PRISMS Center

Center for PRedictive Integrated Structural Materials Science

CASM Demo – Day 1

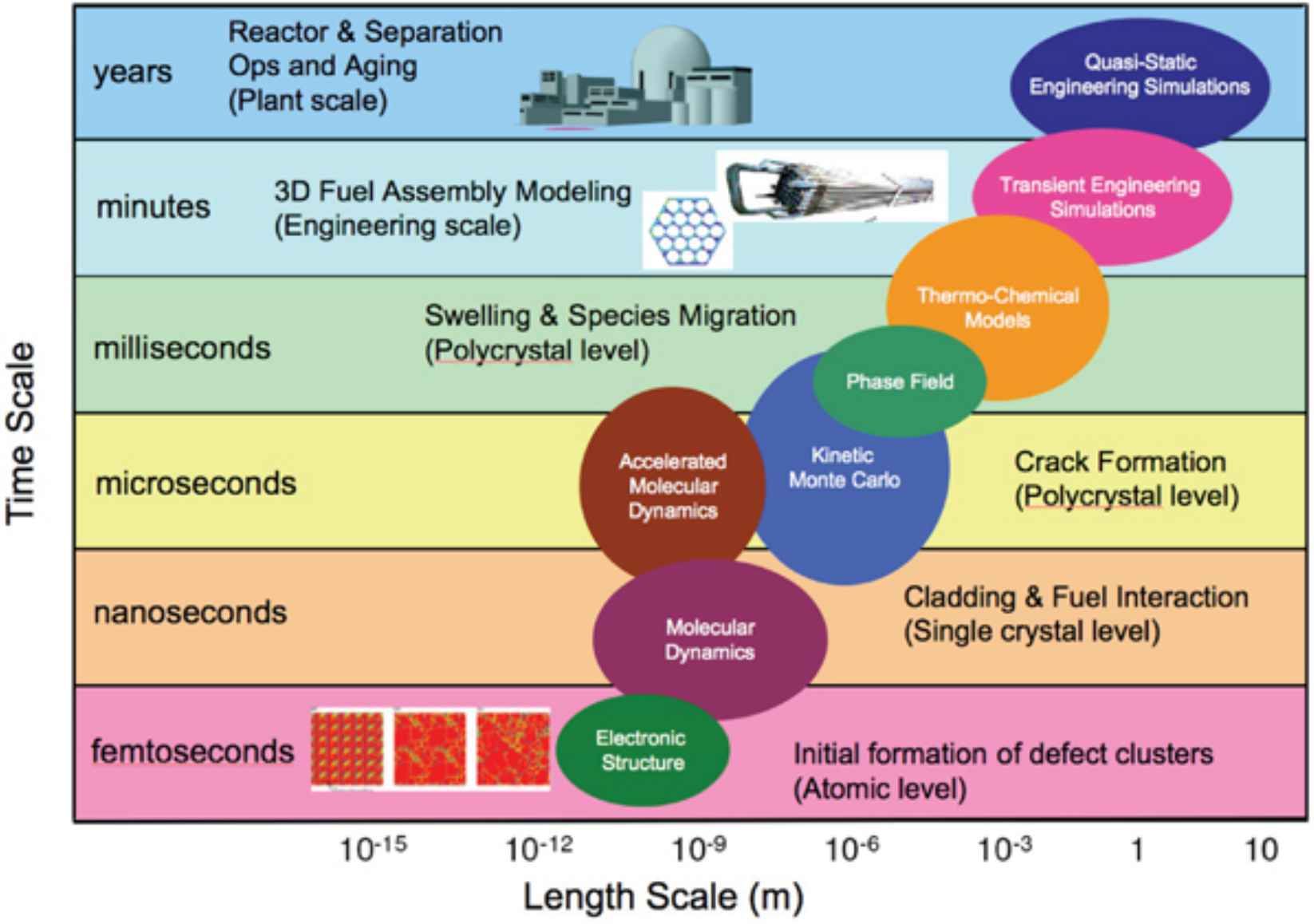
John C. Thomas, Brian Puchala, Anton Van der Ven
Anirudh Natarajan, John Goiri, Min-Hua Chen,
Max Radin, Jonathon Bechtel, Elizabeth Decolvenaere

PRISMS Center Workshop
August 15-17, 2016
Website: prisms-center.org

U.S. DEPARTMENT OF ENERGY | Office of Science

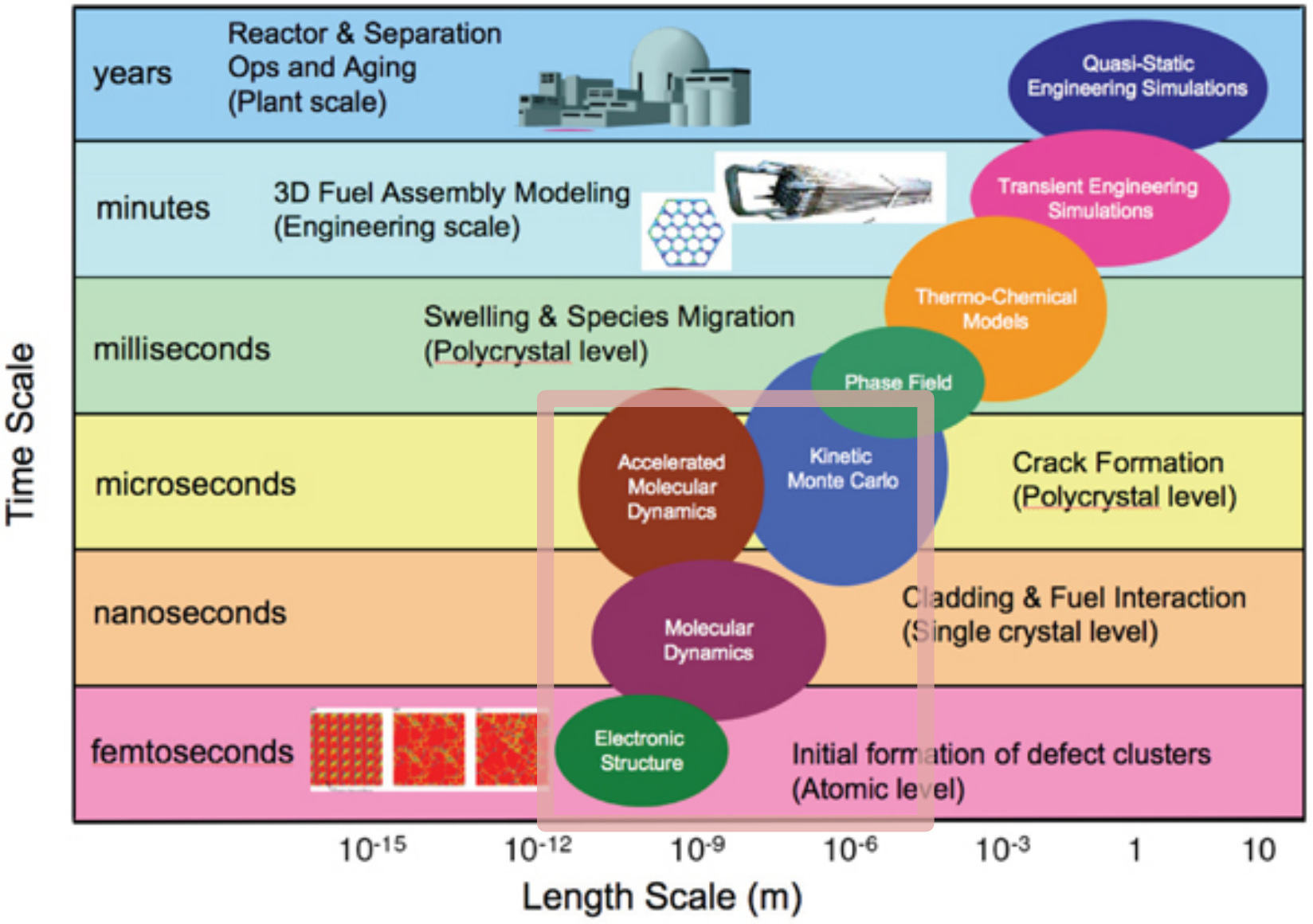
Center for PRedictive Integrated Structural Materials Science

A persisting problem



Source: http://www.lbl.gov/cs/html/exascale4energy

A persisting problem



Source: http://www.lbl.gov/cs/html/exascale4energy

From atoms to microns

Calculate **accurate** energies of the system in 10^2-10^3 different **states**

From atoms to microns

Solve $\hat{H}\Psi = E\Psi$

better than ~10 meV/atom

Calculate **accurate** energies of the system in 10^2-10^3 different **states**

- Chemical species
- Charge states
- Atomic displacement
- Strain

From atom- to micron-scale

Identify suitable variables and basis functions for parameterizing energy model

Calculate accurate energies of the system in 10^2-10^3 different states

From atom- to micron-scale

Unknown

$$U(\varepsilon_{11},\varepsilon_{12},\ldots,x_i,y_i,z_i,\ldots) = \vec{V}^{\mathrm{T}}\vec{\Phi}(\varepsilon_{11},\varepsilon_{12},\ldots,x_i,y_i,z_i,\ldots)$$

Must be invariant to symmetry!!

Identify suitable variables and basis functions for parameterizing energy model

Calculate accurate energies of the system in 10^2-10^3 different states

From atom- to micron-scale

Use model for material optimization or to simulate material properties

Identify suitable variables and basis functions for parameterizing energy model

Calculate accurate energies of the system in 10^2-10^3 different states

From atom- to micron-scale

- Free energy
- Phase diagrams
- Elastic tensors
- Thermal conductivity
- Diffusivity
- Vibrational spectra

As a function of temperature, stress, chemical potential, etc.

Use model for material optimization or to simulate material properties

Identify suitable variables and basis functions for parameterizing energy model

Calculate accurate energies of the system in 10^2-10^3 different states

From atom- to micron-scale

Share data with colleagues for experimental design or simulation at longer length/time scales

Use model for material optimization or to simulate material properties

Identify suitable variables and basis functions for parameterizing energy model

Calculate accurate energies of the system in 10^2-10^3 different states

From atom- to micron-scale

Share data with colleagues for experimental design or simulation at longer length/time scales

Use model for material optimization or to simulate material properties

Identify suitable variables and basis functions for parameterizing energy model

Calculate accurate energies of the system in 10^2-10^3 different states

Construct structural hypotheses of what happens in the real material

From atom- to micron-scale

Share data with colleagues for experimental design or simulation at longer length/time scales

Use model for material optimization or to simulate material properties

Identify suitable variables and basis functions for parameterizing energy model

Calculate accurate energies of the system in 10^2-10^3 different states

Construct structural hypotheses of what happens in the real material

CASM Tutorial Outline

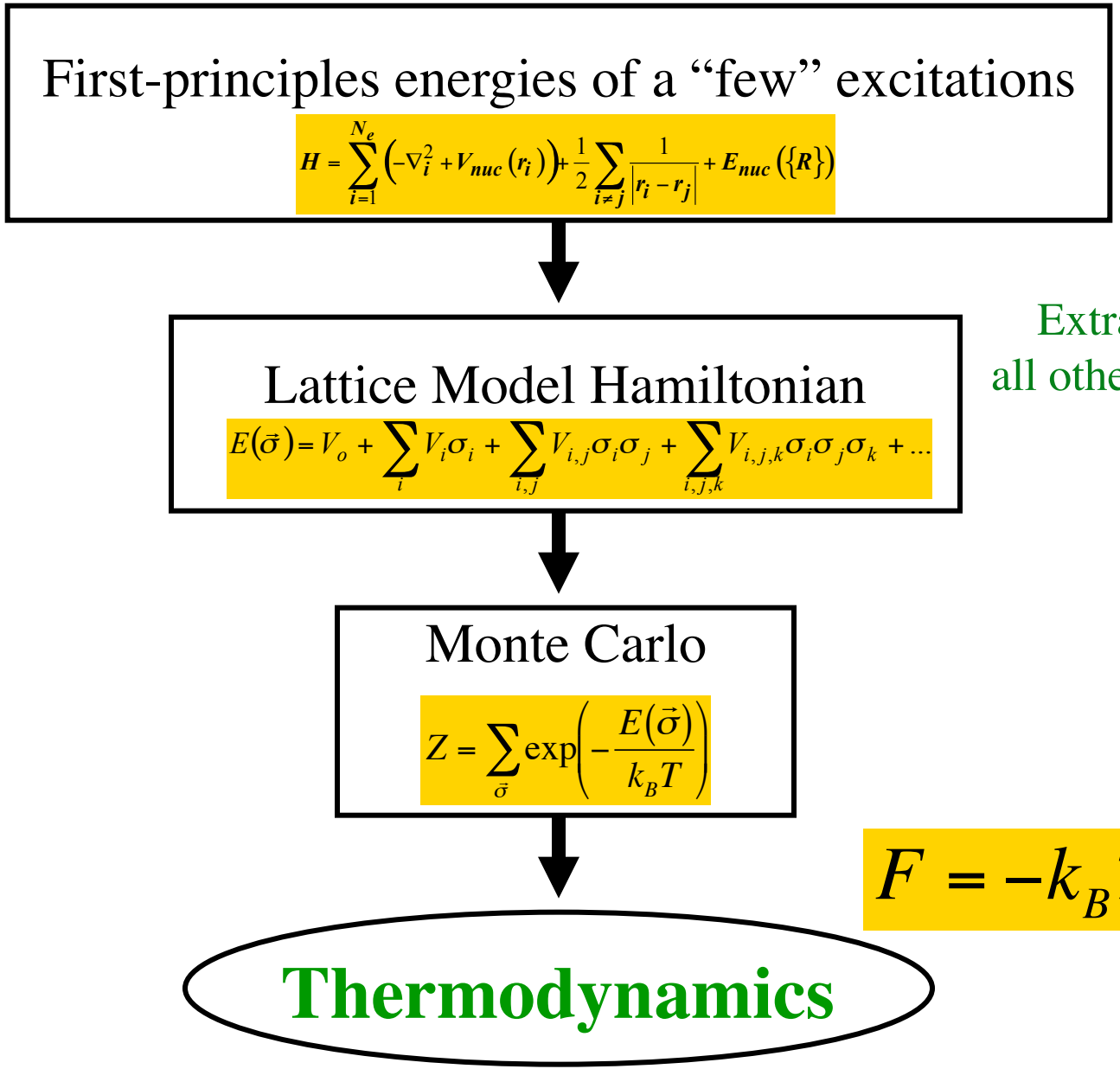
Today:

- Crystal structure identification & specification
- Symmetry analysis
- Using CASM to manage data
  - Configuration selections and query syntax
  - Maintaining sanity with cluster expansion profiles
- Cluster expansion basics
  - Configuration enumeration
  - Cluster expansion basis sets & correlation evaluation

