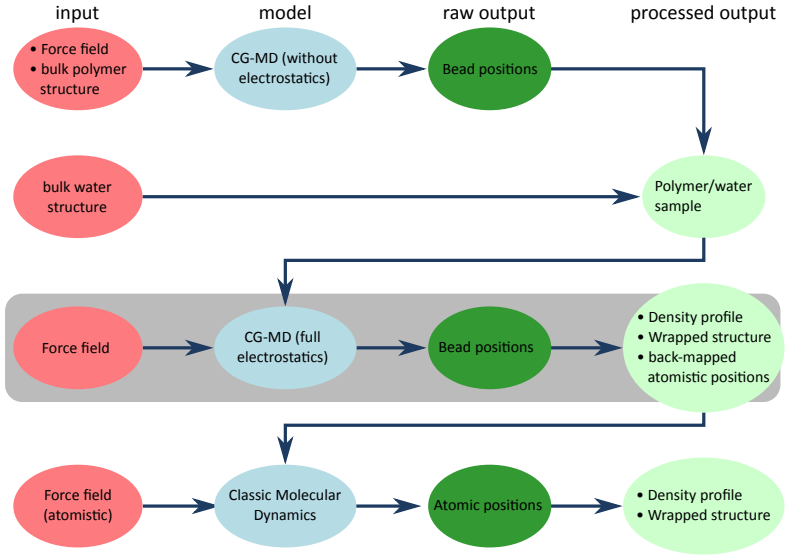
relaxed structure



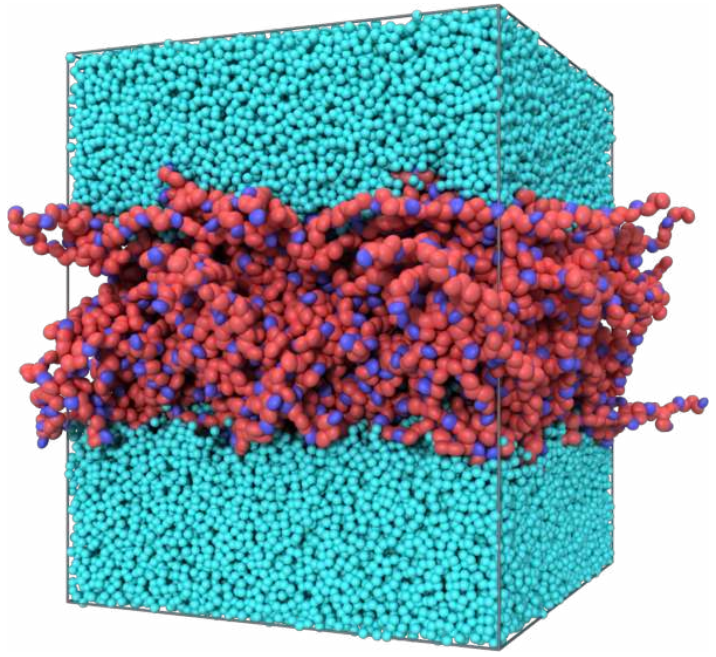
Step 3: post-processing



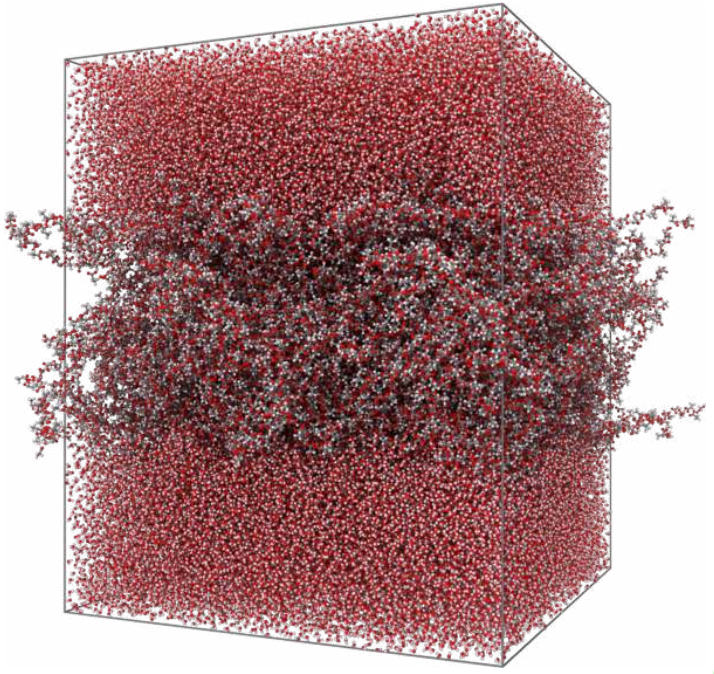
The beads are finite-size spheroids!



