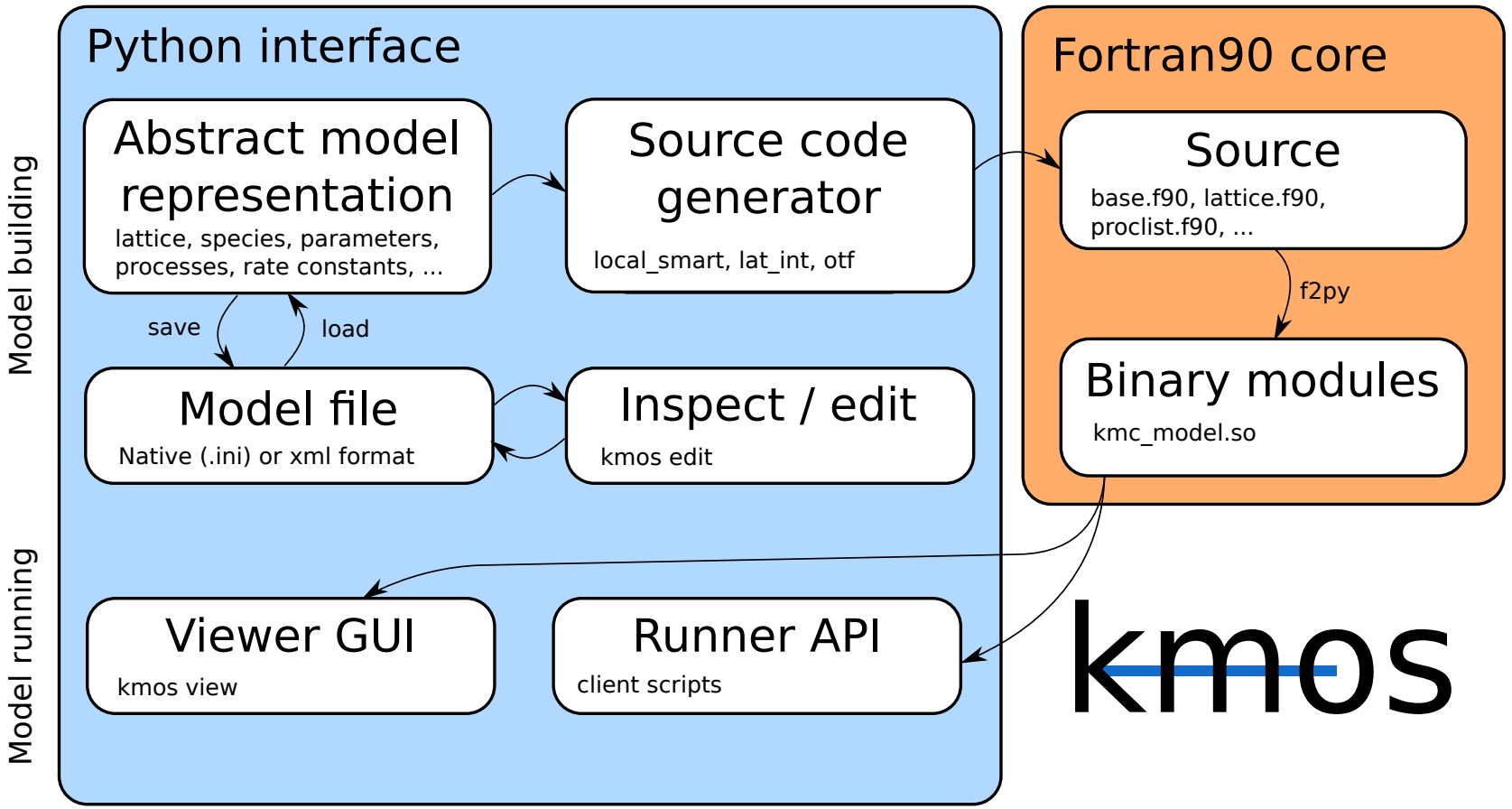
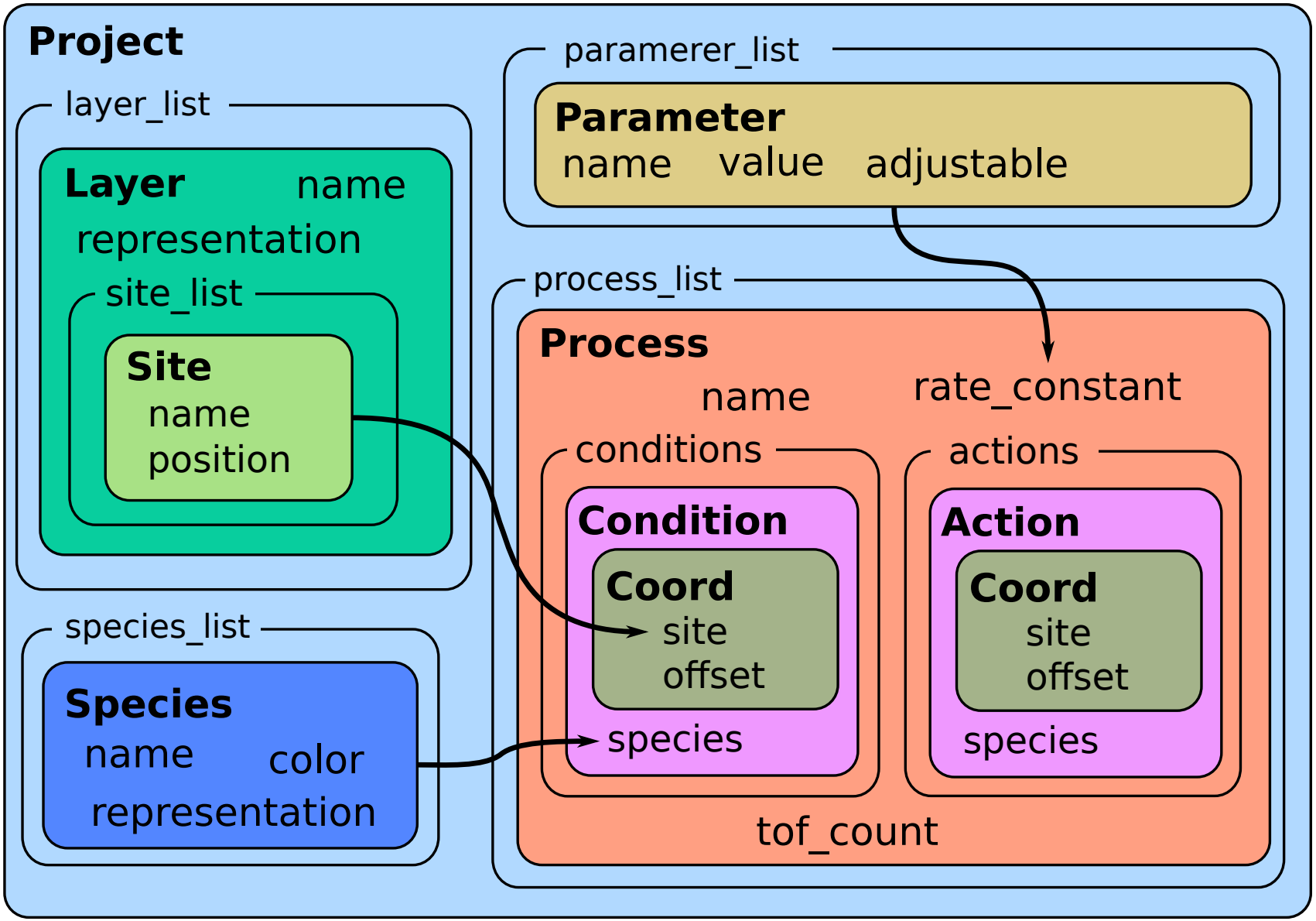
Welcome to Day 2!