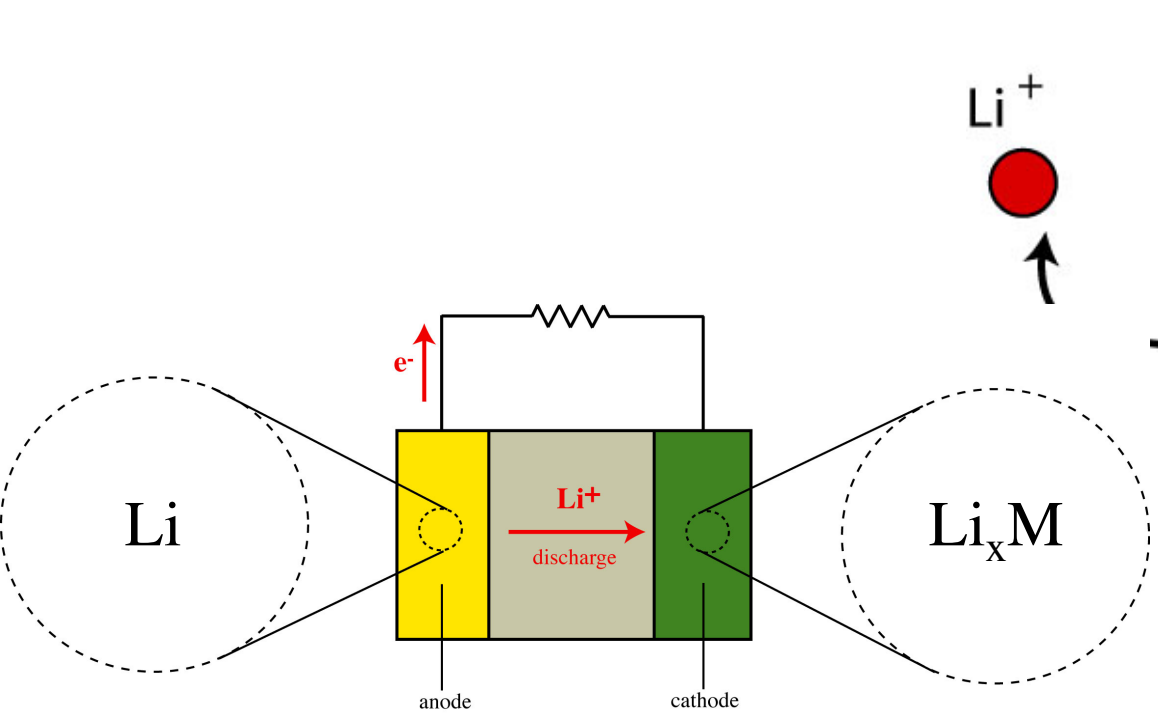
Tomorrow (with Brian Puchala):

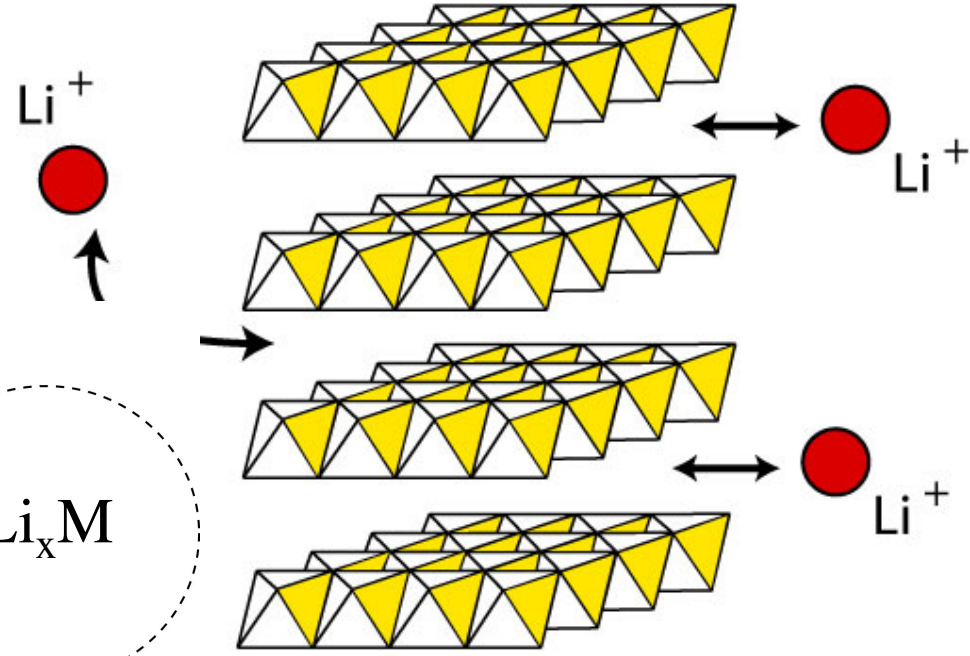
- Parameterizing cluster expansions
- Predicting thermodynamic properties with grand canonical Monte Carlo
- Free energy integration and phase diagram construction

CASM: Clusters Approach to Statistical Mechanics



Lithium ion batteries





Li_xCoO_2 Li_xTiS_2

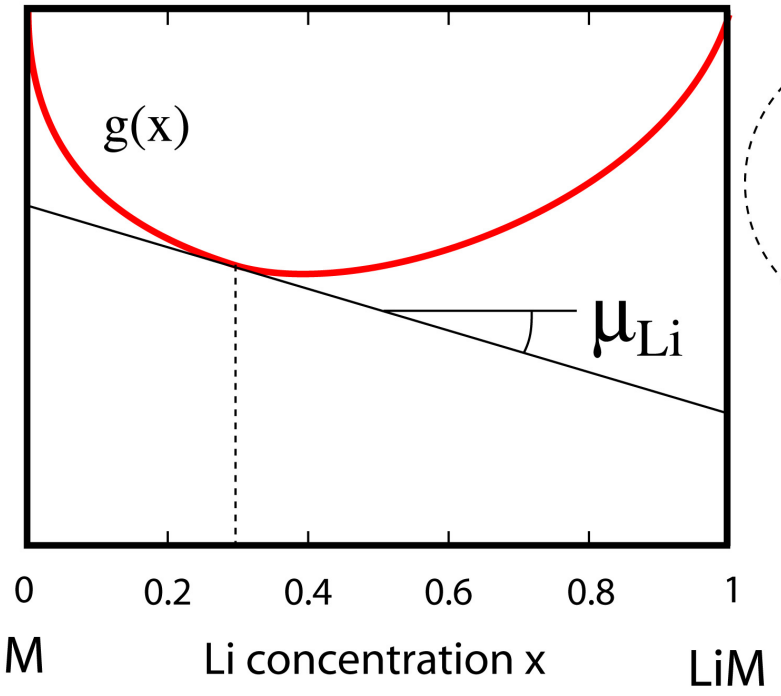
$Li(Ni_{0.5}Mn_{0.5})O_2$

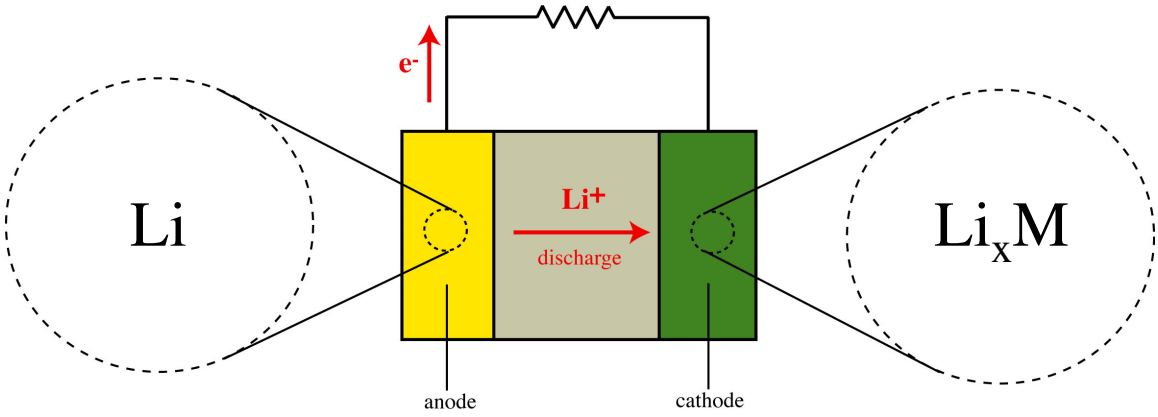
Electrochemical measurements and thermodynamics

$$V = -\left(\mu_{Li}^{cathode} - \mu_{Li}^{anode}\right)/e$$

Gibbs free energy

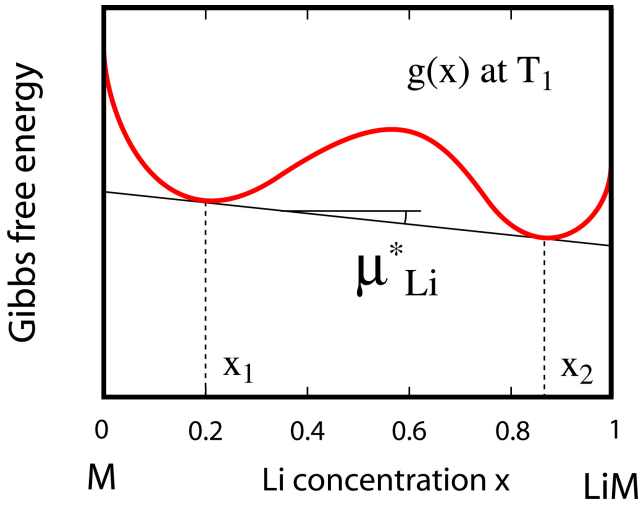
Li_xM



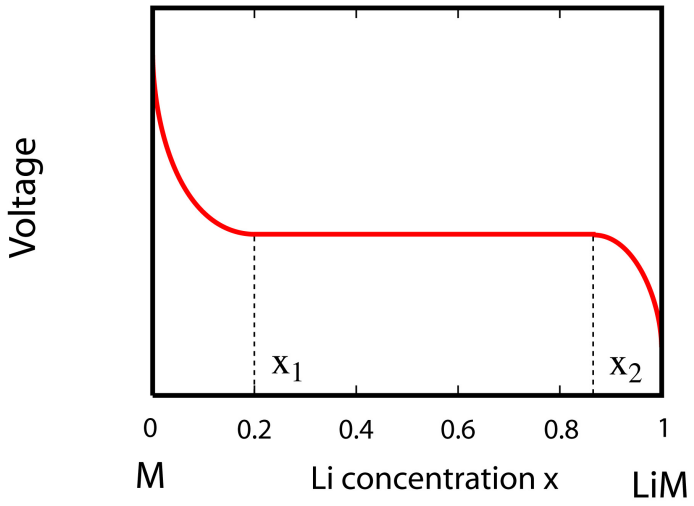


$$\mu_{Li} = \left(\frac{\partial g}{\partial x}\right)_{T,P}$$

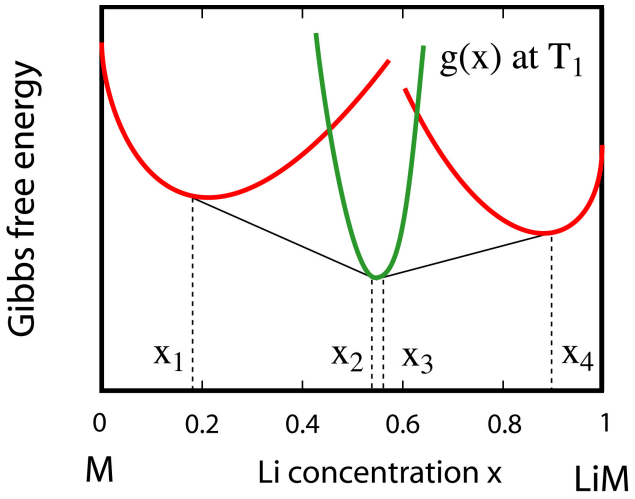
Electrochemical measurements and thermodynamics