wrapped CG structure

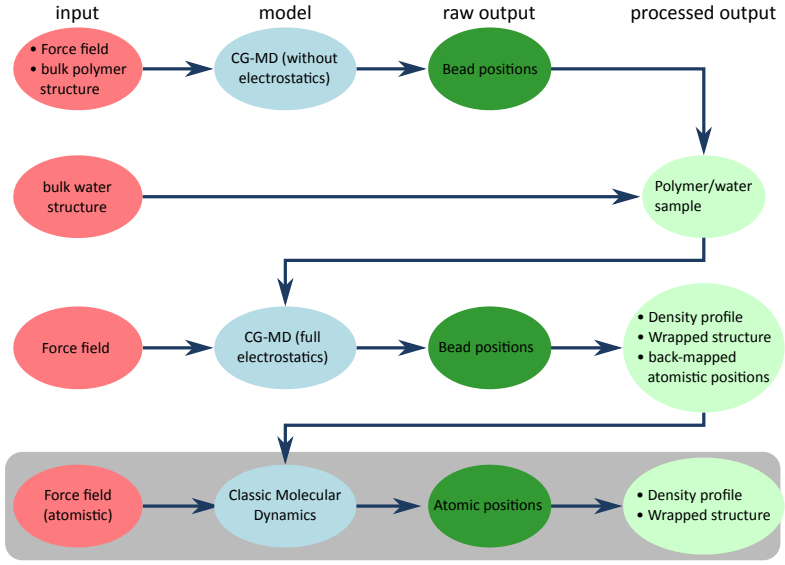


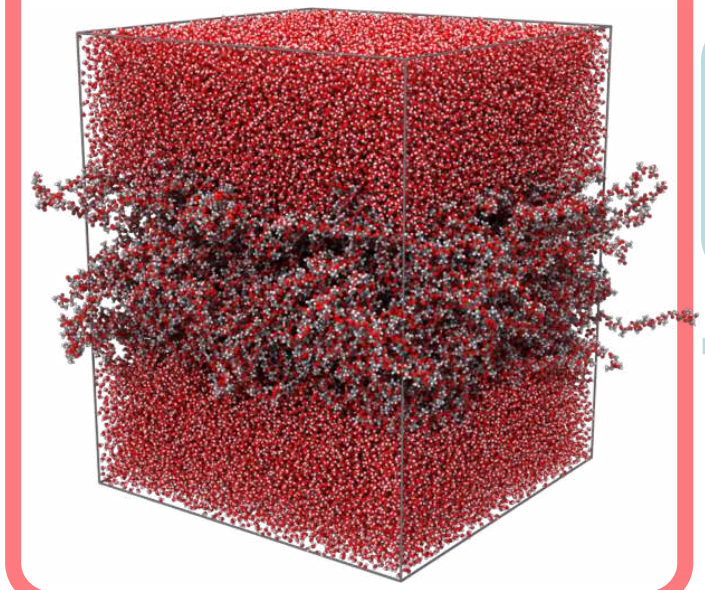
back-mapped structure



Step 4: equilibrate

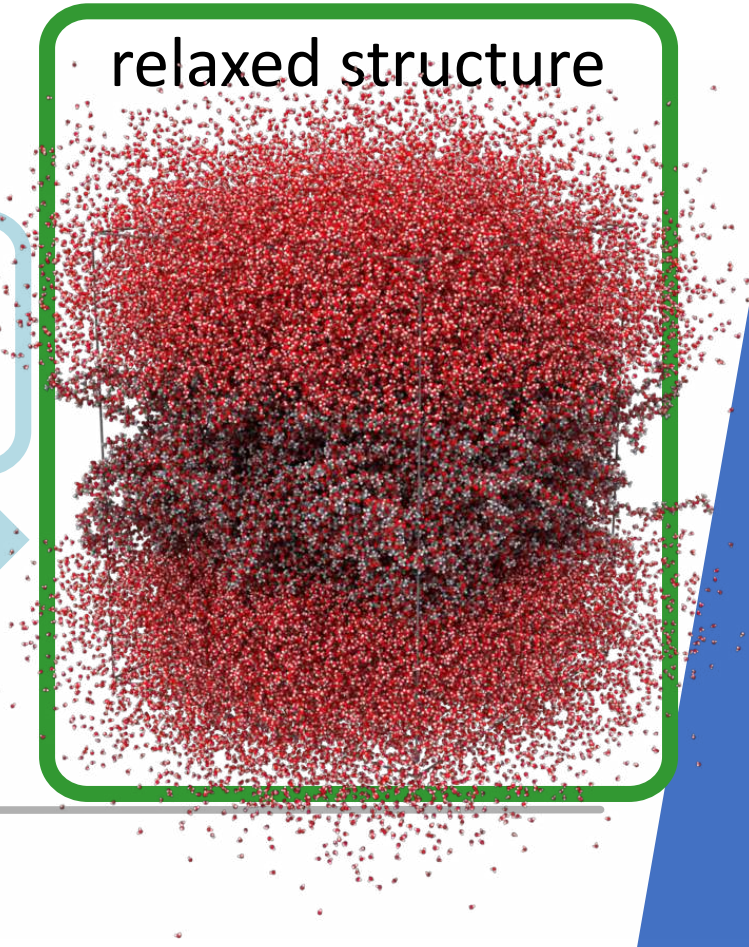
Input from previous step, atomistic force field.