Abstract model representation in kmos



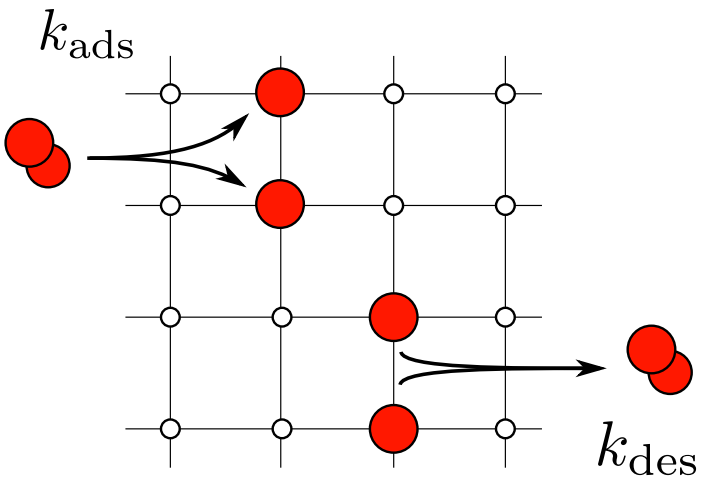
Building a kmos model

Example

Dissociative adsorption and associative desorption of O_2 on a square lattice.

Objectives

- Understand all basic elements of a kmos project: lattice, sites, species, parameters and processes.
- Generate and understand the model file.
- Learn to export the model



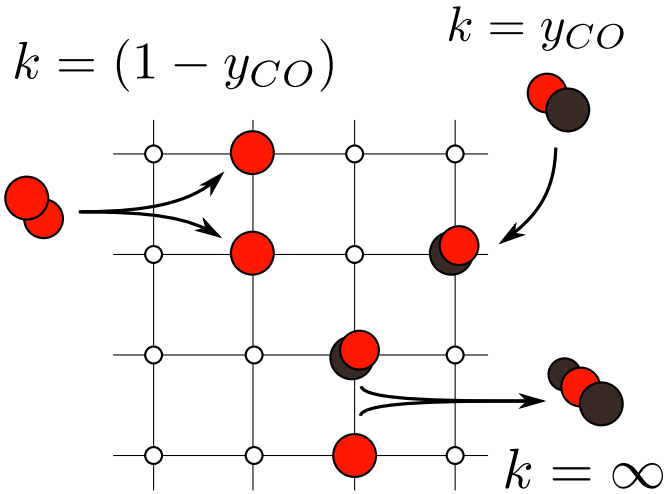
TASK 7: The ZGB Model

Objectives

Extend the simple adsorption/desorption model to a full CO oxidation model.

The ZGB[1] Model

- Simple square lattice.
- Only parameter: CO fraction y_{CO}
- No desorption
  To avoid deadlocks (states with no possible process) include desorption processes with very low rate
- When O and CO approach (nearest neighbors), they react immediately (model this with a very large rate constant).



[1] R. M. Ziff et al. Phys. Rev. Lett. 56 (1986) 2553

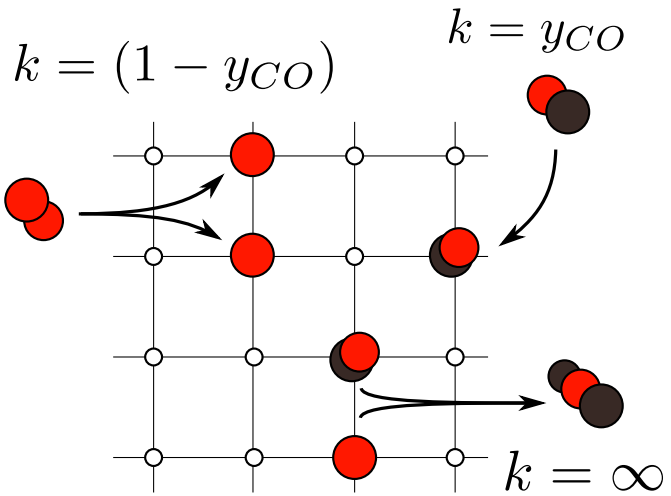
TASK 7: The ZGB Model

Objectives

Extend the simple adsorption/desorption model to a full CO oxidation model.

The ZGB[1] Model

- Extend the O_2 ads/des model to include the missing processes
- Change the parameters to include y_{CO}, k_∞ and k_{slow}.
- Test the model by plotting TOF and coverages for 10-15 values of $y_{CO} \in [0.3, 6.5]$.
- Observe how this is affected by system size.