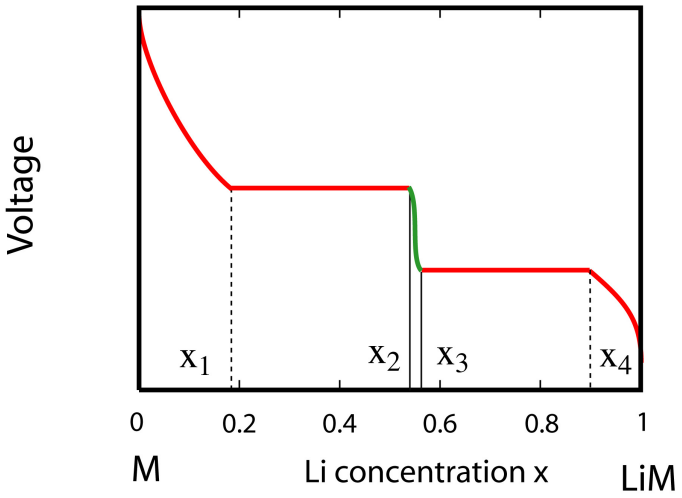


$$V = -\left(\mu_{Li}^{cathode} - \mu_{Li}^{anode}\right)$$

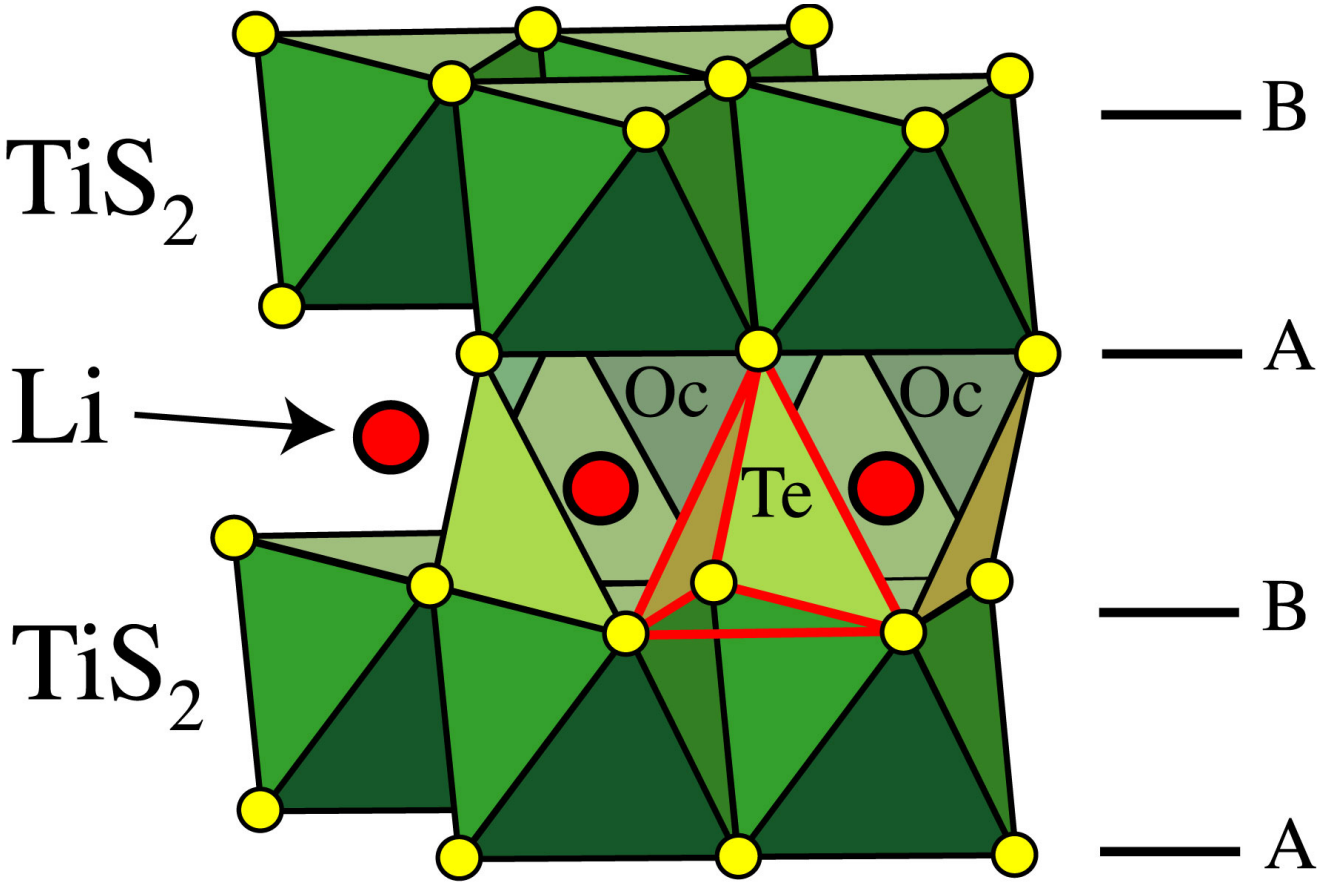


Electrochemical measurements and thermodynamics



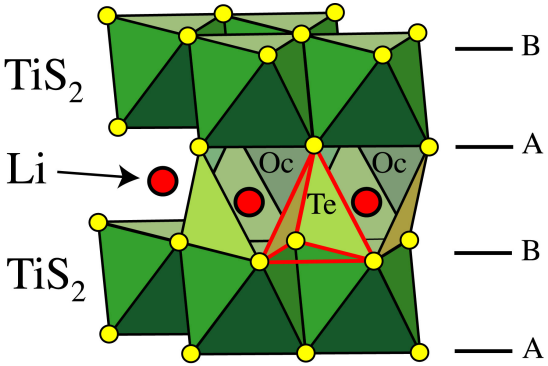


Layered Li_xTiS_2



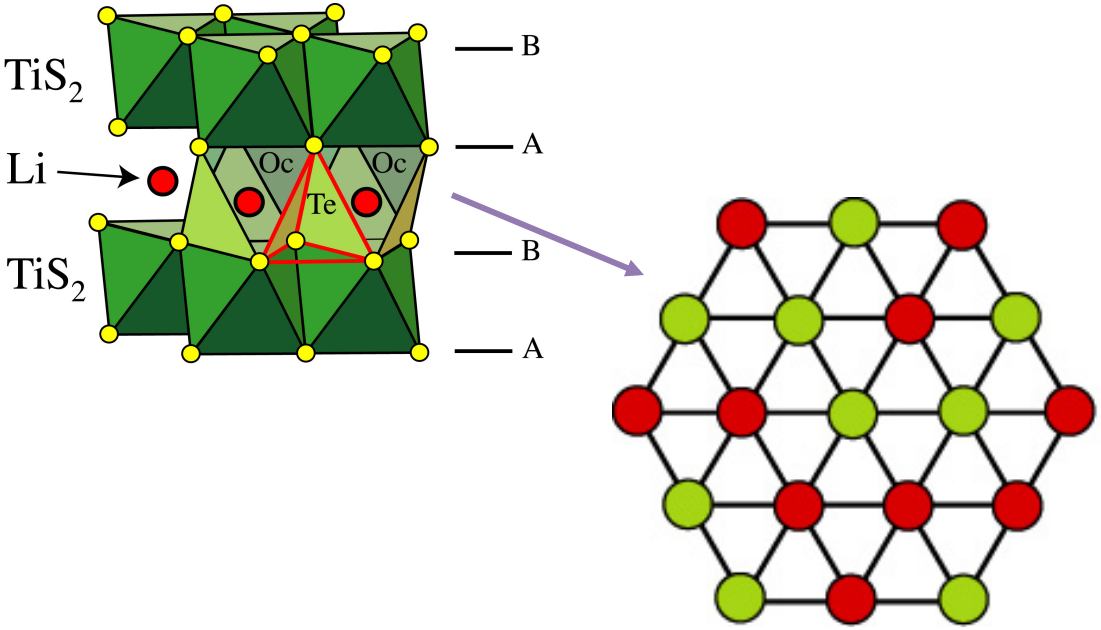
Configurational degrees of freedom

Lithium-vacancy disorder over interstitial sites



Configurational degrees of freedom

Lithium-vacancy disorder over interstitial sites

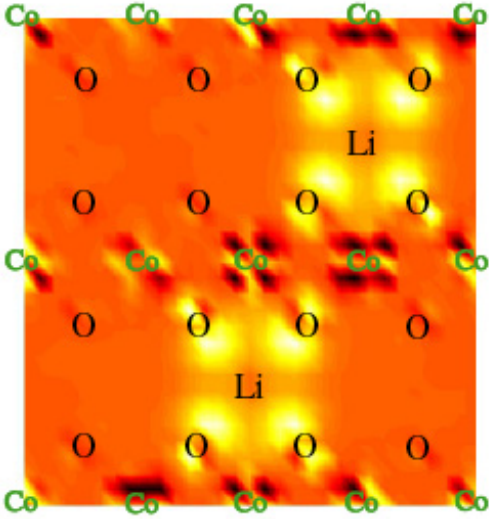


Total of 2^N configurations

Density Functional Theory

$\rho(r)$ = charge density

$$\left[-\nabla_i^2 + V_{ion}(r_i) + \int \frac{\rho(r')}{r - r'} dr' + V_{xc}\left(\rho(r_i)\right)\right]\phi_i(r_i) = \varepsilon_i \phi(r_i)$$



Approximation

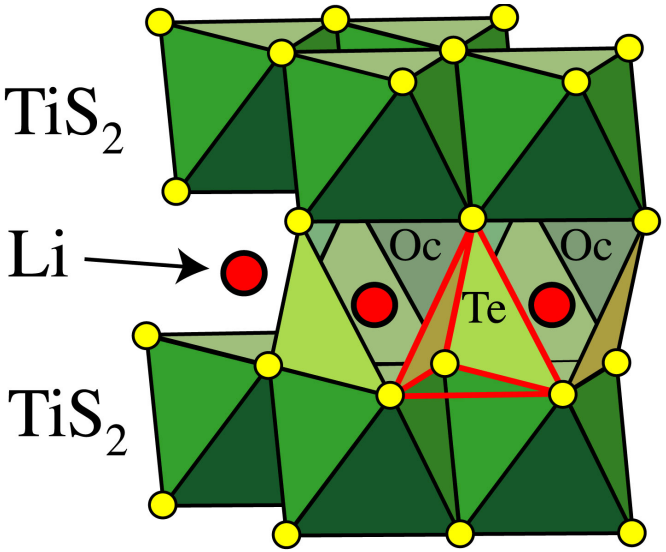
- Local density (LDA)
- Generalized Gradient (GGA)

Energy

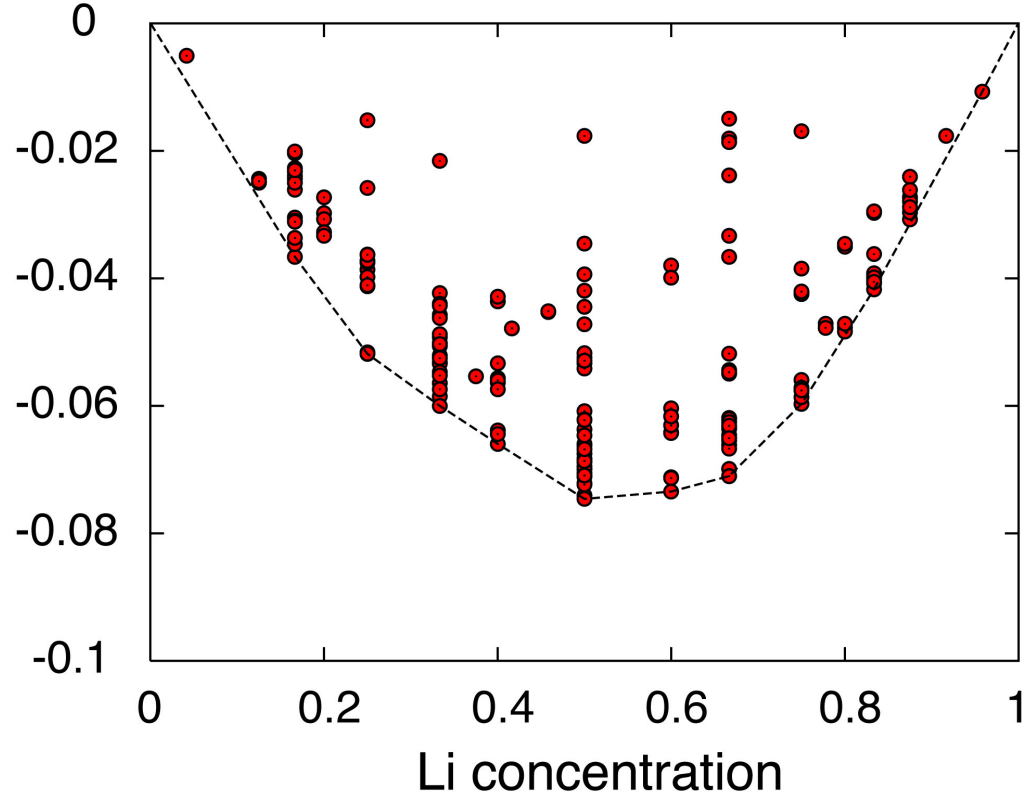
Lattice parameters

Charge density

Layered $LiTiS_2$

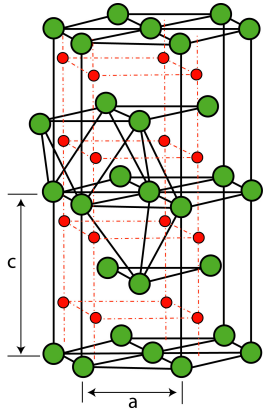


Formation energies at zero Kelvin



DFT (LDA), PAW method using VASP

Tutorial Project: Zr-O binary system



Zr:

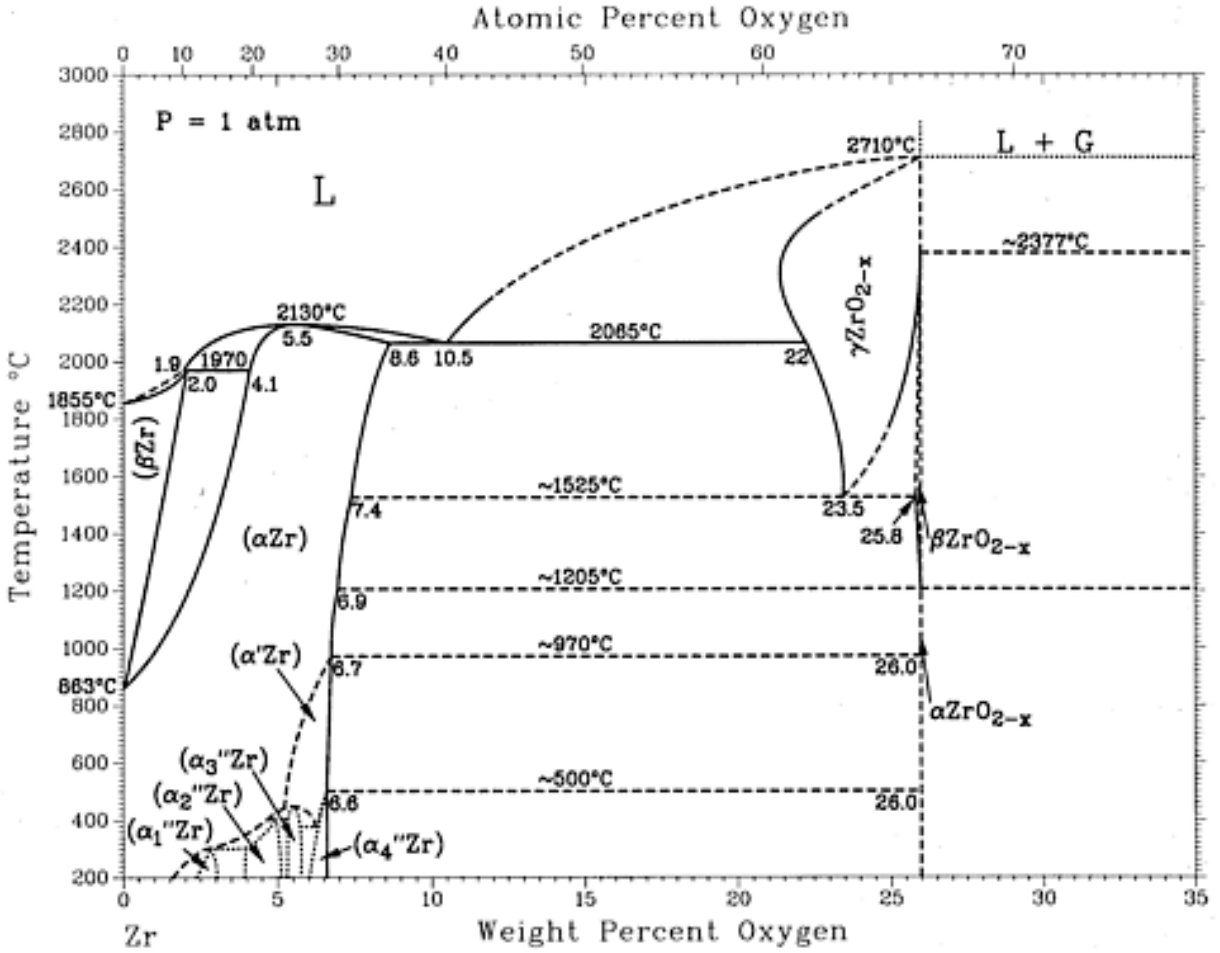
- Very high O solubility

ZrO_x

- Series of suboxide phases

ZrO_{2-x}

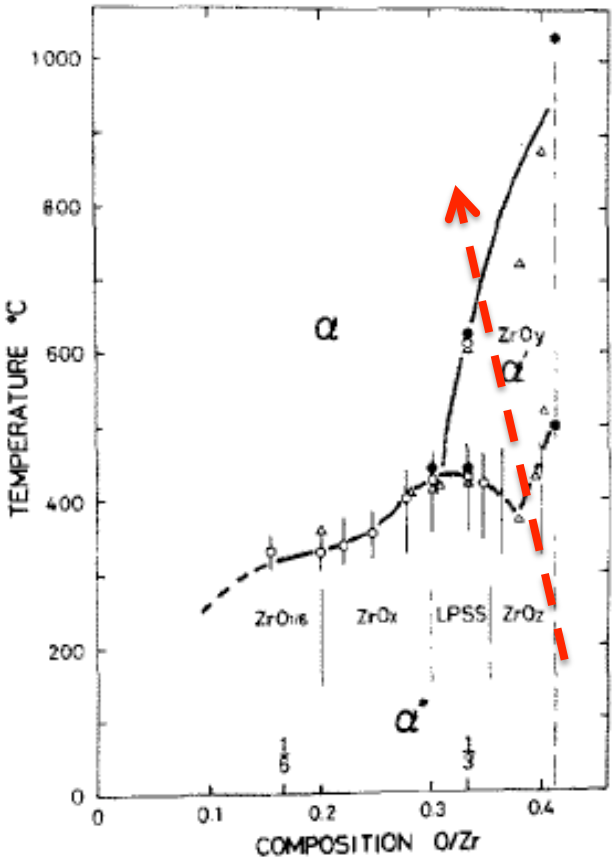
- Oxide

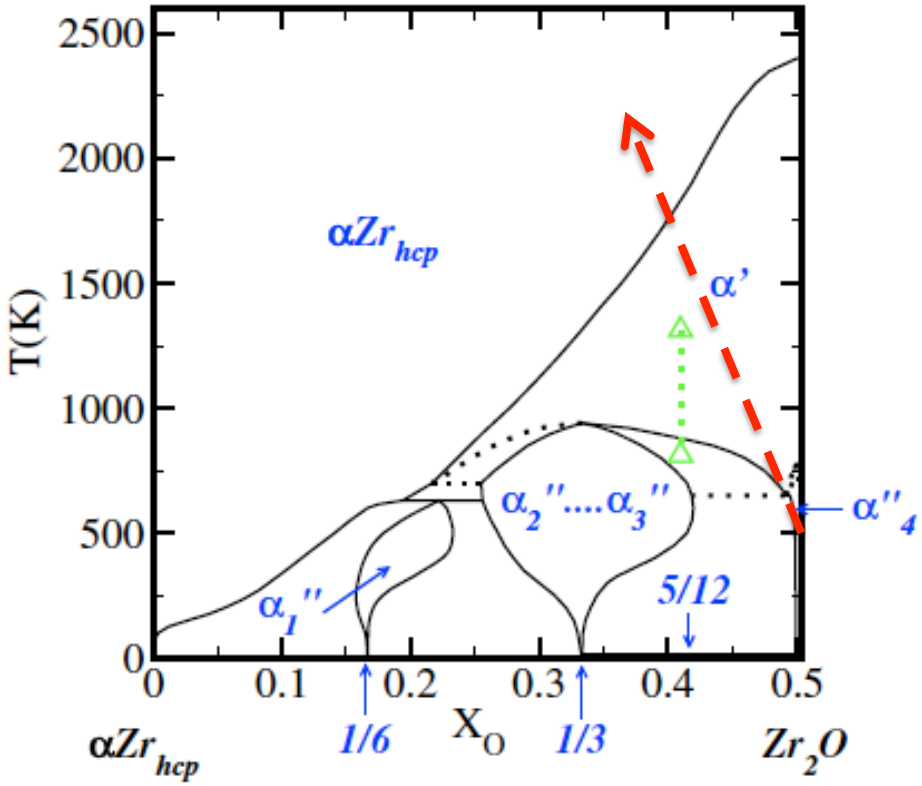


J.P. Abriata, J. Garcés, and R. Versaci, Bulletin of Alloy Phase Diagrams Vol. 7 No. 2 1986 .