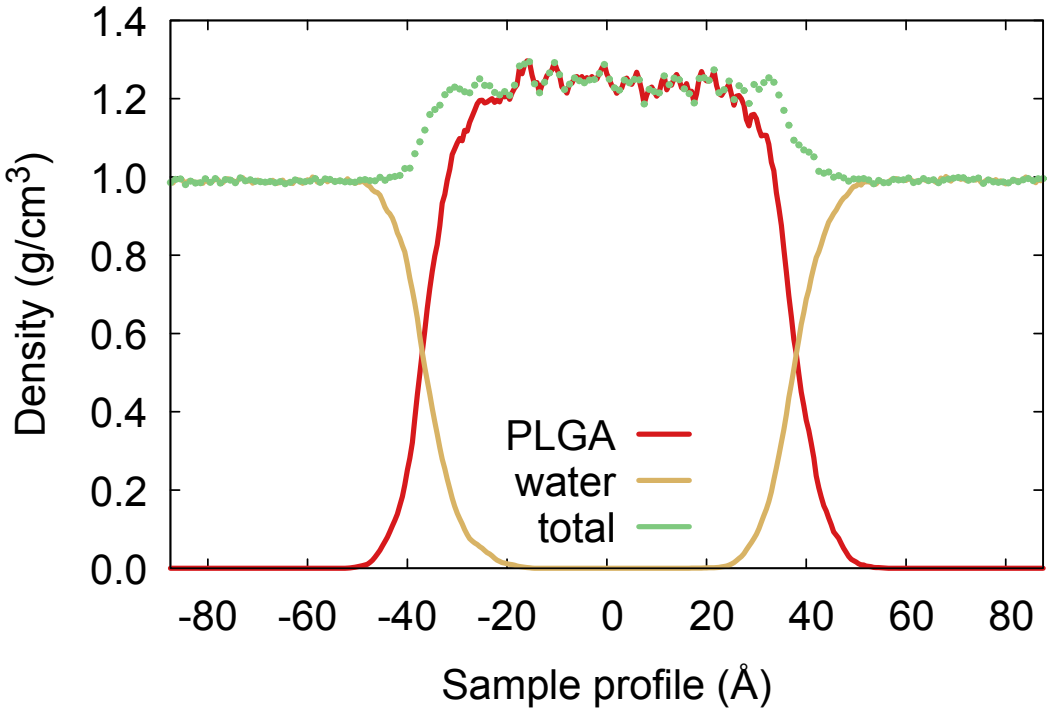
AA-MD

relaxed structure

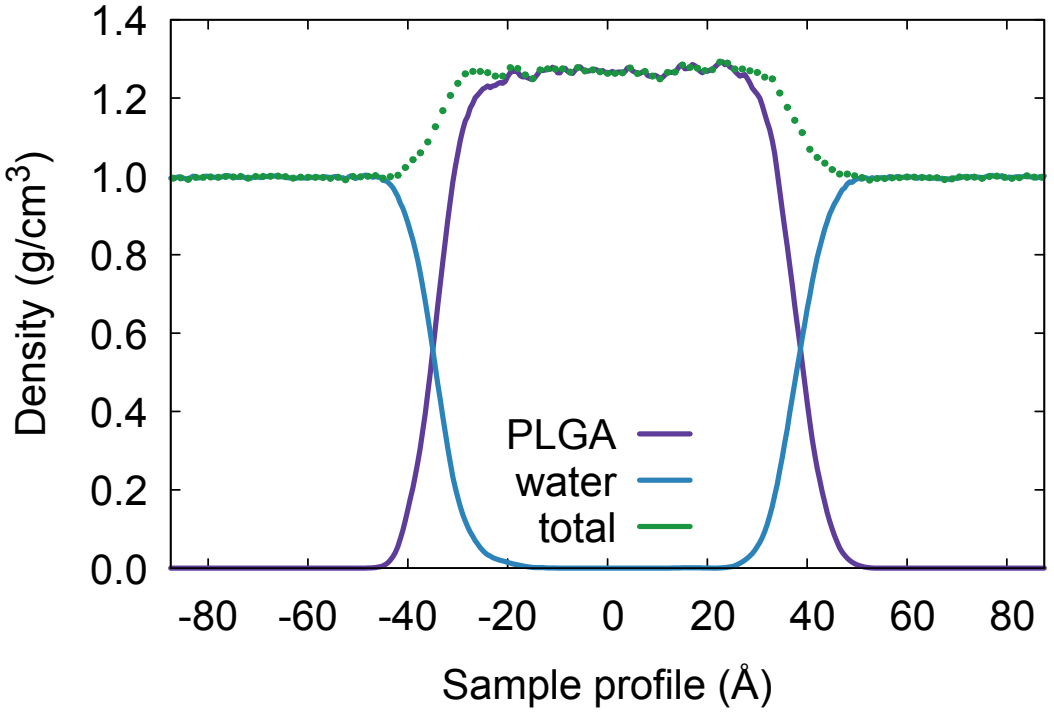


Step 4: post-processing

Density profiles



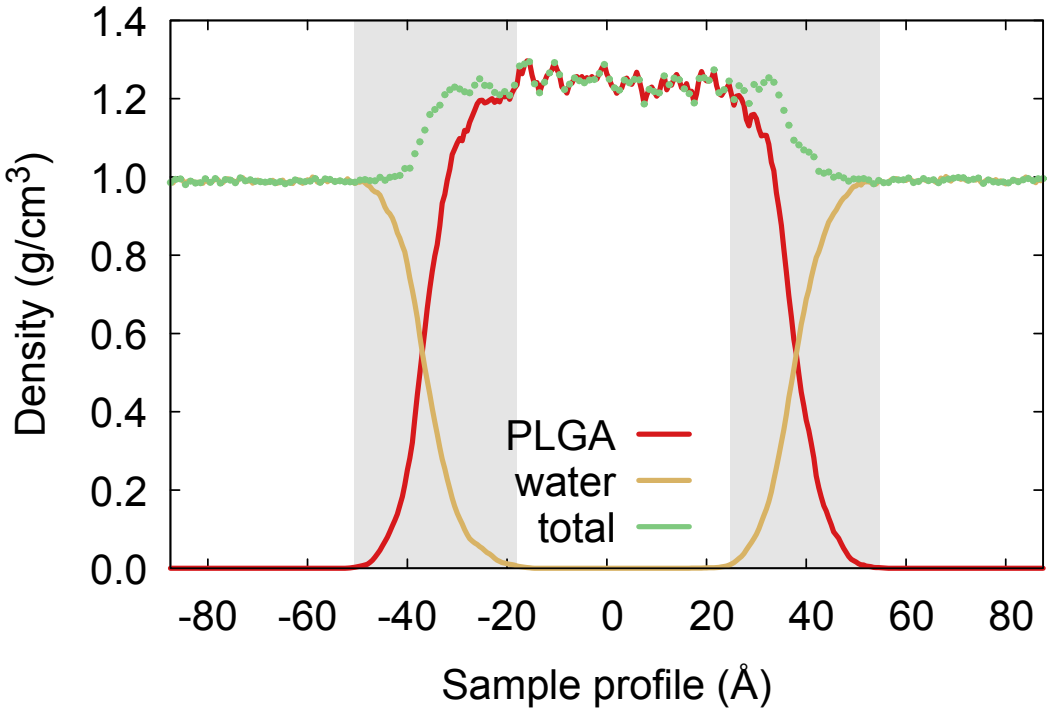
coarse-grained

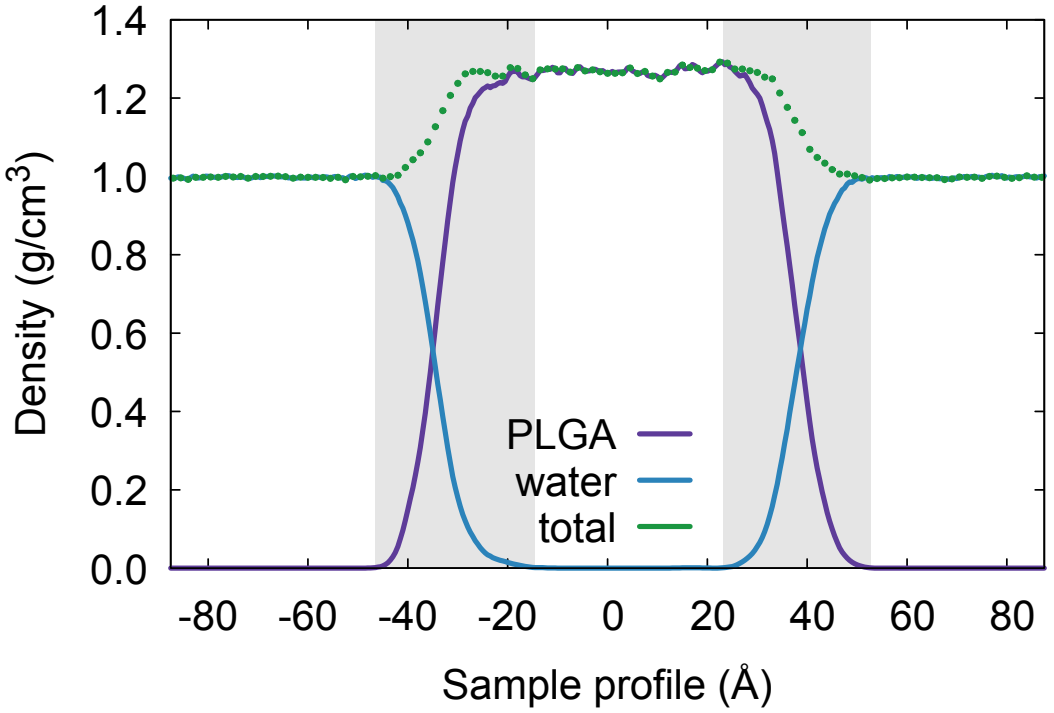


atomistic

Step 4: post-processing

Density profiles

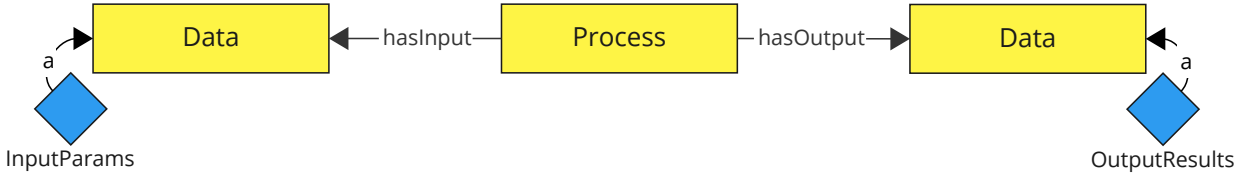


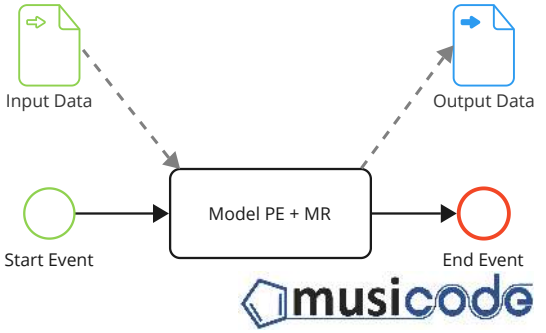


A soft interface (3 nm) is present on both samples.