[1] R. M. Ziff et al. Phys. Rev. Lett. 56 (1986) 2553

Sampling Turnover Frequencies

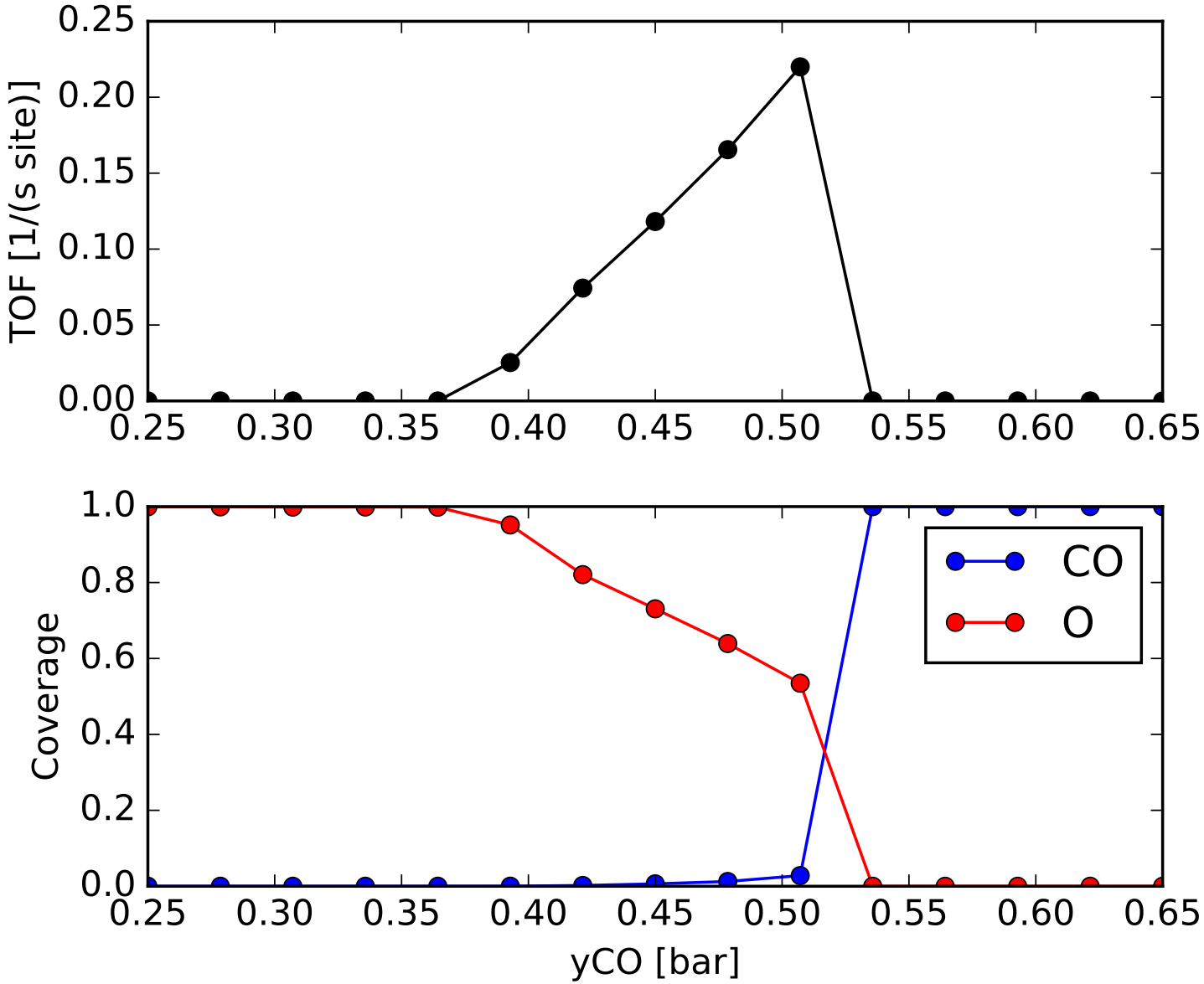
We need to tell `kmos` which processes we want to calculate the turnover frequency (TOF) for. For this we use the optional parameter `tof_count` when adding a process

Example:

```
pt.add_process(name = 'CO_oxi_right'.format(i),
               conditions = some_conds, actions = some_acts,
               rate_constant = 'kfast',
               tof_count = {'CO_oxidation' : 1})
```

- `tof_count` accepts a python dictionary
  - *keys* are the name of the TOF (e.g. `'CO_oxidation'`)
  - *values* indicate how this processes contributes to the corresponding TOF
- Each process can contribute to more than one TOF.
- Processes can contribute negatively to a TOF (e.g. reverse processes).
- TOFs are *defined* by the processes' `tof_count` dictionaries

TASK 7: The ZGB Model



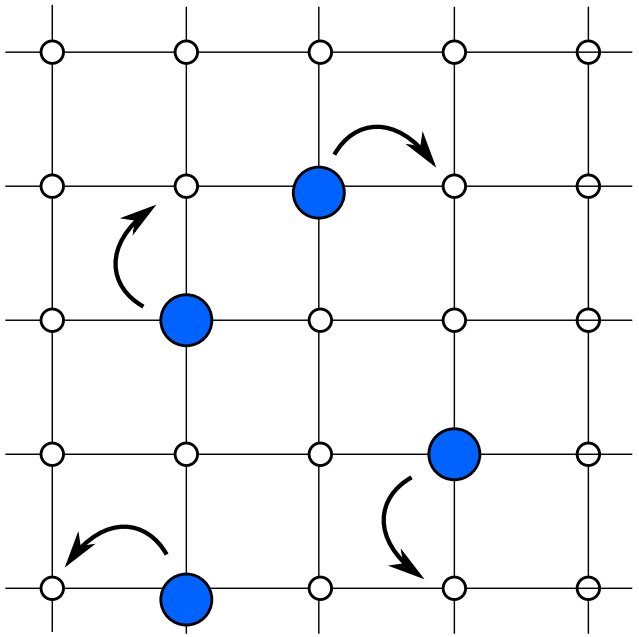
Modeling (lattice) diffusion

A first step in modeling ion diffusion in energy materials.