Nature of $ZrO_{1/2}$ disordering transitions?

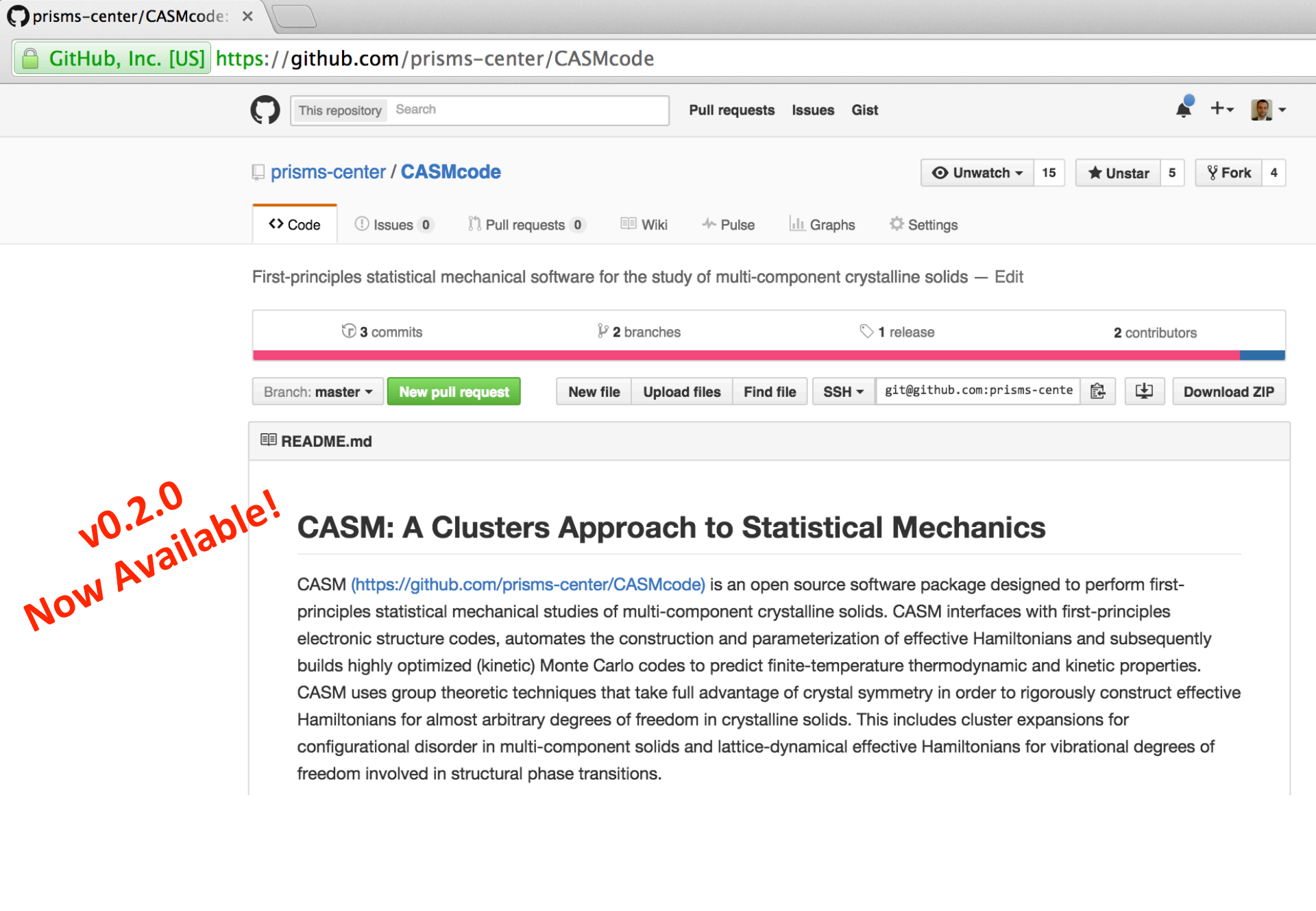
"the rearrangement of the oxygen atoms from uneven to even distribution on the neighboring interstice planes, and then [to a] random distribution in each plane." or is low temperature transition first order?





Arai and Hirabayashi, J. Less-Common Metals, 44, 291 (1976).

Burton, Van de Walle, and Stokes, J. Phys. Soc. Japan, 81, 014004 (2012).



v0.2.0 Now Available!

CASM Design Principles

Reproducibility

- Log all commands over the course of a project
- Try to record provenance of all data as part of normal workflow

Data Management and Automation

- Tools to generate, screen, and dispatch DFT calculations, and then organize, filter, and analyze the results.
- Managed directory tree and dependency framework that allows multiple calculation and model parameters to be tested and compared within a single project

Interoperability

- Most input and output files are stored in JavaScript Object Notation (JSON)file formats, which is widely used and highly readable.
- Weak coupling between CASM and DFT codes, so that CASM can be adapted to work with various existing and future codes.

CASM Design Principles

Collaboration

- Easy archival of projects ('casm files' command), with granular control over amount and type of data to include.
- Parameterized physics models can be shared without data and used to perform finite-temperature simulation

Efficiency

- CASM implements unique and highly optimized algorithms for generating and analyzing data and building physics models.
- Models are automatically converted to optimized C++ code and then compiled for use in Monte Carlo, enabling larger simulations and improved convergence.

CASM: Overview

Specify the problem: a crystal in which one or more sites has multiple possible occupants:

- Oct (O) in HCP-based ZrO_x
- Ni—Al substitution in FCC Ni_xAl_{1-x}

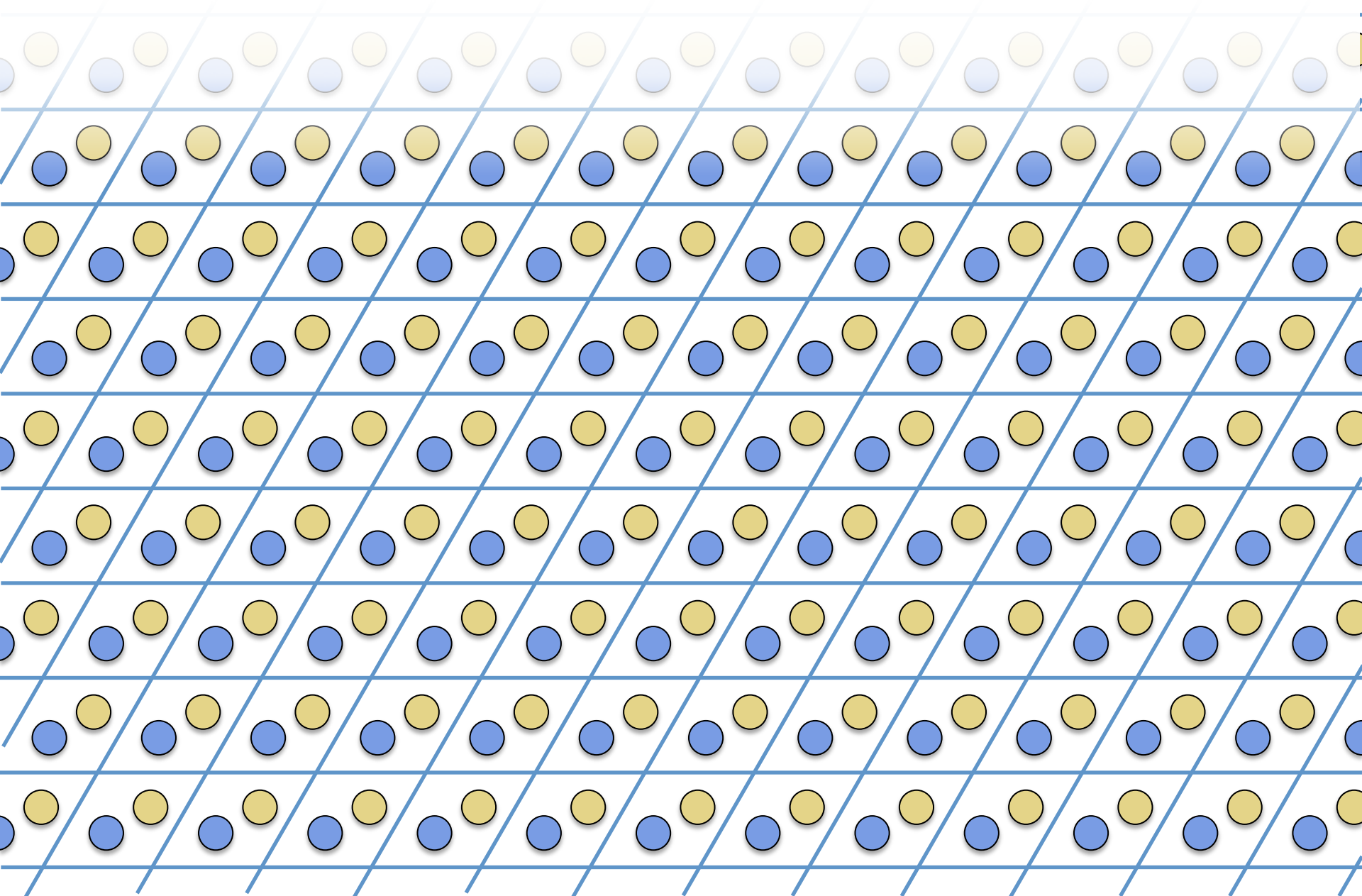
Enumerate many possible symmetrically distinct configurations of the crystal and for each one (i)

- Calculate energy $E^{(i)}$
- Calculate symmetry-invariant correlation vector $\vec{\Gamma}^{(i)}$ (the configuration's "fingerprint")

Fit the linear functional $H\left(\vec{\Gamma}^{(i)}\right) = \vec{J}^T \vec{\Gamma}^{(i)}$ that is the best approximation of $E^{(i)}$

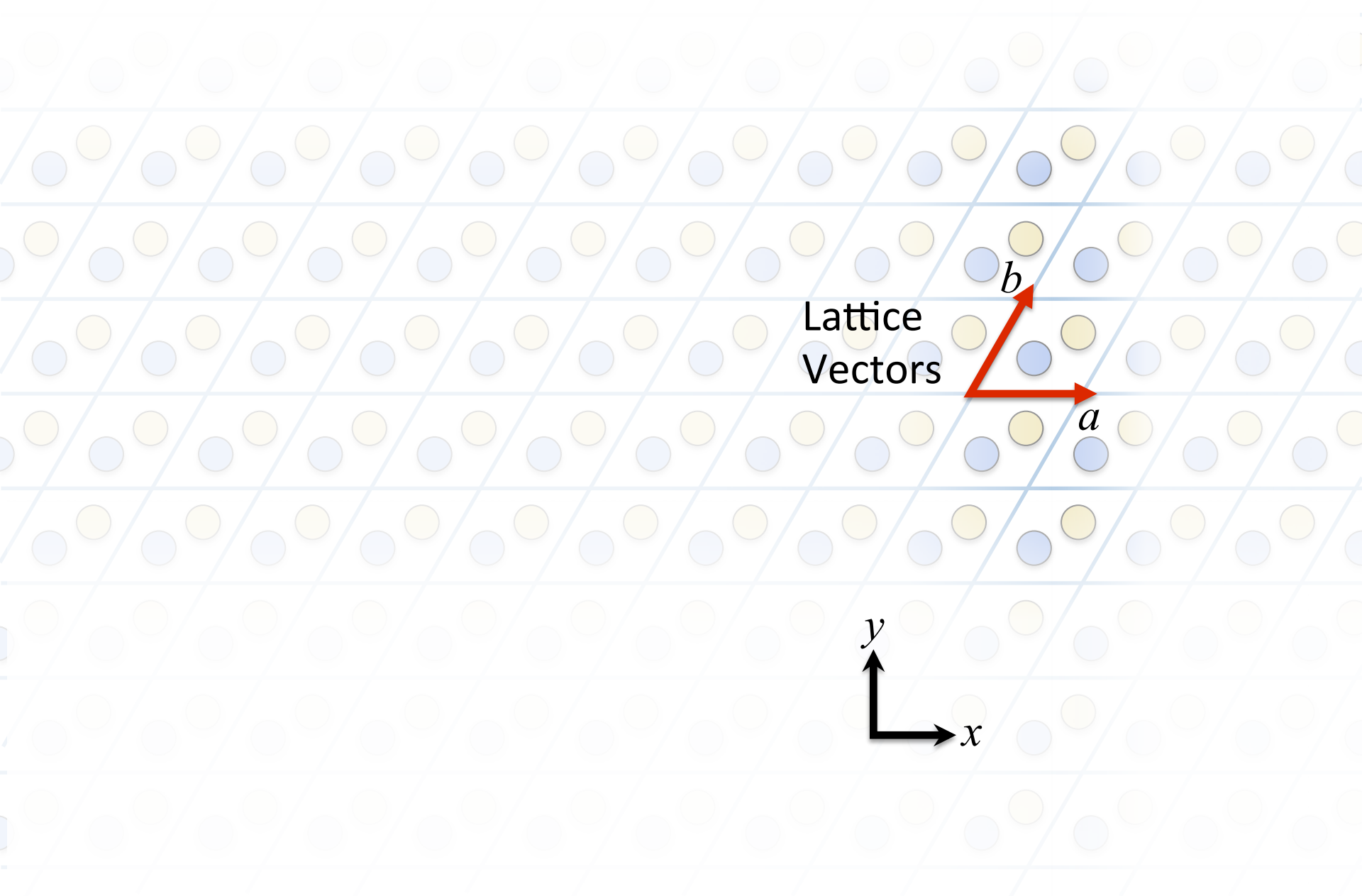
Use $H\left(\vec{\Gamma}^{(i)}\right)$ to run Monte Carlo simulations and calculate finite-temperature properties