Semantic documentation of workflows

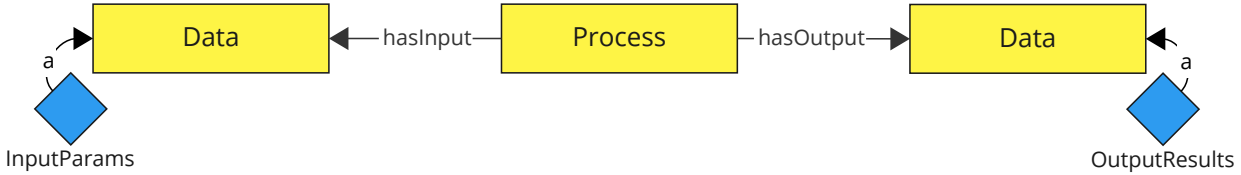
- The topology of a workflow is build using **hasInput** and **hasOutput** relations connecting processes to datasets.

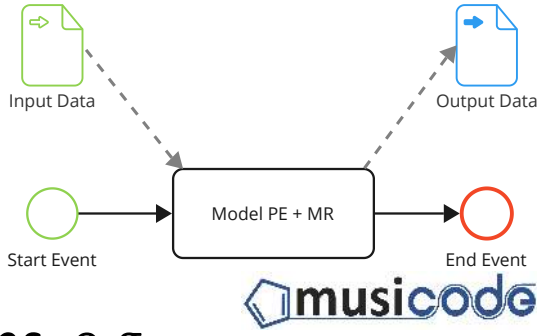




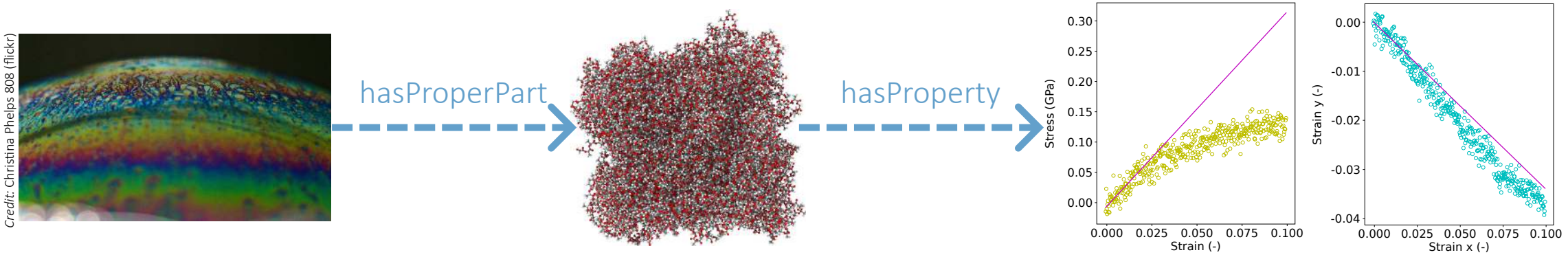
Semantic documentation of workflows

- The topology of a workflow is build using **hasInput** and **hasOutput** relations connecting processes to datasets.





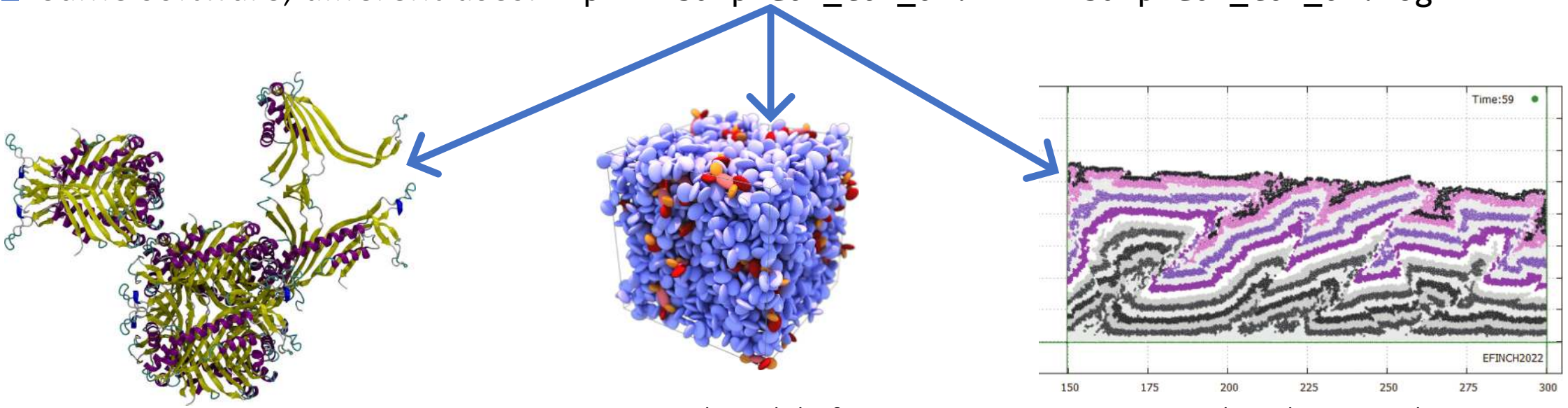
- The input and output **datasets** are "decorated" with **semantic relations**, e.g.



Polymer membrane PLGA Elastic modulus

Breaking workflows into single tasks

- Each workflow is broken down into a series of individual tasks.
- Each task is a computational **process** transforming input data into output data.
- Each process has a **limited scope** and is described with a **wrapper**.
- Same software, different uses: `lmp -i sample01_t02_02.in -l sample01_t02_02.log`