Model

- Square lattice
- Diffusing species *"ion"*
- Diffusion in all directions

$$k_{\text{diff}} = \frac{k_B T}{h} e^{-\frac{E_0}{k_B T}}$$



Issue

Starting with an empty lattice no process can occur. Two possible solutions:

- Pre-fill the lattice with some ions.
- Implement special boundary conditions

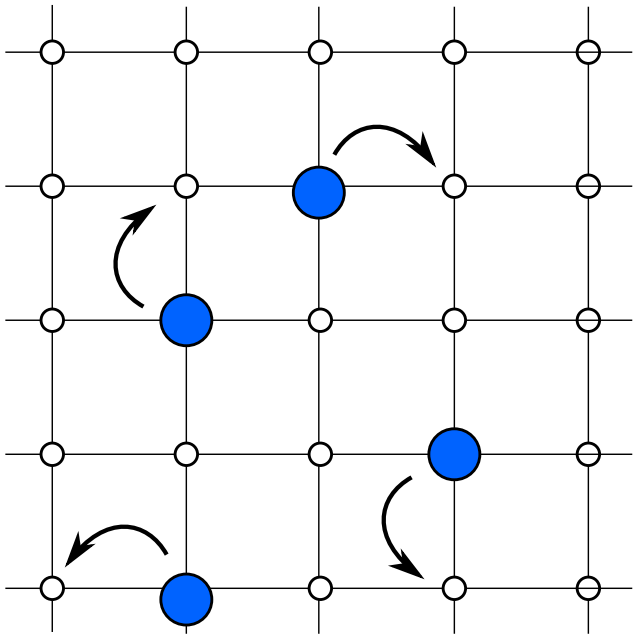
Modeling (lattice) diffusion

A first step in modeling ion diffusion in energy materials.

Model

- Square lattice
- Diffusing species *"ion"*
- Diffusion in all directions

$$k_{\text{diff}} = \frac{k_B T}{h} e^{-\frac{E_0}{k_B T}}$$



Issue

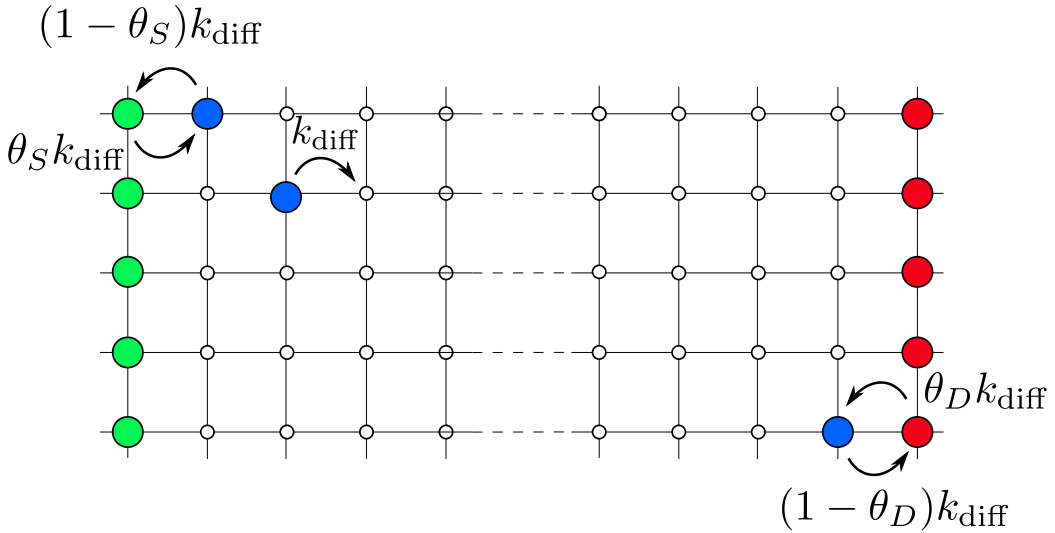
Starting with an empty lattice no process can occur. Two possible solutions:

- Pre-fill the lattice with some ions.
- **Implement special boundary conditions**

Special boundary conditions

Reservoirs

Boundaries of the system as a couple of reservoirs (*source* and *drain*) at fixed nominal concentrations (θ_S and θ_D).



The `kmc_settings.py` file

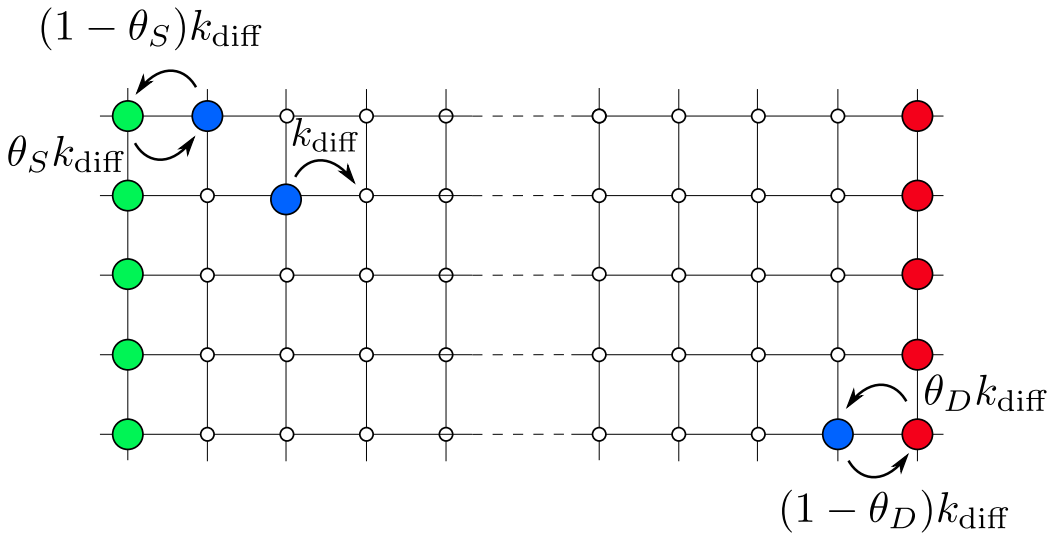
We want our model to always be initialized with these boundary conditions. `kmc_settings.py` controls a model's start-up options.