Problem Definition

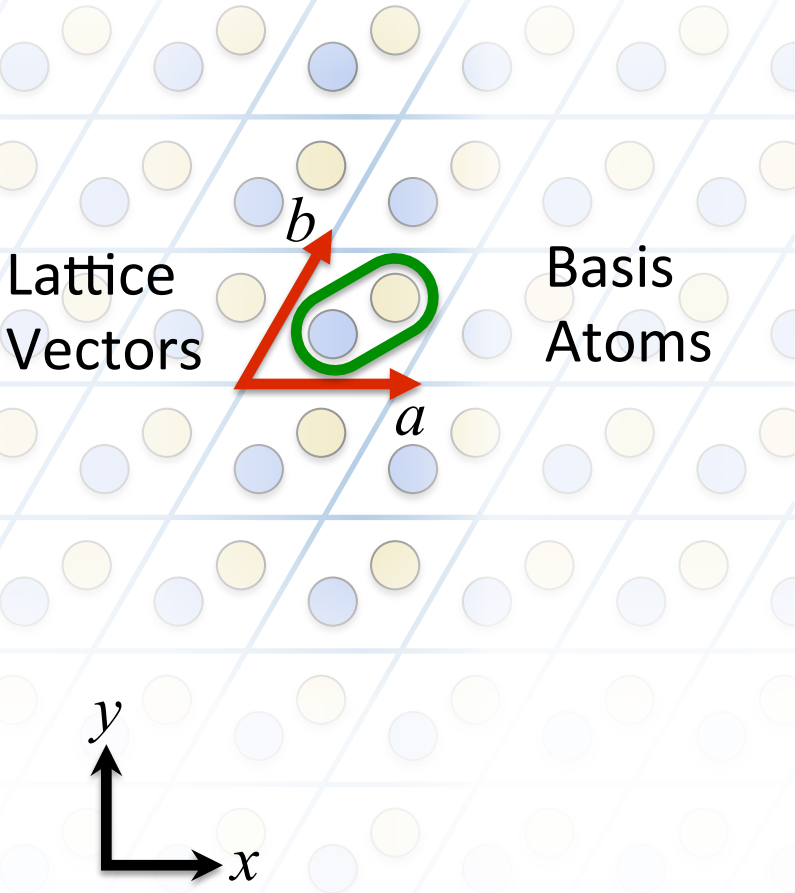


Problem Definition

Problem Definition



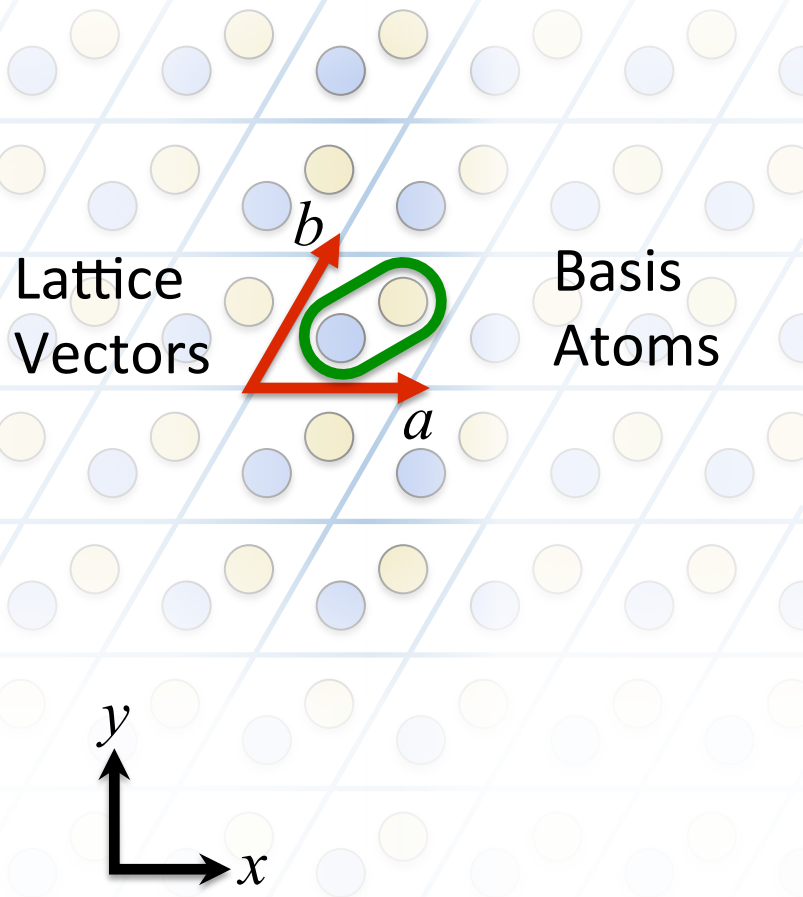
Problem Definition



Problem Definition

prim.json:

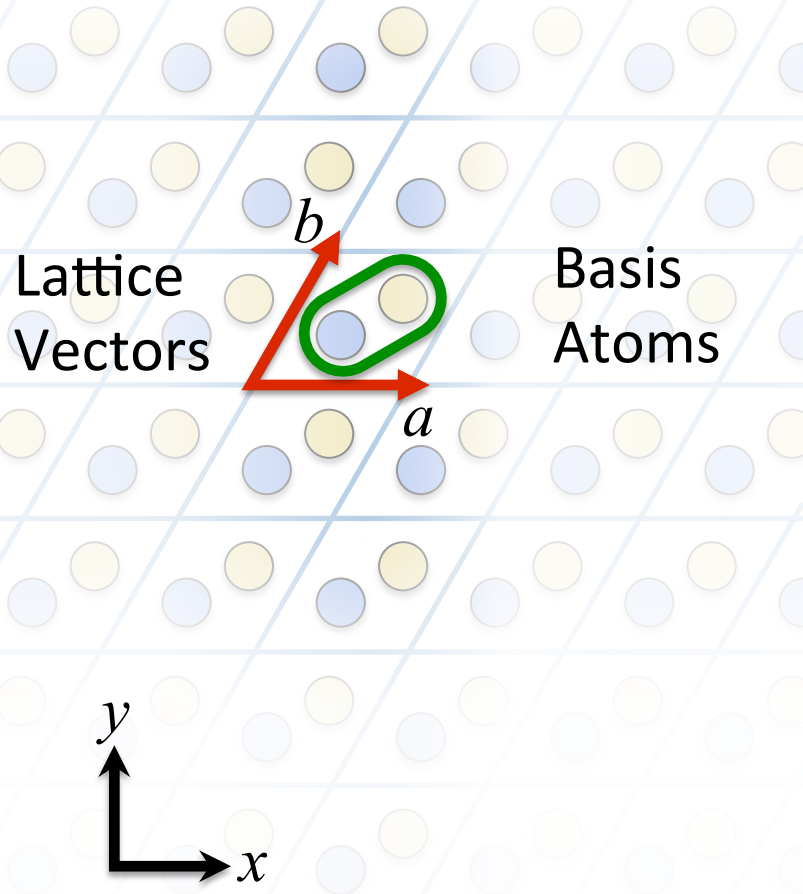
```
{
  "title" : "ZrO",
  "coordinate_mode" : "Fractional",
  "description" : "HCP Zr with oct (O) ",
  "lattice_vectors" : [
    [  3.233986860000, 0.000000000000, 0.000000000000 ],
    [ -1.616993430000, 2.800714770000, 0.000000000000 ],
    [ -0.000000000000, 0.000000000000, 5.168678340000 ]
  ],
  "basis" : [
    {
      "coordinate" : [ 0.0000000, 0.0000000, 0.000000 ],
      "occupant_dof" : [ "Zr" ]
    },
    {
      "coordinate" : [ 0.6666667, 0.3333334, 0.500000 ],
      "occupant_dof" : [ "Zr" ]
    },
    {
      "coordinate" : [ 0.3333333, 0.6666667, 0.250000 ],
      "occupant_dof" : [ "Va", "O" ]
    },
    {
      "coordinate" : [ 0.3333333, 0.6666667, 0.750000 ],
      "occupant_dof" : [ "Va", "O" ]
    }
  ]
}
```



Problem Definition

prim.json:

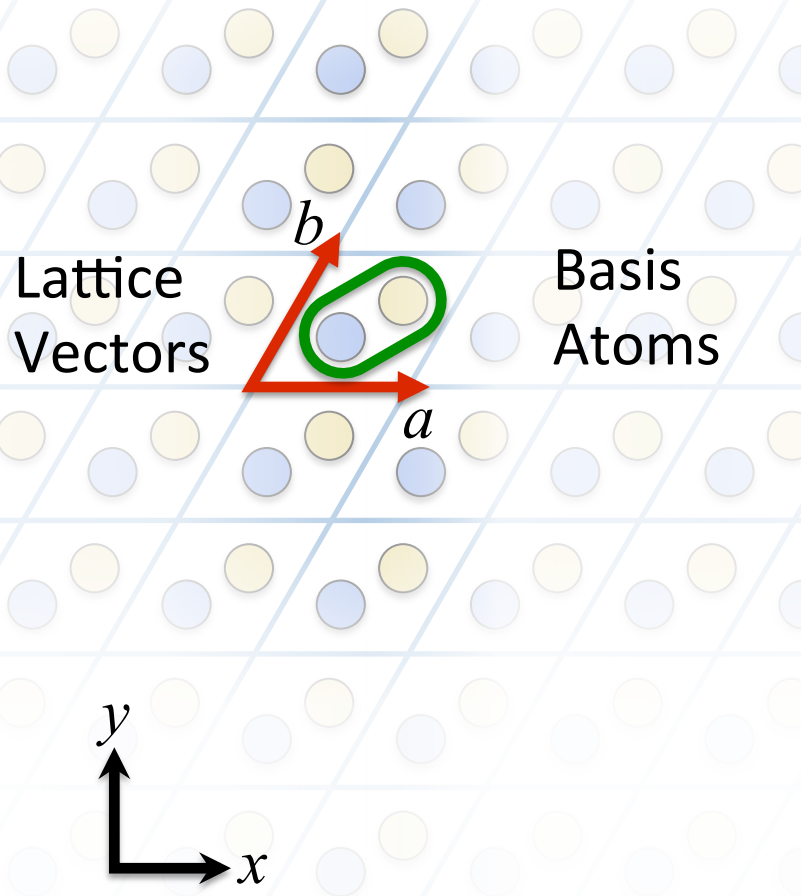
```
{
  "title" : "ZrO",
  "coordinate_mode" : "Fractional",
  "description" : "HCP Zr with oct (O) ",
  "lattice_vectors" : [
a [  3.233986860000, 0.000000000000, 0.000000000000 ],
b [ -1.616993430000, 2.800714770000, 0.000000000000 ],
c [ -0.000000000000, 0.000000000000, 5.168678340000 ]
  ],        x                y                z
  "basis" : [
    {
      "coordinate" : [ 0.0000000, 0.0000000, 0.000000 ],
      "occupant_dof" : [ "Zr" ]
    },
    {
      "coordinate" : [ 0.6666667, 0.3333334, 0.500000 ],
      "occupant_dof" : [ "Zr" ]
    },
    {
      "coordinate" : [ 0.3333333, 0.6666667, 0.250000 ],
      "occupant_dof" : [ "Va", "O" ]
    },
    {
      "coordinate" : [ 0.3333333, 0.6666667, 0.750000 ],
      "occupant_dof" : [ "Va", "O" ]
    }
  ]
}
```



Problem Definition

prim.json:

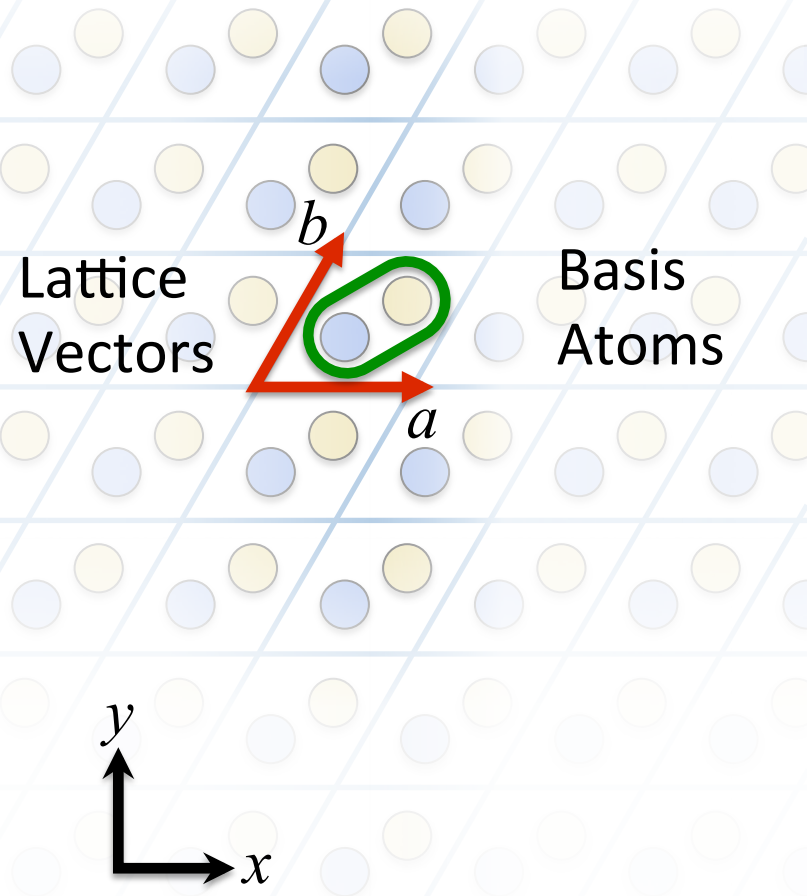
```
{
  "title" : "ZrO",
  "coordinate_mode" : "Fractional",
  "description" : "HCP Zr with oct (O) ",
  "lattice_vectors" : [
a [  3.233986860000, 0.000000000000, 0.000000000000 ],
b [ -1.616993430000, 2.800714770000, 0.000000000000 ],
c [ -0.000000000000, 0.000000000000, 5.168678340000 ]
  ],        x                y               z
  "basis" : [
    {
      "coordinate" : [ 0.0000000, 0.0000000, 0.000000 ],
      "occupant_dof" : [ "Zr" ]
    },
    {
      "coordinate" : [ 0.6666667, 0.3333334, 0.500000 ],
      "occupant_dof" : [ "Zr" ]
    },
    {
      "coordinate" : [ 0.3333333, 0.6666667, 0.250000 ],
      "occupant_dof" : [ "Va", "O" ]
    },
    {
      "coordinate" : [ 0.3333333, 0.6666667, 0.750000 ],
      "occupant_dof" : [ "Va", "O" ]
    }
  ]
}
```



Problem Definition

prim.json:

```
{
  "title" : "ZrO",
  "coordinate_mode" : "Fractional",
  "description" : "HCP Zr with oct (O) ",
  "lattice_vectors" : [
a [  3.233986860000, 0.000000000000, 0.000000000000 ],
b [ -1.616993430000, 2.800714770000, 0.000000000000 ],
c [ -0.000000000000, 0.000000000000, 5.168678340000 ]
  ],        x                y                z
  "basis" : [
    {
      "coordinate" : [ 0.0000000, 0.0000000, 0.000000 ],
      "occupant_dof" : [ "Zr" ]
    },
    {
      "coordinate" : [ 0.6666667, 0.3333334, 0.500000 ],
      "occupant_dof" : [ "Zr" ]
    },
    {
      "coordinate" : [ 0.3333333, 0.6666667, 0.250000 ],
      "occupant_dof" : [ "Va", "O" ]
    },
    {
      "coordinate" : [ 0.3333333, 0.6666667, 0.750000 ],
      "occupant_dof" : [ "Va", "O" ]
    }
  ]
}
```