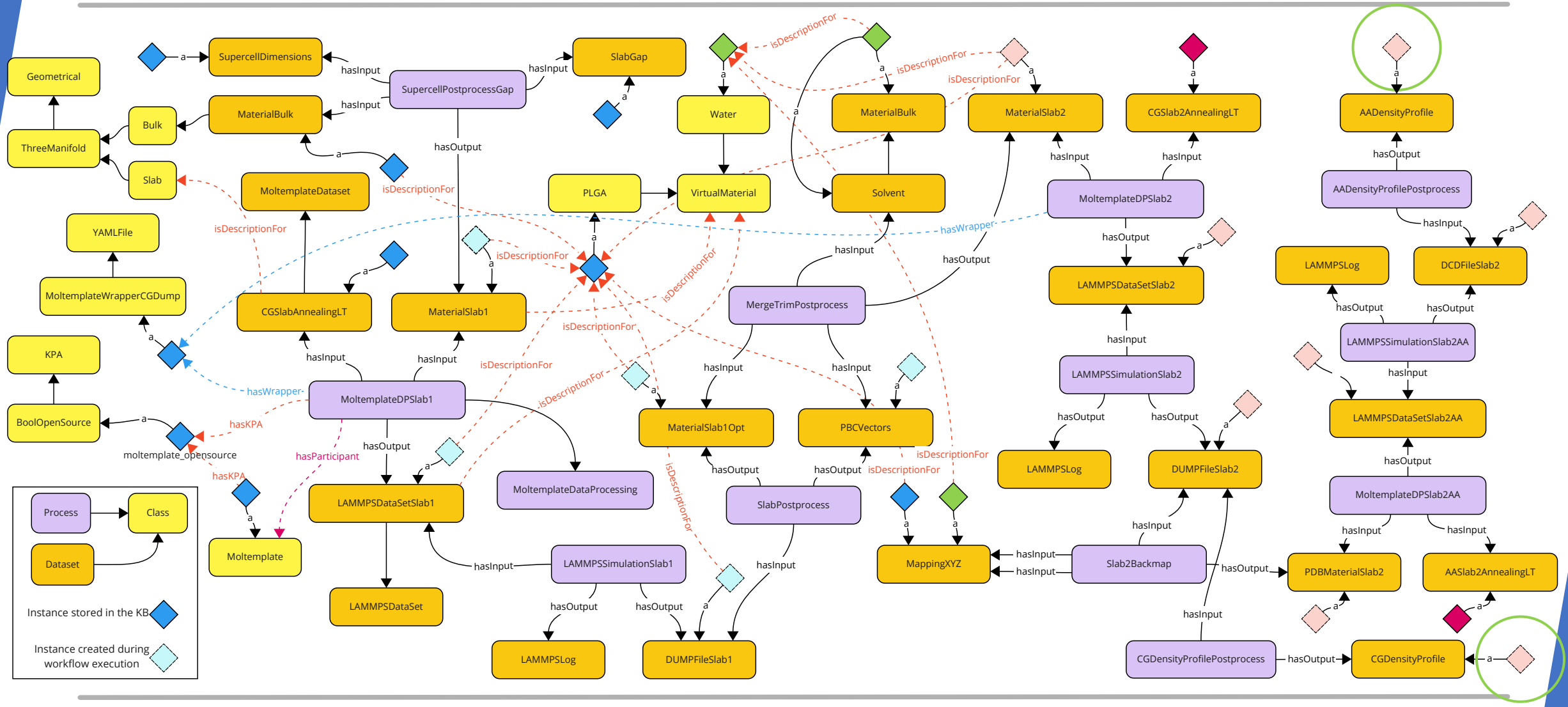
7PWN protein, 22K atoms

Coarse-grained model of organic semiconductors (MOLC model)