TASK 8: Implement special boundary conditions

Objective

Implement these boundary conditions on the diffusion model.

- Add the auxiliary species and the entry and exit processes.
- Write a function to add the reservoirs on the (lateral) edges of the system.
- Include such function in kmc_settings.py, to be loaded at start-up automatically.
- Test using kmos view.



Going deeper into the kmos API.

Calculating ion current

- kmos' TOFs are not so useful for this diffusion model.
- We can use the kmos API instead
  - `model.base.get_procstat` takes the number of a process and return the number of times this process has been executed.
  - The process number are stored in the attributes `model.proclist.<process_name>`

For example

To get the number of times the process called `'source_entry'` executed, we use

```
nr_of_entries = model.base.get_procstat(model.proclist.source_entry)
```

in a client script or interactively.

Try on your own!: Open the interactive kmos shell, run some steps (e.g. `model.do_steps(1e5)`) and then check how many entry processes occurred.

TASK 9: Extending and testing the lattice diffusion model

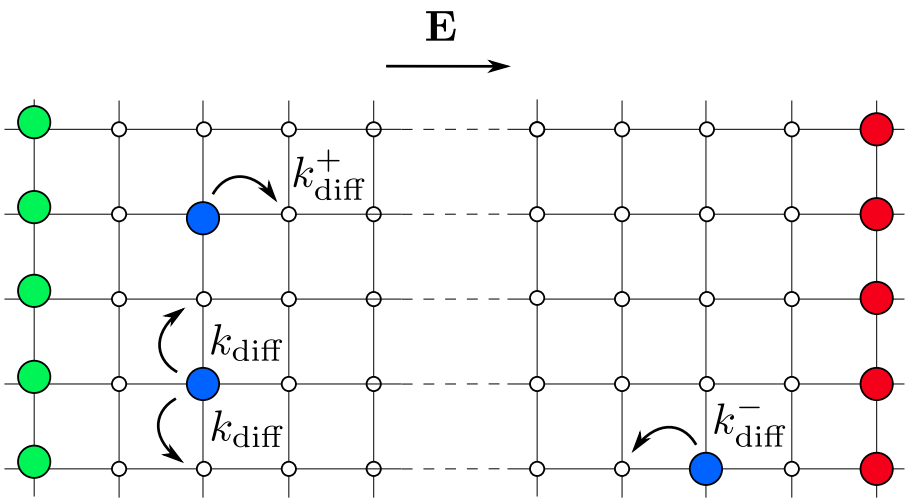
Objective

Extend the model to account for an external electric field. Analyze the effects of field and carrier density on current.

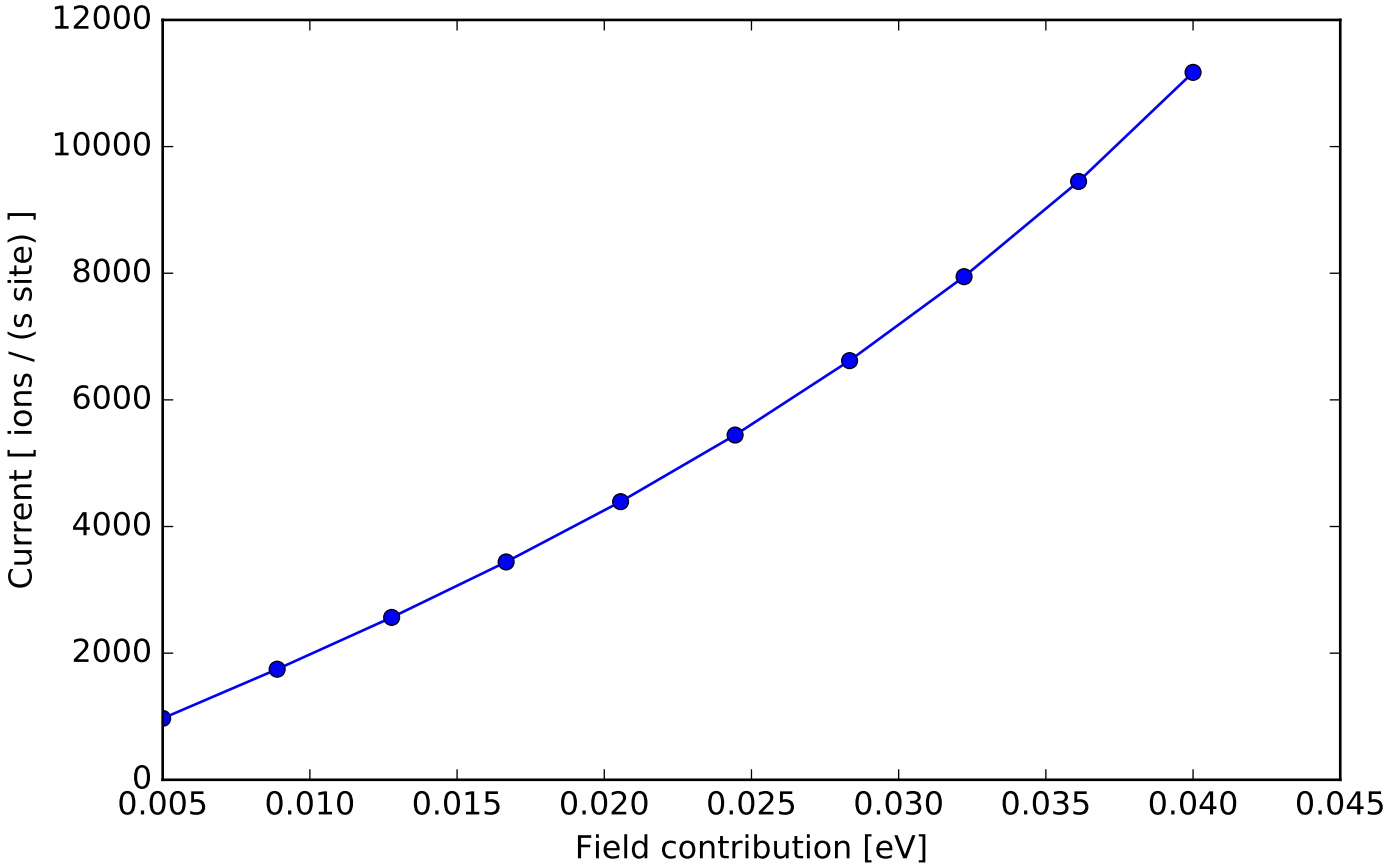
- Add a parameter ϵ_{field} that represents the field strength.
- Modify the right and left diffusion processes accordingly.
- Write a client script to calculate the (steady-state) current.
- Make plots of
  - Current as a function of ϵ_{field} for fixed $\theta_S = \theta_D = 0.5$
  - Current as a function of θ, for fixed $\epsilon_{\text{field}} = 0.02$ and $\theta_S = \theta_D = \theta$.