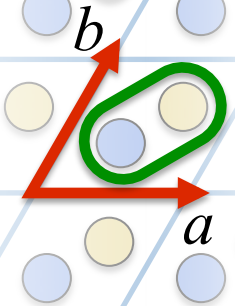


JSON: JavaScript Object Notation

Problem Definition

PRIM (legacy format):

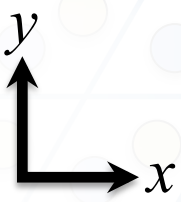
```
// HCP Zr-O
 1.00000000
a     1.61699343      2.80071477      0.00000000
b    -1.61699343      2.80071477      0.00000000
c     0.00000000      0.00000000      5.16867834
 2 2
Direct
   0.0000000   0.0000000   0.0000000 Zr
   0.6666667   0.6666667   0.5000000 Zr
   0.3333333   0.3333333   0.2500000 Va O
   0.3333333   0.3333333   0.7500000 Va O
```



Lattice Vectors

Basis Atoms

Similar format to VASP POSCAR



CASM: Getting started

Material definition

Initialization

Generated directories

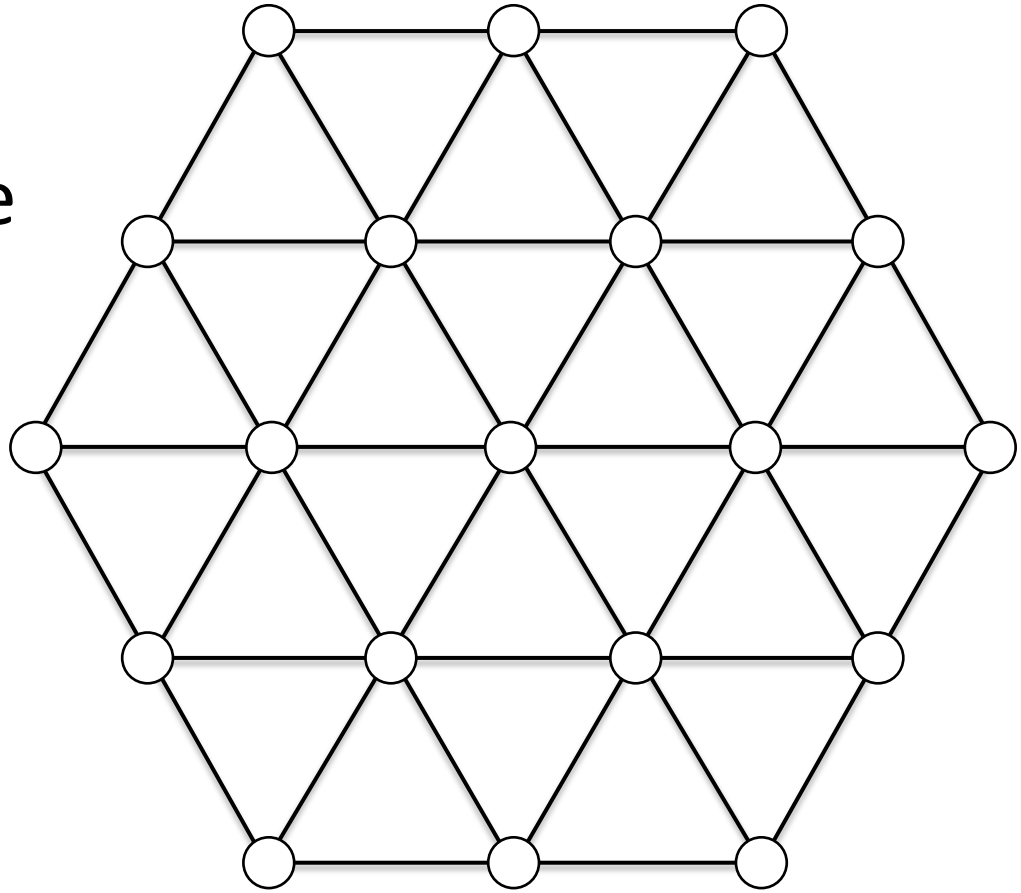
```
Adamant:~$ cd ~/Documents/ZrO
Adamant:~/Documents/ZrO$ ls
prim.json
Adamant:~/Documents/ZrO$ cat prim.json
{
  "title" : "ZrO",
  "coordinate_mode" : "Fractional",
  "description" : "HCP Zr with oct (O)",
  "lattice_vectors" : [
    [ 3.233986860000, 0.000000000000, 0.000000000000 ],
    [ -1.616993430000, 2.800714770000, 0.000000000000 ],
    [ -0.000000000000, 0.000000000000, 5.168678340000 ]
  ],
  "basis" : [
    {
      "coordinate" : [ 0.000000000000, 0.000000000000, 0.000000000000 ],
      "occupant_dof" : [ "Zr" ]
    },
    {
      "coordinate" : [ 0.666666700000, 0.333333400000, 0.500000000000 ],
      "occupant_dof" : [ "Zr" ]
    },
    {
      "coordinate" : [ 0.333333300000, 0.666666600000, 0.250000000000 ],
      "occupant_dof" : [ "Va", "O" ]
    },
    {
      "coordinate" : [ 0.333333300000, 0.666666600000, 0.750000000000 ],
      "occupant_dof" : [ "Va", "O" ]
    }
  ]
}
Adamant:~/Documents/ZrO$ casm init

**************************

Initializing CASM project 'ZrO'
  DONE

Adamant:~/Documents/ZrO$ ls
basis_sets         cluster_expansions prim.json          symmetry           training_data
Adamant:~/Documents/ZrO$
```

CASM: Getting started

- Install 'casm' command line programs
- 'casm status'
  - Quick look at your current project
  - Suggestions for next steps
- 'casm format'
  - Documentation of project directory and file formats
- 'casm –h'
  - Help menus show allowed options

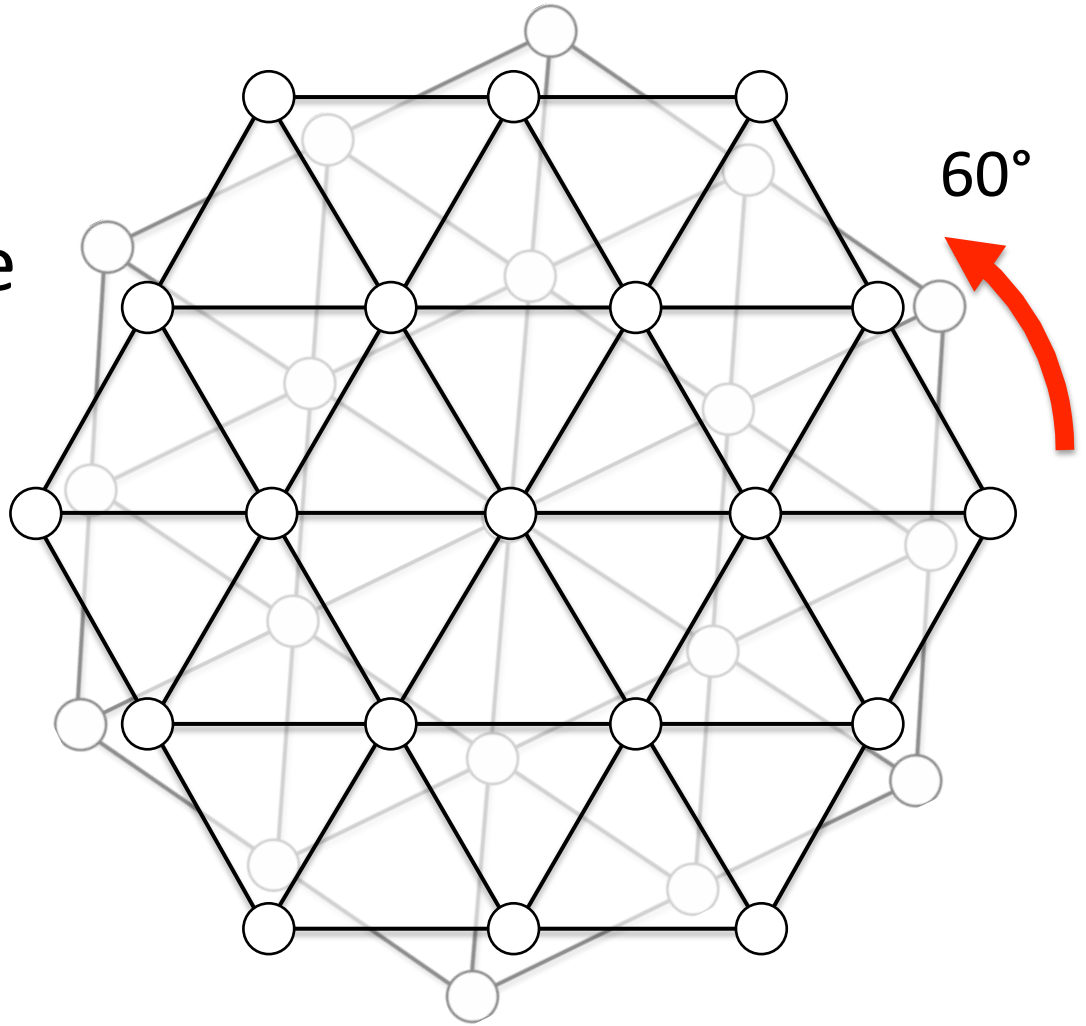
Crystal Symmetry

Apply rotation, reflection, inversion, and/or rotation to the crystal.



Crystal Symmetry

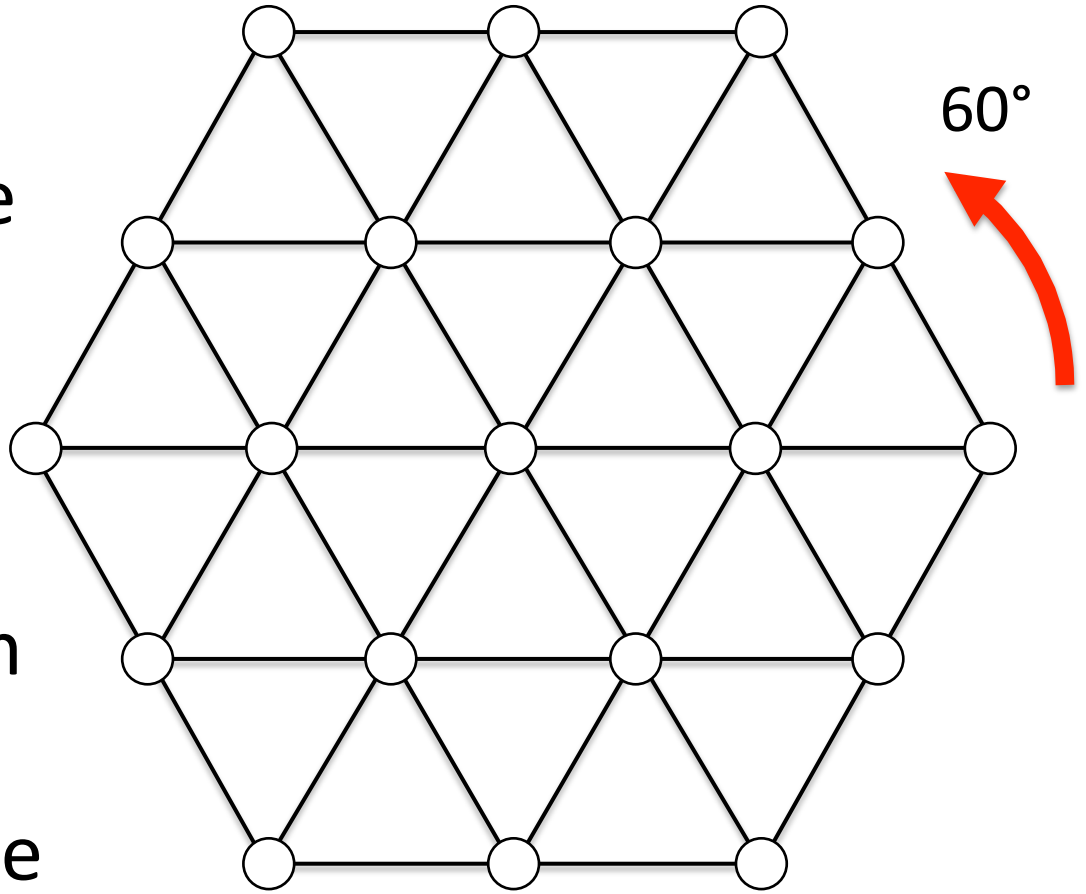
Apply rotation, reflection, inversion, and/or rotation to the crystal.



Crystal Symmetry

Apply rotation, reflection, inversion, and/or rotation to the crystal.

If the result is indistinguishable from the initial state, then that operation is in the crystal's space group.



Crystal Symmetry

Common notation: $\{R_\alpha|\tau\}$

- R_α: point group operations such as rotations, reflections, improper rotations, inversions

$$r' = \{R_\alpha|\tau\}r = \alpha r' + \tau$$

- τ: translation vector
- α: 3x3 transformation matrix

M.S. Dresselhaus, G. Dresselhaus, and A. Jorio.
Group Theory: Application to the Physics of Condensed Matter.