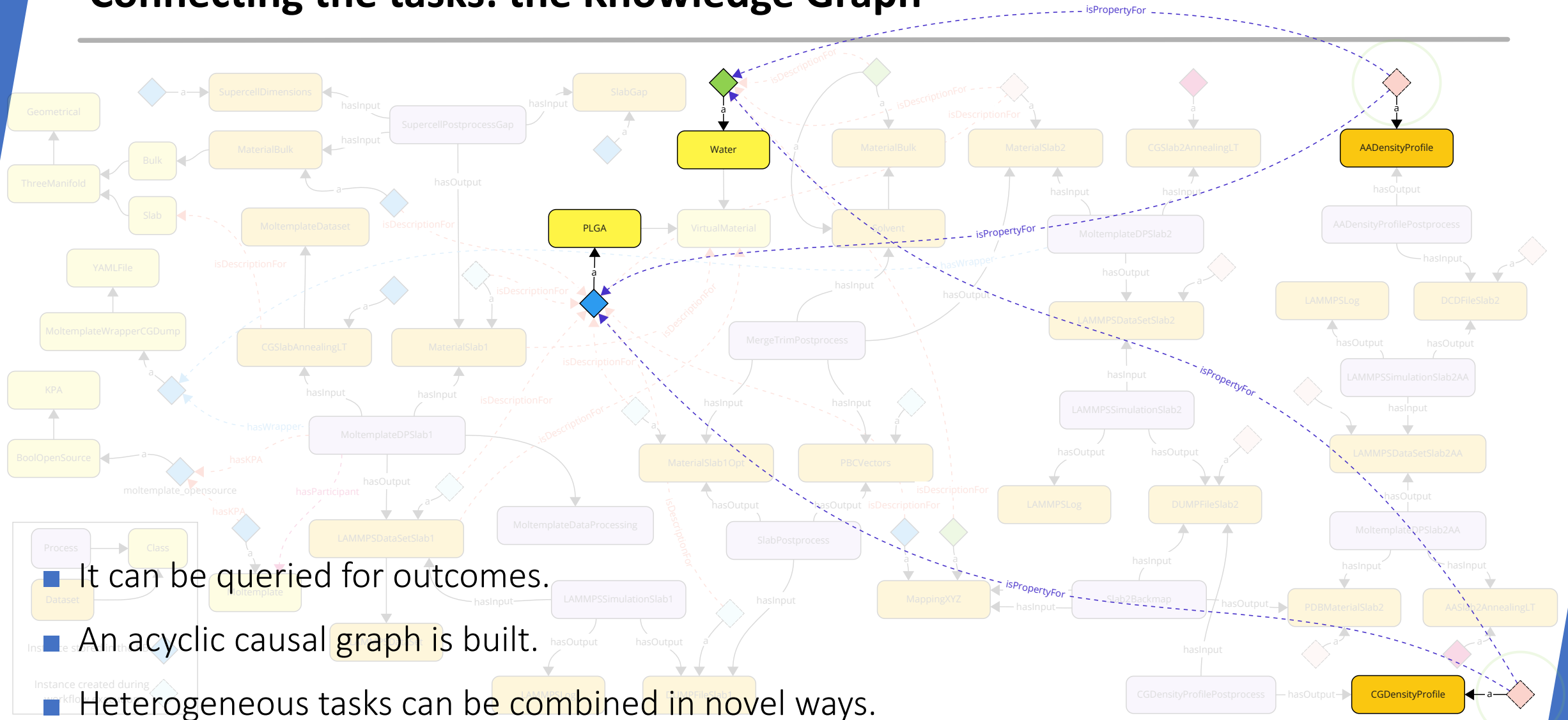
Structural geology simulations

Credit: Emma Finch, U. Manchester

Connecting the tasks: the Knowledge Graph



Connecting the tasks: the Knowledge Graph



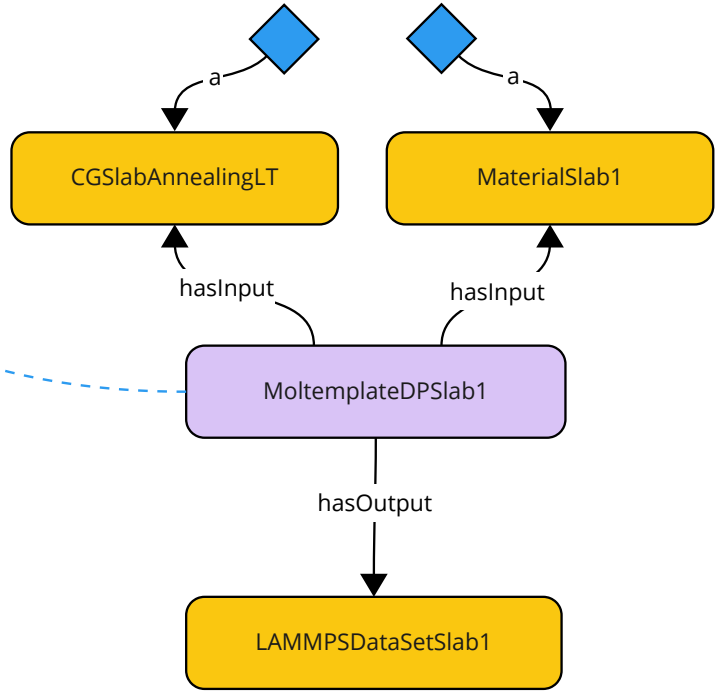
- It can be queried for outcomes.
- An acyclic causal graph is built.
- Heterogeneous tasks can be combined in novel ways.

Executing tasks with wrappers

Suppose we want to describe a simulation that is executed from the CLI:

```
$ moltemplate.sh -atomstyle "hybrid molecular ellipsoid" -overlay-bonds -dump input.dump input.lt
```

```
- workflow: execflow.exec_wrapper
    inputs:
      shelljob:
       metadata:
         options:
           resources:
             num_machines: "{{ ctx.mt.mn_machines }}"
             num_mpiprocs_per_machine: "{{ ctx.mt.mn_mpiprocs }}"
      command: "moltemplate.sh"
      arguments:
        - "-atomstyle"
        - "{{ ctx.mt.atomstyle }}"
        - "{{ ctx.mt.overlay }}"
        - "-dump"
        - "{{ ctx.mt.dump }}"
        - "{{ ctx.mt.ltinput }}"
      outputs:
        - input.in
    postprocess:
      - "{{ ctx.current.outputs['remote_folder']|to_ctx('lammps_dir') }}"
      - "{{ ctx.current.outputs['input_in']|to_ctx('lammps_in') }}"
```

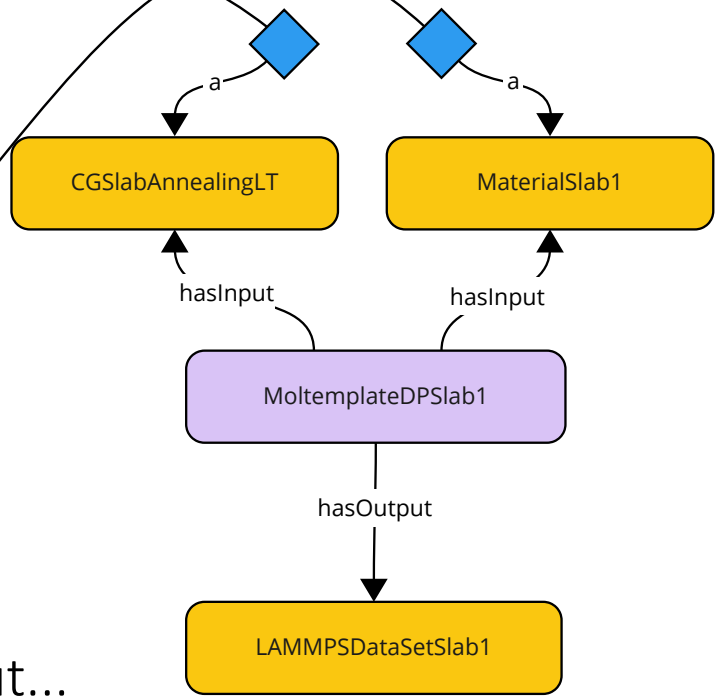


Documenting the datasets

The input can be stored as a single dataset, representing a particular instance of a simulation and its parameters.

```
---
egbRESXwrn:
 meta: http://open-model.eu/meta/1.0/CGSlabAnnealingLT
  dimensions: {}
  properties:
     atomstyle: "hybrid molecular ellipsoid"
     overlay: "-overlay-bonds"
     ltinput: /tmp/Inputs/02_cg_slab_01.lt
     mn_machines: 1
     mn_mpiprocs: 1
...
```

```
---
VHNl24cU4M:
 meta: http://open-model.eu/meta/1.0/MaterialSlab1
  dimensions: {}
  properties:
     dump: /tmp/Inputs/01_slab.dump
...
```

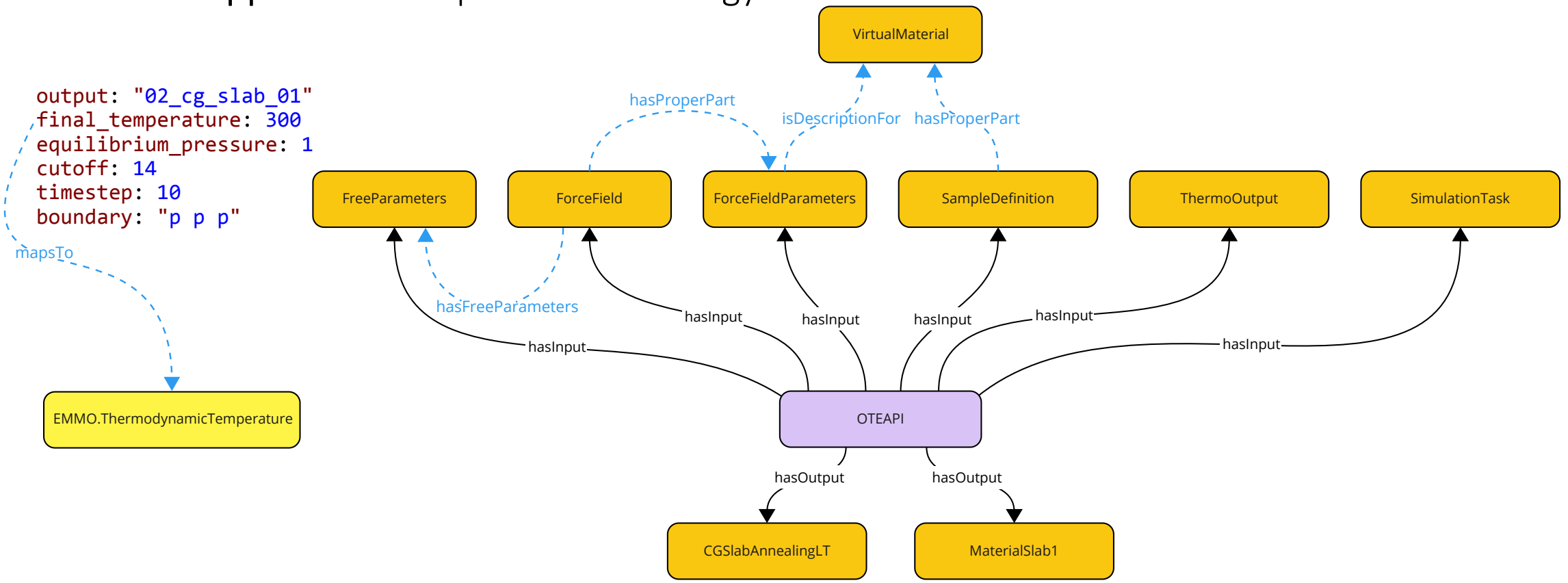


It ensures **reproducibility**, but...