$$k_{\text{diff}}^{\pm} = \frac{k_B T}{h} e^{\frac{E_0 \mp \epsilon_{\text{field}}}{k_B T}}$$

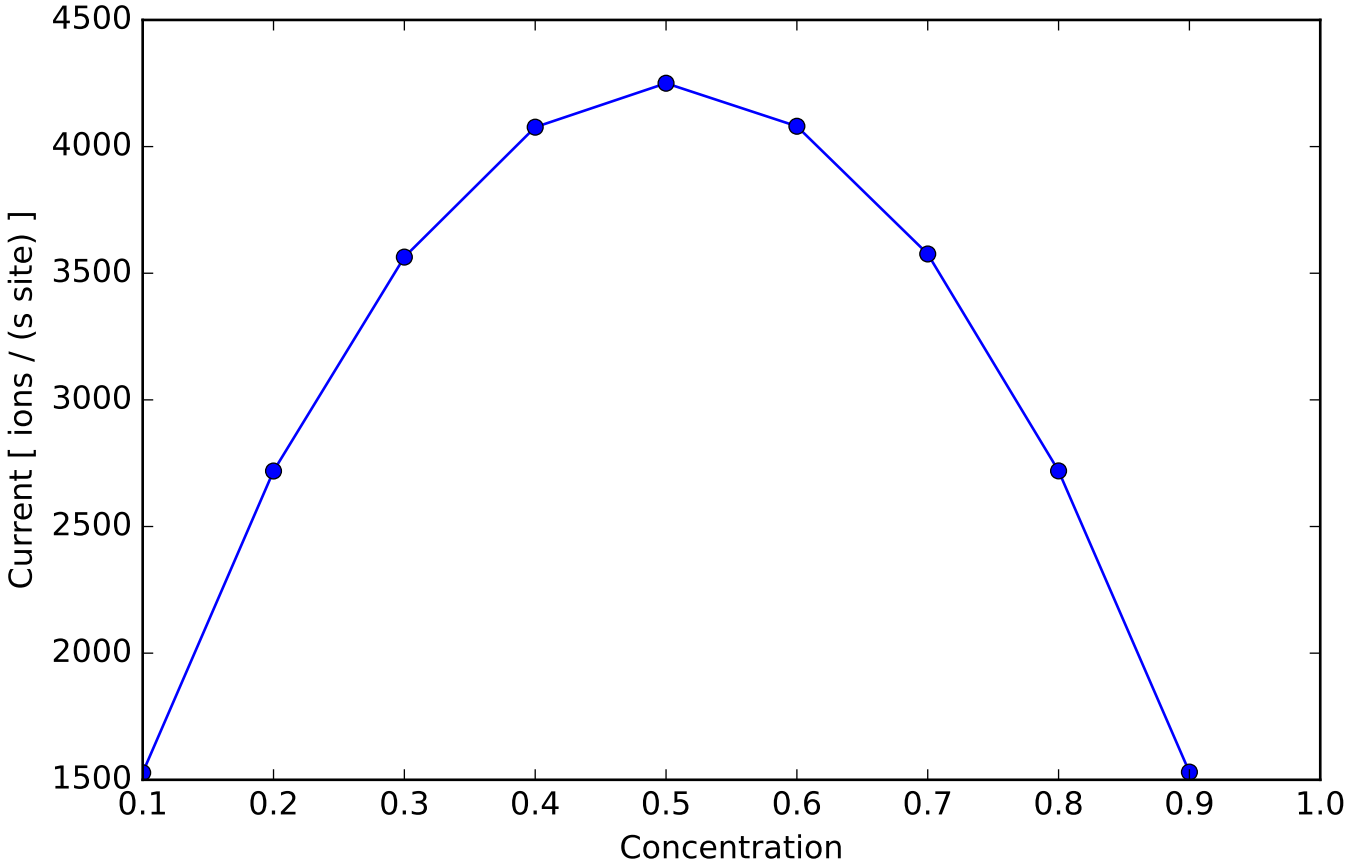
$$= k_{\text{diff}}\ e^{\pm \epsilon_{\text{field}}/k_B T}$$



TASK 9: Extending and testing the lattice diffusion model



TASK 9: Extending and testing the lattice diffusion model



Handling lateral interactions

Example

Presence of nearest neighbors (NN) increases the rate of diffusion according to:

$$k_{\text{diff}}(n_{NN}) = \exp\left(-\frac{E_0 - n_{NN}\epsilon_{\text{int}}}{k_B T}\right) = k_{\text{diff}}(0)\exp\left(\frac{n_{NN}\epsilon_{\text{int}}}{k_B T}\right)$$

where n_{NN} is the number of neighbors.

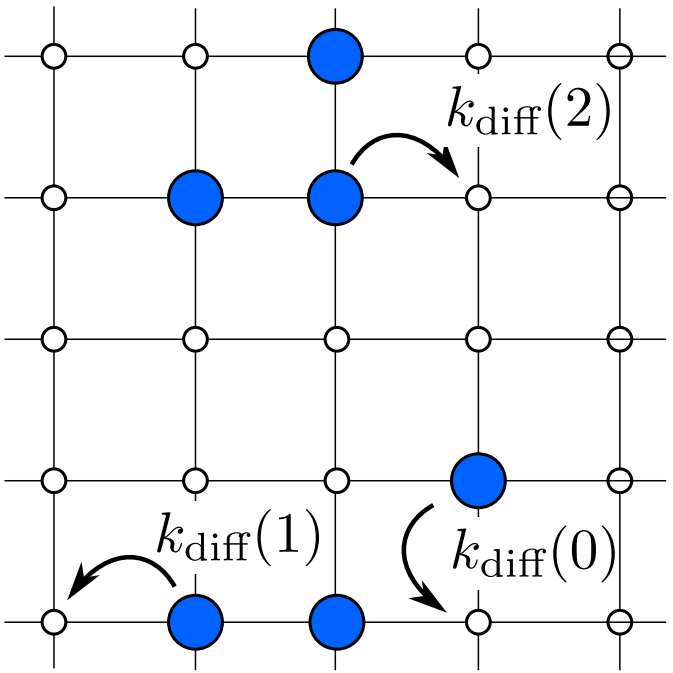
Lateral interactions

- Different rate constant $\rightarrow$ different process
- Impractical to define each process manually

Solution

The `itertools.product` python module.