Symmetry

Lattice point group:

Symmetry operations that map lattice onto itself, leaving origin invariant (rotation, mirror, inversion, rotoinversion)

Crystal factor group:

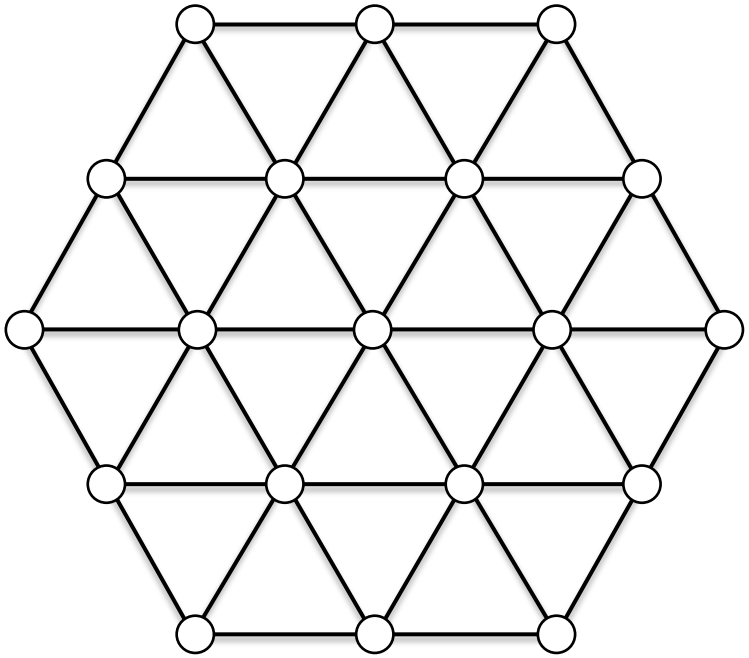
Symmetry operations that map lattice and identical basis sites onto themselves (translations may result in screw axes or glide planes)

Crystal point group:

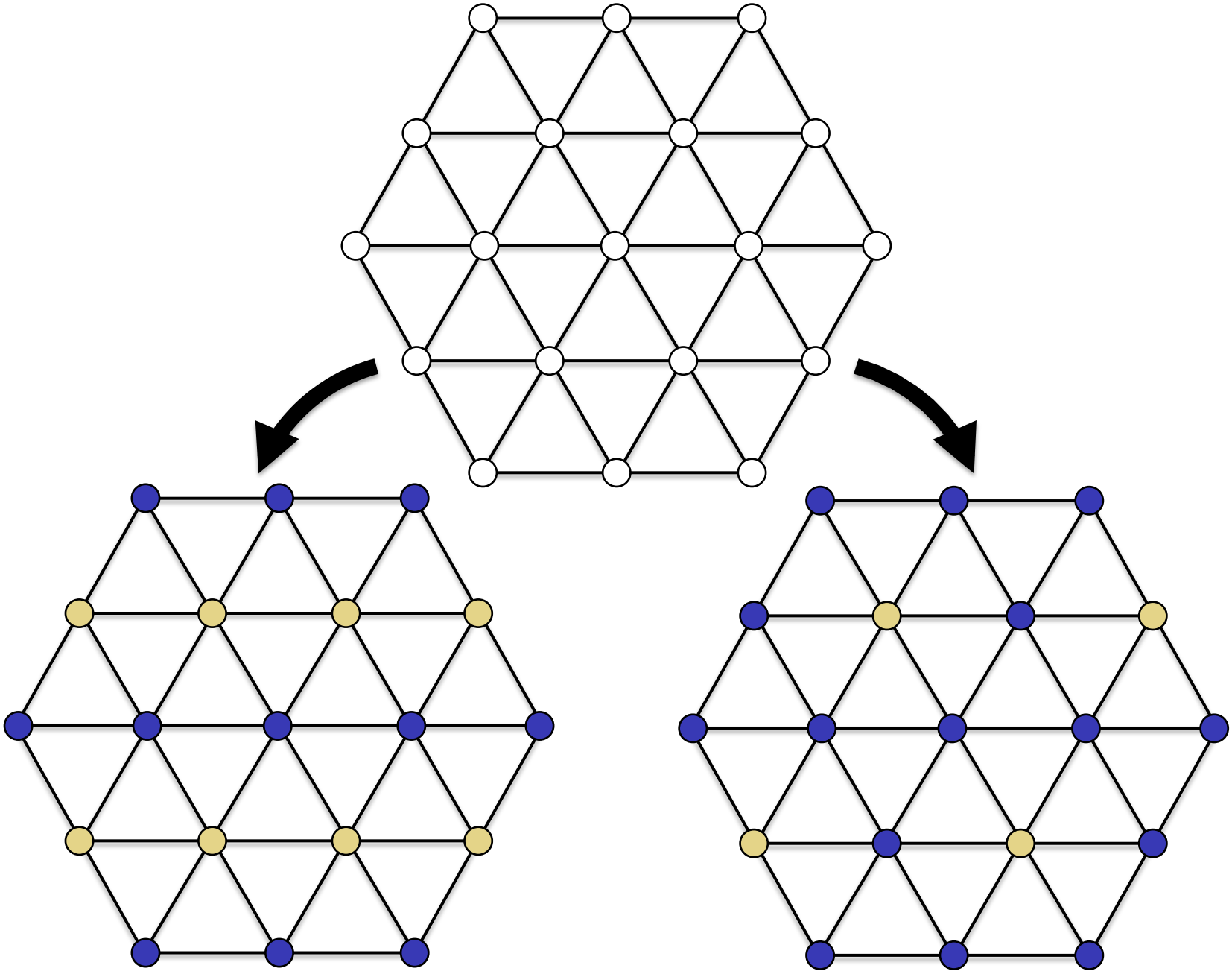
Crystal factor group operations, with translations removed

```
Adamant:~/Documents/ZrO$ ls symmetry
crystal_point_group.json
factor_group.json
lattice_point_group.json
Adamant:~/Documents/ZrO$ casm sym
Generating lattice point group.

  Lattice point group size: 24
  Lattice point group is: D6h

Generating factor group.

  Factor group size: 24
  Crystal point group is: D6h

Adamant:~/Documents/ZrO$
```