It does not explain **how** the simulation is carried out.

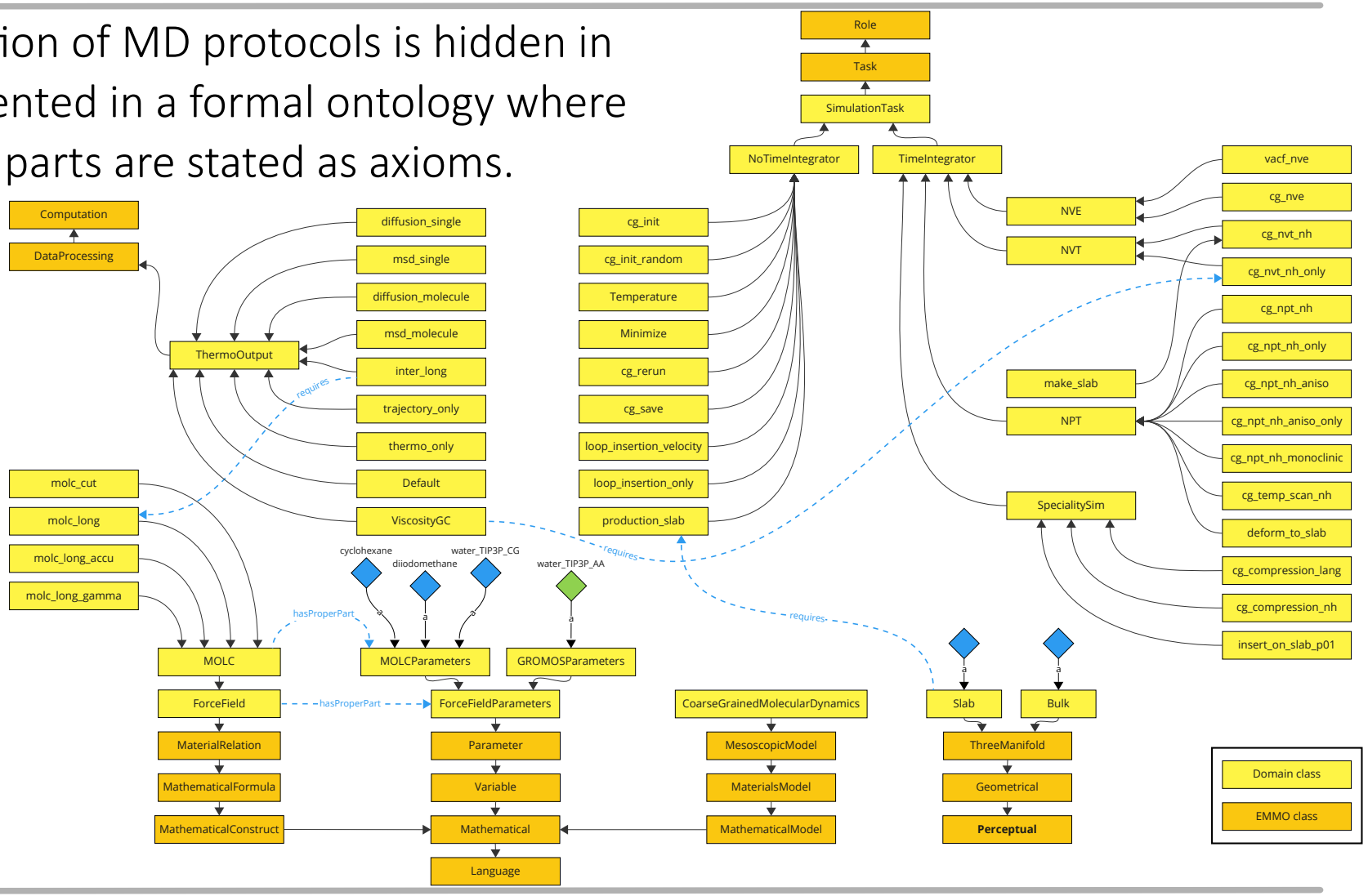
Documenting the datasets

OTEAPI pipelines reveal the content of the input datasets, exposing free and model parameters, materials relations, physical equations, and their **semantic relations**.
Data are **mapped** to concepts in the ontology.



Documenting the datasets: MD simulations

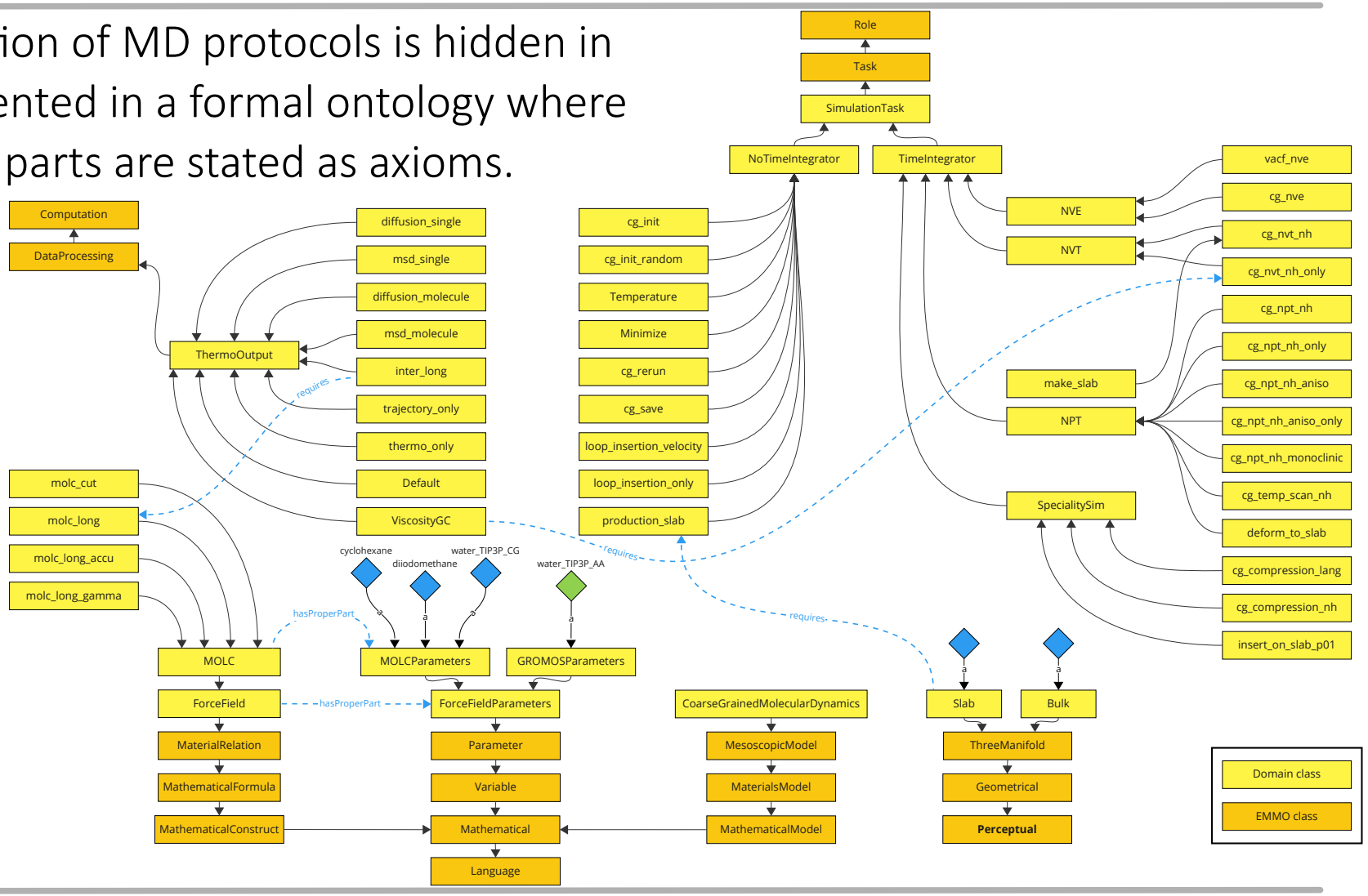
The syntactic implementation of MD protocols is hidden in macros, which are documented in a formal ontology where the relationships between parts are stated as axioms.