In particular: `itertools.product`.

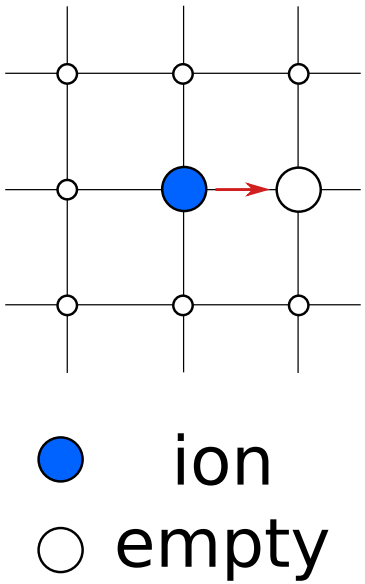


Handling Lateral Interactions

Example

Diffusion to the right

- Conditions:
  `ion@center`; `empty@right`
- Actions:
  `empty@center`; `ion@right`

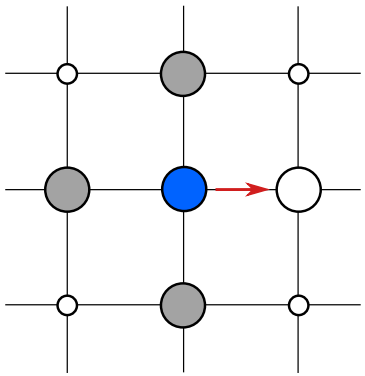


Handling Lateral Interactions

Example

Diffusion to the right

- Conditions:
  `ion@center`; `empty@right`
  `??@up`; `??@left`; `??@down`
- Actions:
  `empty@center`; `ion@right`



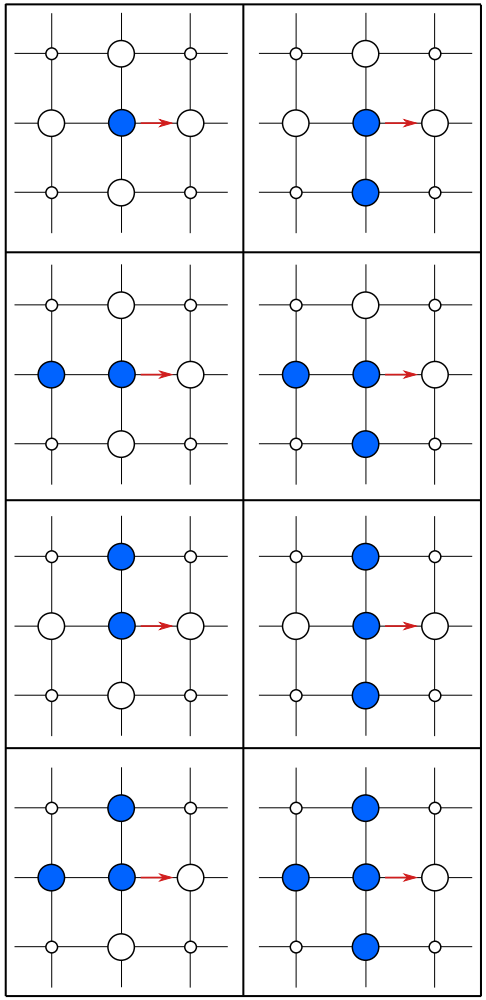
Handling Lateral Interactions

All combinations

```
from itertools import product
for conf in product(['empty','ion'], repeat=3):
    print conf
```

Output

```
('empty', 'empty', 'empty')
('empty', 'empty', 'ion')
('empty', 'ion', 'empty')
('empty', 'ion', 'ion')
('ion', 'empty', 'empty')
('ion', 'empty', 'ion')
('ion', 'ion', 'empty')
('ion', 'ion', 'ion')
```



Handling lateral interactions

In kmos

```
import itertools
counter = 0  # process counter
# Combinations of ['empty','ion'] taken in sets of three
for conf in itertools.product(['empty','ion'], repeat=3):
    # default conditions and actions
    conds = [Condition(species='ion', coord=center),
             Condition(species='empty', coord=right)]
    acts = [Action(species='ion', coord=right),
            Action(species='empty', coord=center)]
    # add case-specific conditions
    for i, coord in enumerate([left, down, right]):
        conds.append(Condition(species=conf[i], coord = coord))
    # define the rate constant and add the process
    nint = conf.count('ion')  # count number of NN ions
    rc = '1/(beta*h)*exp(-beta*(E0-{}*eps_int)*eV)'.format(nint))
    pt.add_process(name='diffusion_up_{:03d}'.format(counter),
                   conditions=conds, actions=acts, rate_constant=rc)
    counter += 1 # update the process counter
```

TASK 10: Solid-on-Solid crystal growth model

Solid-on-solid growth model

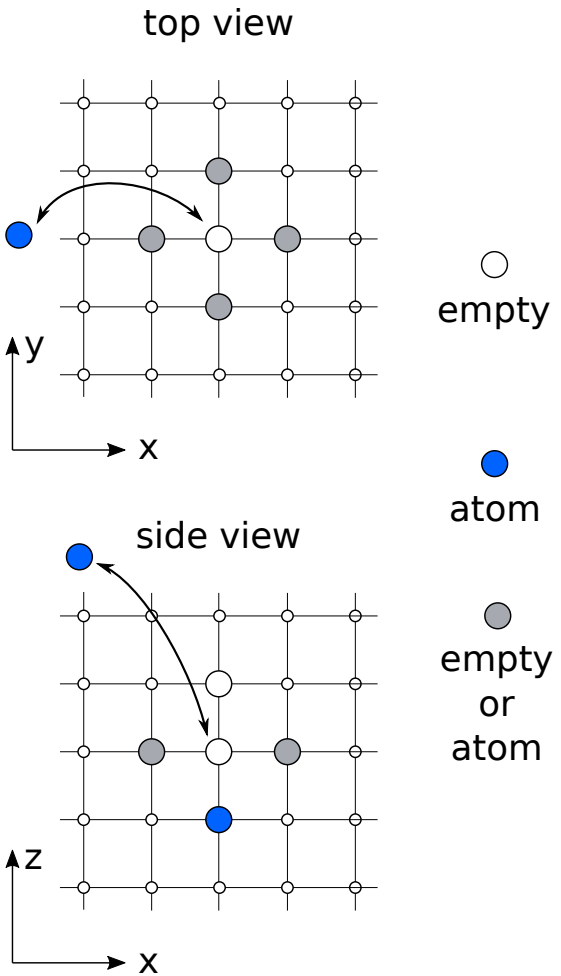
- Simple growth model
- Adsorption rate $k_{\text{ads}} = 3 \times 10^{-3}\ \text{s}^{-1}$
- Desorption rate

$$k_{\text{des}} = \frac{k_B T}{h} \exp\left(-\frac{\Delta E_0 + n_{NN}\epsilon_{\text{int}}}{k_B T}\right)$$

  - Depends on T
  - Affected by attractive interactions
  - $\Delta E_0 = 1.0\,\text{eV} \quad \epsilon_{\text{int}} = 0.5\,\text{eV}$

Objectives

- Implement the SOS model in kmos
- Test how growth structure depends on T



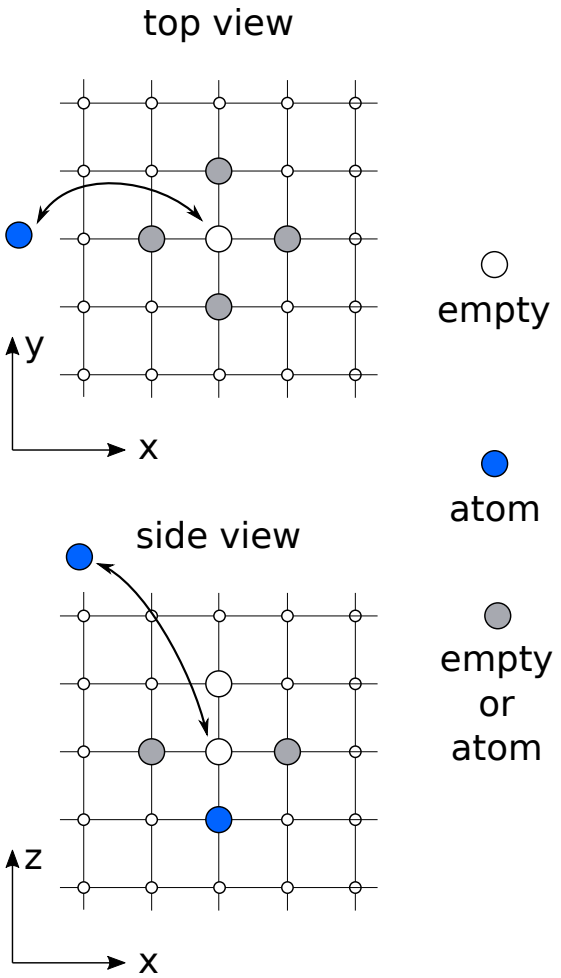
TASK 10: Solid-on-Solid crystal growth model

Caveats

- `kmos edit` only works for 2D models.
- Growing crystal must not reach the top of the simulation box.

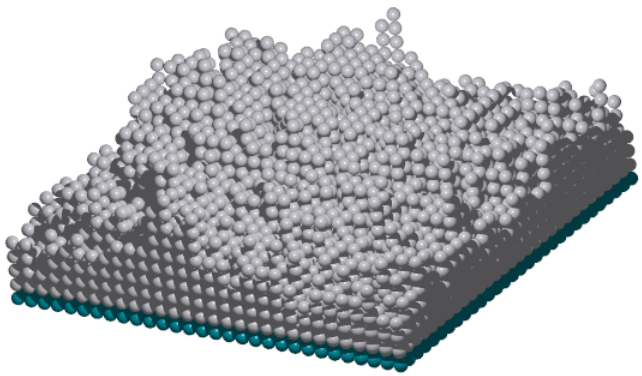
Hints

- The systems state needs to be specially prepared to allow initial processes.
- An auxiliary `'substrate'` species might be necessary to start growth process.
- The state of the system can be saved by `model.dump_config('some_name')`
- The generated file can be read using `model.load_config('some_name')`

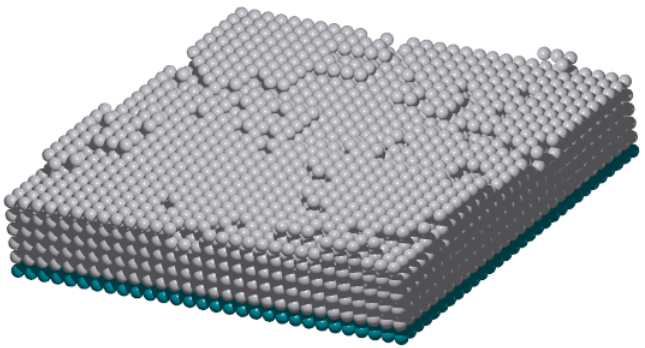


TASK 10: Solid-on-Solid crystal growth model

T = 350K



T = 450K



Thank you!

Juan M. Lorenzi
juan.lorenzi@tum.de

Mie Andersen
mie.andersen@ch.tum.de

KMOS is under active development

- Enhanced simulation of *stiff* problems
- Efficient lateral interactions
- Trajectories and auto-correlation functions
- ...

Development: `https://github.com/mhoffman/kmos`
Documentation: `http://kmos.readthedocs.io/`
Mailing list: `https://groups.google.com/group/kmos-users/`