Documenting the datasets: MD simulations

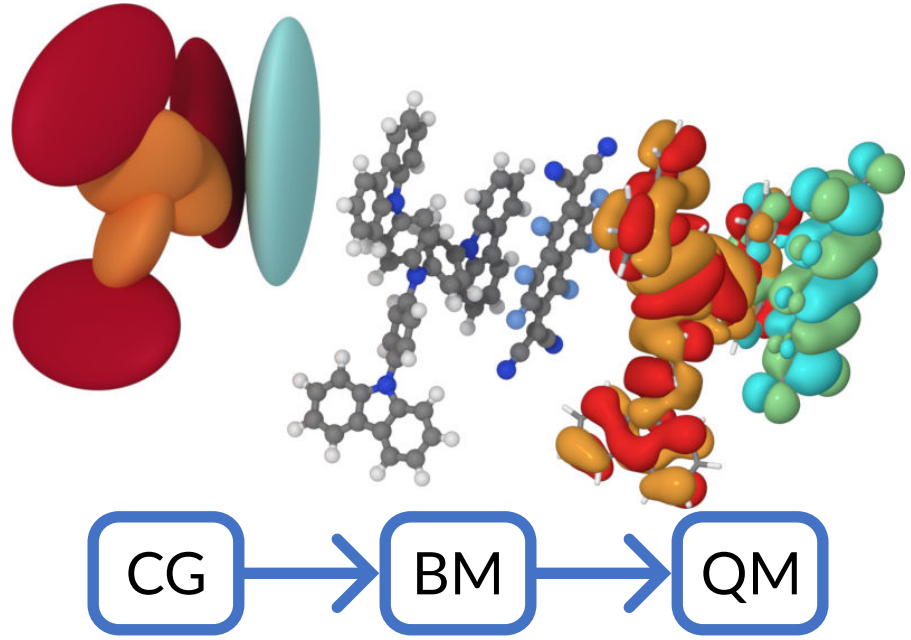
The syntactic implementation of MD protocols is hidden in macros, which are documented in a formal ontology where the relationships between parts are stated as axioms.

```
# Input variables (task dependant).
write_once("In Init"){

 # Input variables.
 variable run    string 02_cg_slab_01
 variable ts     equal  20
 variable tf     equal  300.
 variable p      equal  1.
 variable cutoff equal  14.
 variable skin   equal  4.
 variable s      equal  5
 variable cl     equal  200
 variable prod   equal  50000

 # PBC
 boundary p p p
}

# Define the force field.
import /usr/local/share/lt/cg_molc.lt
ff = new molc_long

# Single molecule definition of a PLGA chain.
import ../molc/plga/plga_cg_012.lt

# Create the molecules.
mol0 = new PLGA12[360]

# Quick annealing at fixed volume.
import /usr/local/share/lt/cg_tasks.lt
thermo = default
task[1] = new cg_nvt_nh
```

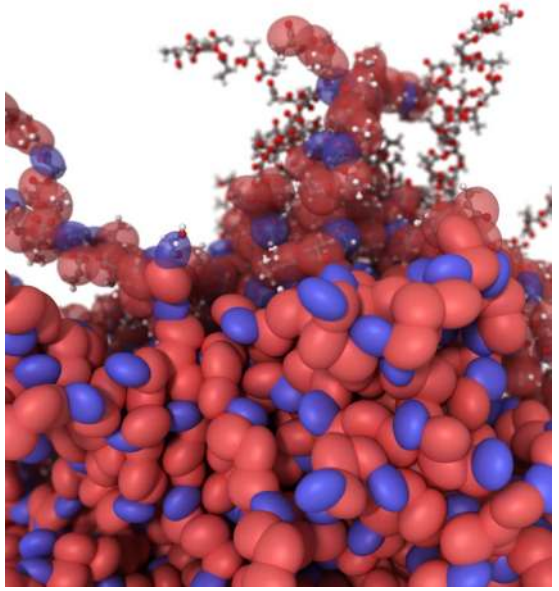


The science behind

Mesoscopic simulations executed with the **MOLC model**: a coarse-grained force field based on **ellipsoids**, long-range **electrostatics**, and **directional bonds**.



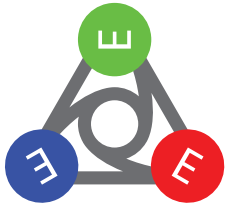
Are Coarse-Grained Structures as Good as Atomistic Ones for Calculating the Electronic Properties of Organic Semiconductors?
Roscioni et al. *J. Phys. Chem. C*, **127**, 9225 (2023).



Mesoscopic Modeling of Bio-Compatible PLGA Polymers with Coarse-Grained Molecular Dynamics Simulations
Bellussi et al. *J. Phys. Chem. B*, accepted (2025).

Take-home messages

- Semantic documentation of workflows allows users to ask **high-level queries** and obtain **machine-executable** code.
- Semantic interoperability allows to combine **heterogeneous tasks** in novel ways, retrieving existing workflows and creating new ones.
- A **formal description** of input parameters, materials relations, and numerical approximations ensures **reproducible** results and **physical consistency**.
- A syntactically valid simulation can yield bogus results, with no semantic understanding.
- Part of the **EMMO** ecosystem: https://emmo-repo.github.io/



Acknowledgements

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 953167.