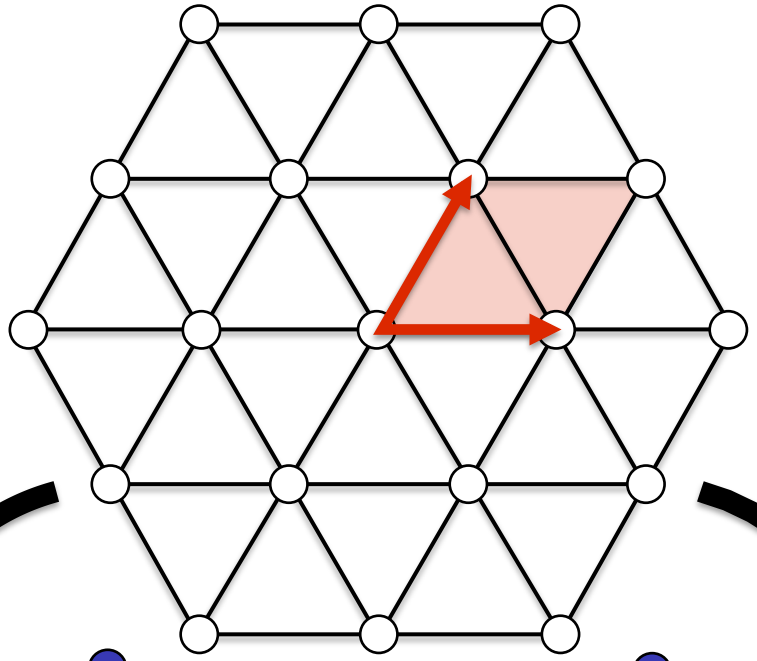
Crystal Configurations



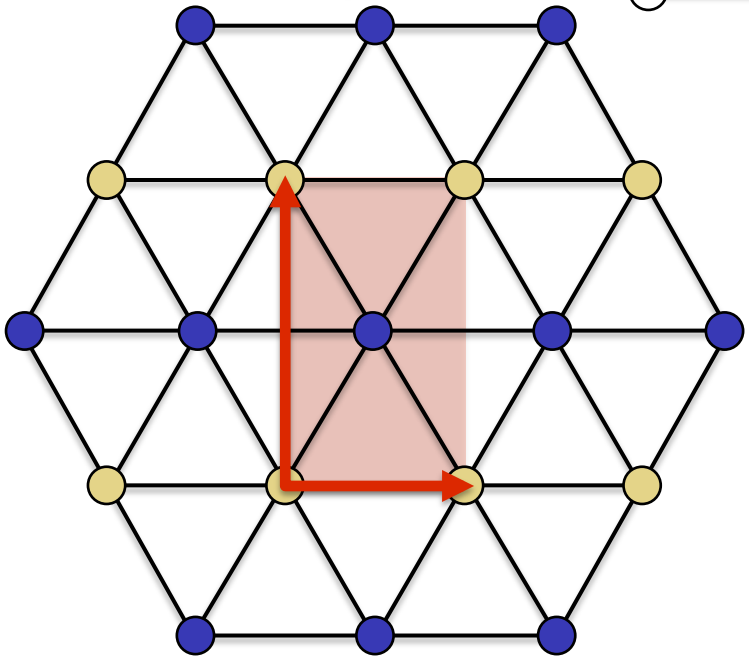
Crystal Configurations



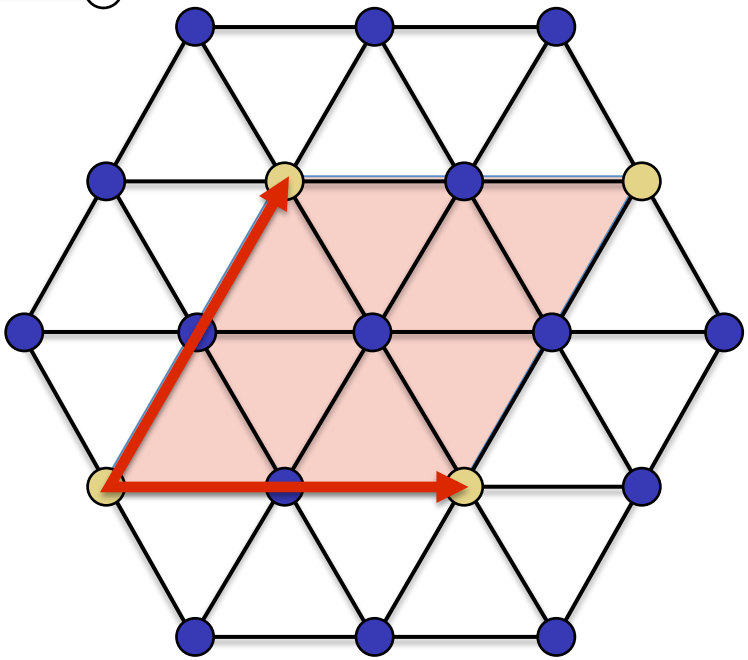
Crystal Configurations



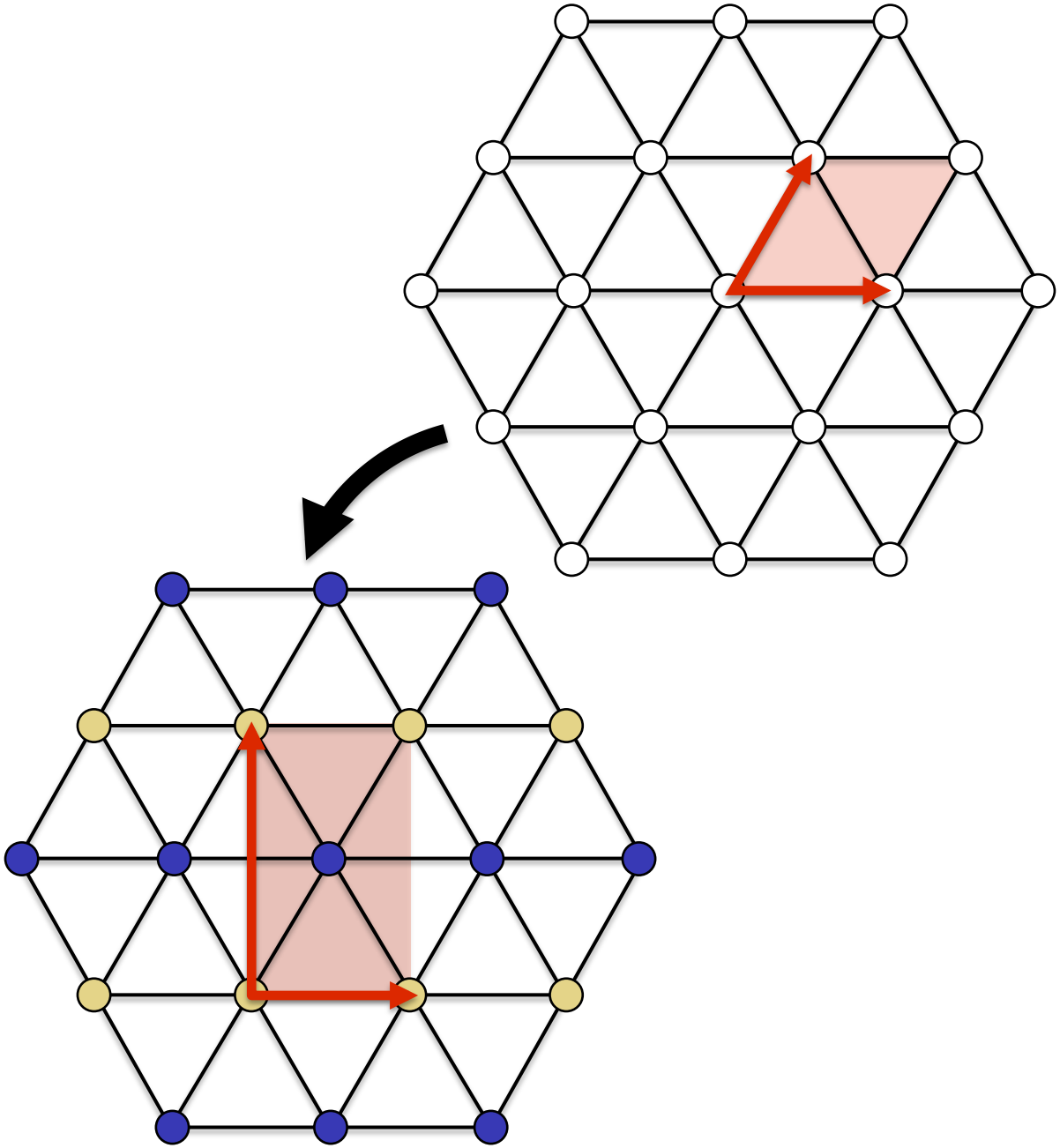
Vol 2 Supercell



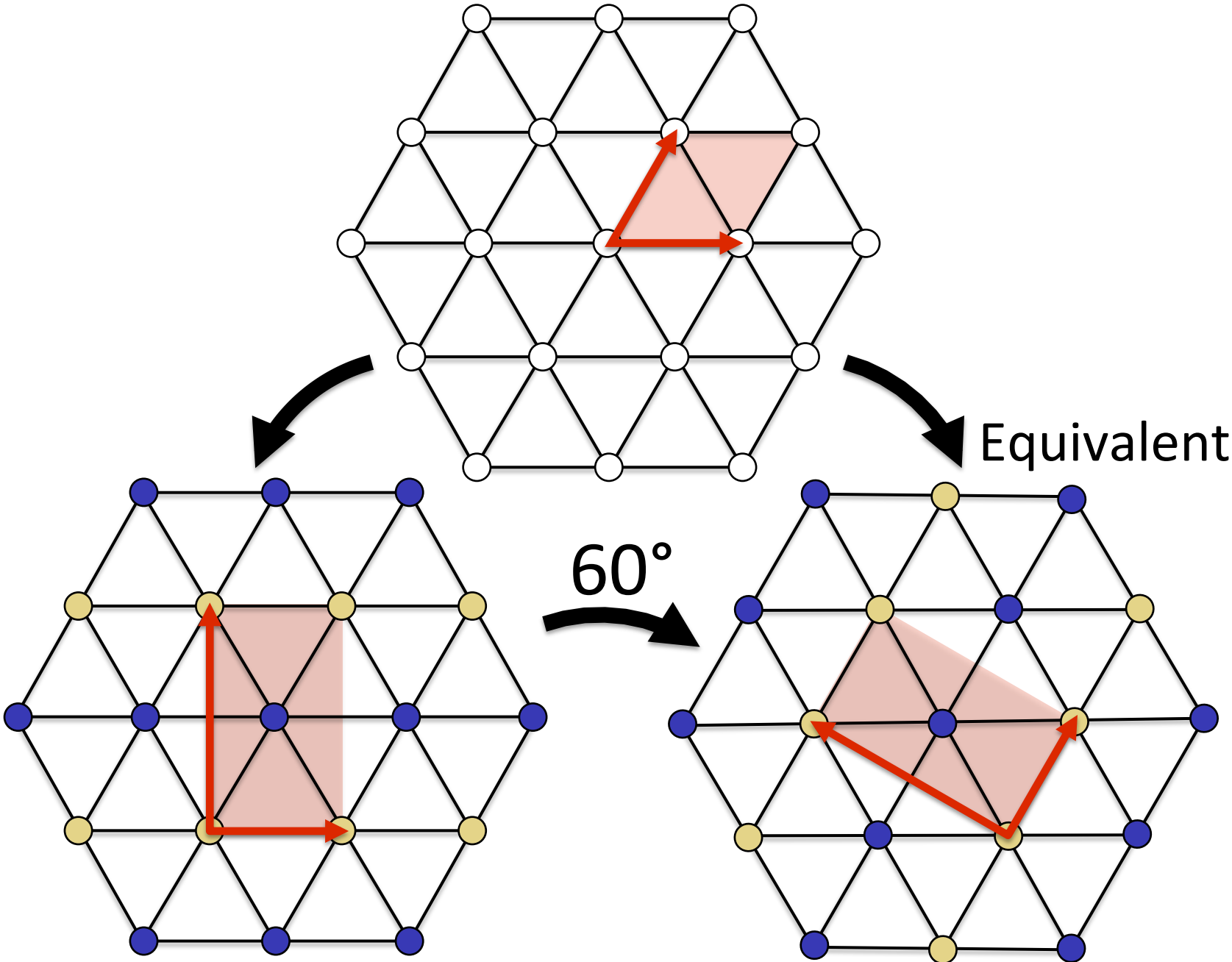
Vol 4 Supercell



Crystal Configurations

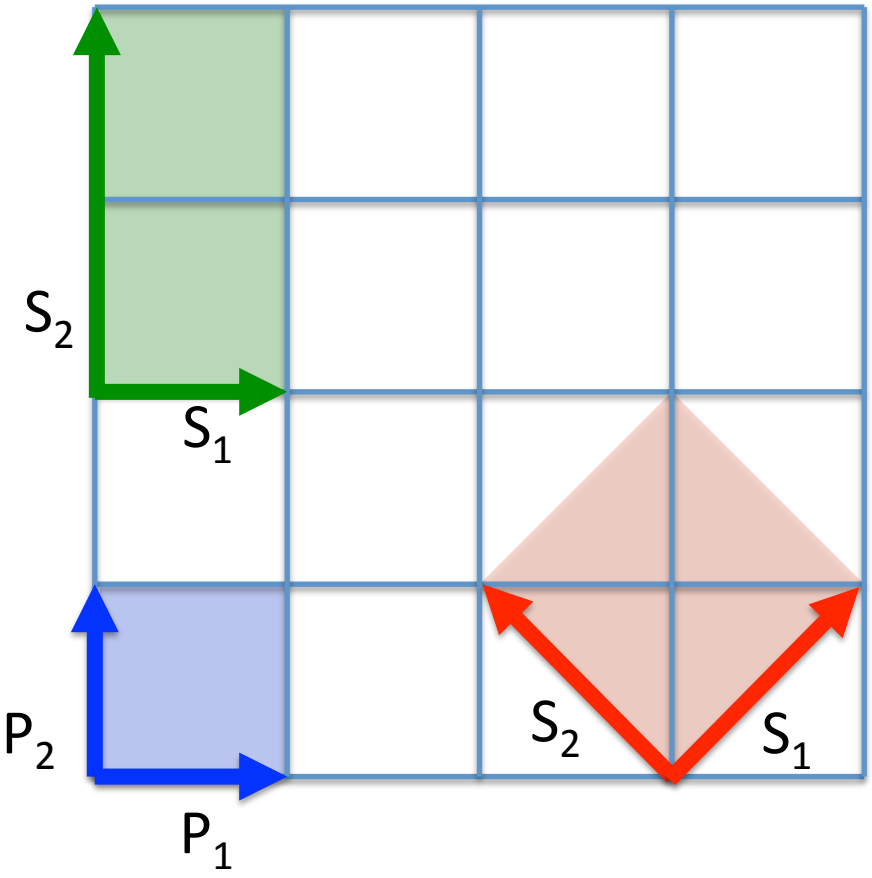


Crystal Configurations



Supercell Enumeration

$S_1=P_1$
$S_2=2P_2$



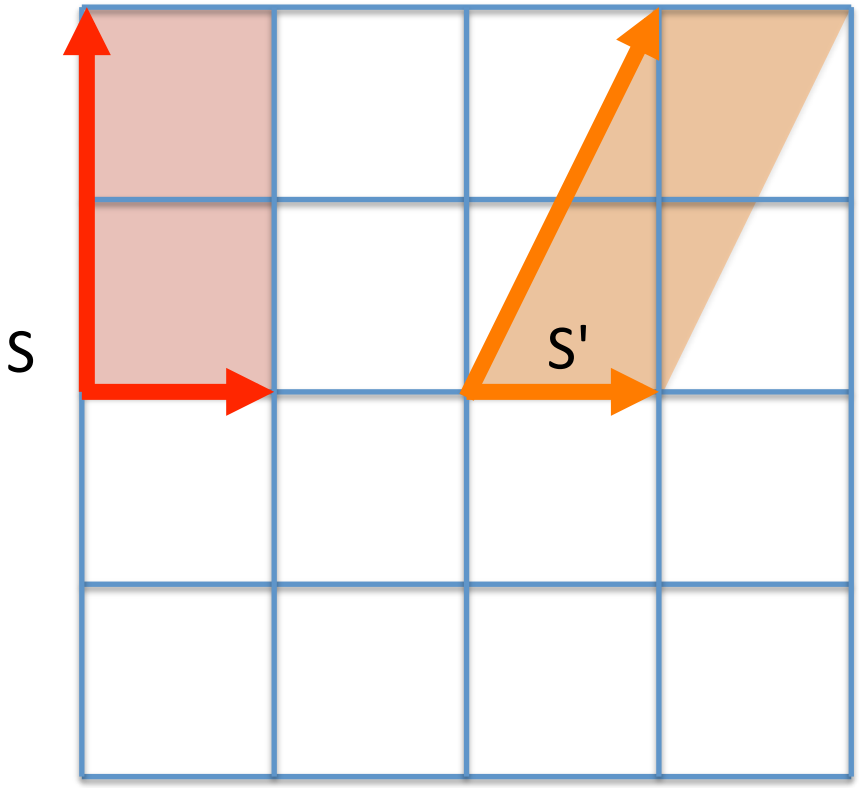
Represent lattice using column vector matrices:

$$S = PT$$

- P: primitive lattice
- S: supercell lattice
- T: integer supercell transformation matrix

$S_1=P_1+P_2$
$S_2=-P_1+P_2$

Supercell Enumeration

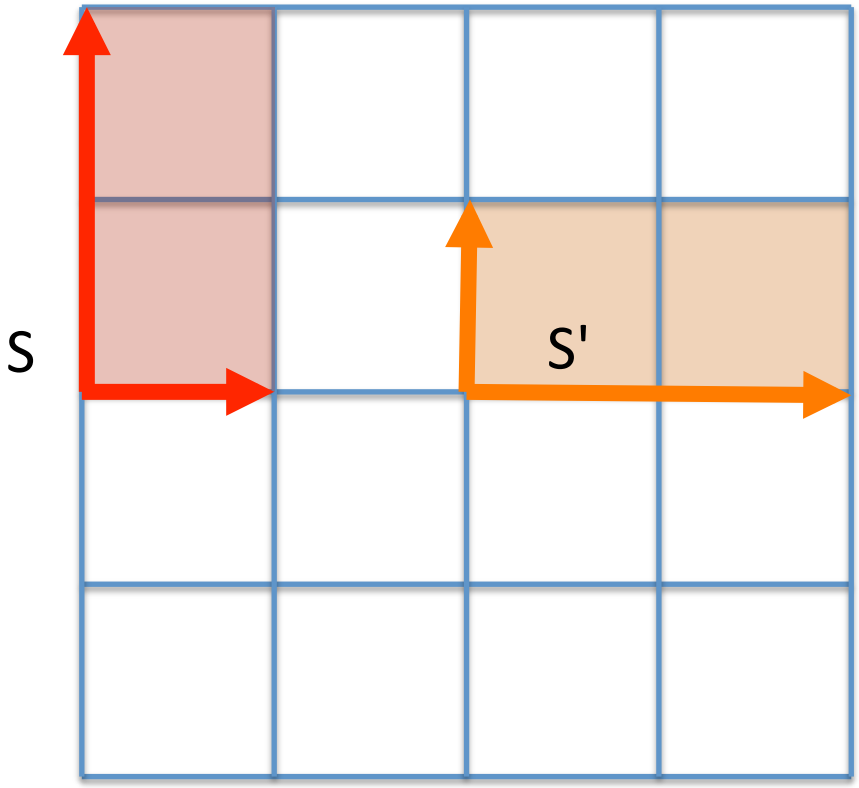


Equivalent supercells:

$$S' = ST$$

- Linear combination of supercell lattice vectors

Supercell Enumeration



Equivalent supercells:

$$S' = (R_\alpha S)T$$

- Linear combination of symmetrically equivalent supercell lattice vectors
- For a crystal, allowed symmetry operations depends on crystal basis

Supercell enumeration

If unit cell lattice, P, is represented as column vector matrix, then supercell lattices, S, are:

$$S = PT$$

T is an integer supercell transformation matrix, and S has integer volume $\det(T)$, relative to the primitive cell.

Supercells are equivalent if for some T:

$$S' = (R_\alpha S)T$$

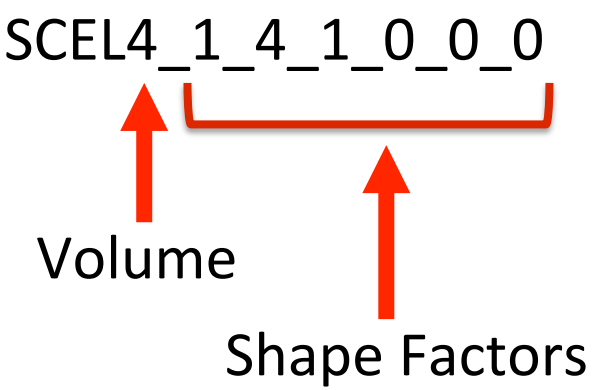
At a given volume ($\det(T)$), loop over possible T matrices and use crystal point group operations to check if we've found a new symmetrically distinct supercell

Supercell enumeration

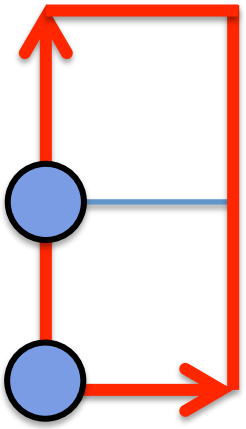
Max an min supercell volume can be specified for enumeration

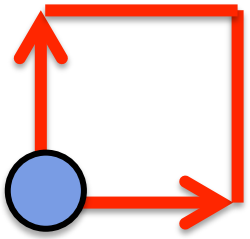
Supercell enumeration can be restricted to 1D and 2D. See `casm enum --help` for details.

```
Adamant:~/Documents/ZrO$ casm enum --supercells --max 4
-- Construct: CASM Project --
from: "/Users/johnct/Documents/ZrO"

-- Load project data --
read: "/Users/johnct/Documents/ZrO/.casm/composition_axes.json"


**************************

Generating supercells from 1 to 4
Lattice vectors  b and c will be used. (supercells will be 3-dimensional)

  Generated: SCEL1_1_1_1_0_0_0
  Generated: SCEL2_1_2_1_0_0_0
  Generated: SCEL2_1_2_1_1_0_0
  Generated: SCEL2_1_1_2_0_0_0
  Generated: SCEL3_3_1_1_0_0_1
  Generated: SCEL3_3_1_1_0_0_2
  Generated: SCEL3_1_3_1_2_0_0
  Generated: SCEL3_3_1_1_0_2_2
  Generated: SCEL3_1_1_3_0_0_0
  Generated: SCEL4_1_4_1_0_0_0
  Generated: SCEL4_2_2_1_0_0_1
  Generated: SCEL4_1_4_1_1_0_0
  Generated: SCEL4_4_1_1_0_3_2
  Generated: SCEL4_1_4_1_2_0_0
  Generated: SCEL4_2_2_1_0_1_1
  Generated: SCEL4_2_2_1_0_0_0
  Generated: SCEL4_2_2_1_0_1_0
  Generated: SCEL4_1_2_2_0_0_0
  Generated: SCEL4_1_2_2_1_0_0
  Generated: SCEL4_1_1_4_0_0_0

  DONE.

Write SCEL.

Writing config_list...
  DONE

Adamant:~/Documents/ZrO$
```



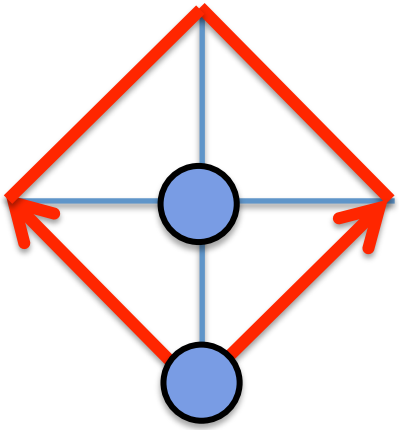
Configuration enumeration





A:

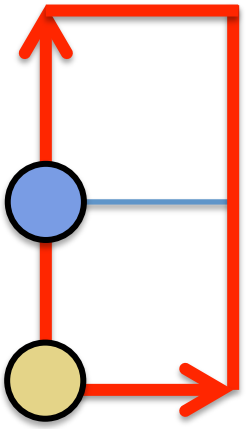
B:

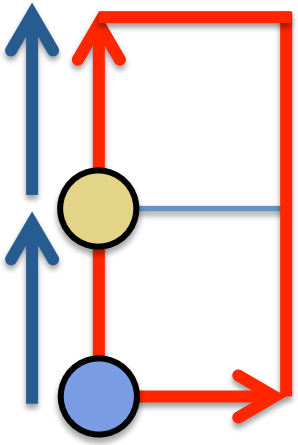


Equivalent configurations:

- Supercells of the volume 1 configuration

Configuration enumeration





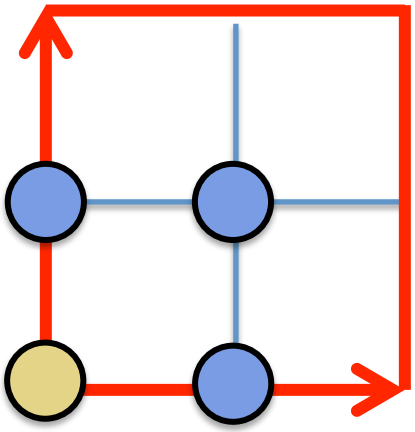
A:

B:

Equivalent configurations:

- Translation maps one configuration onto the other

Configuration enumeration

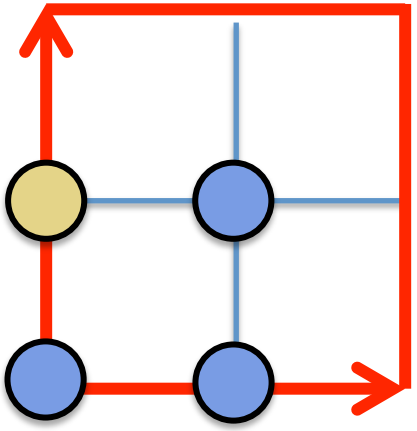


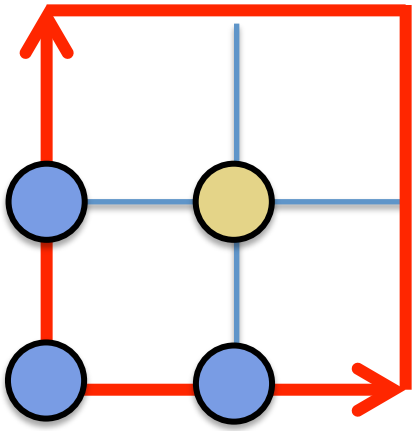
A:

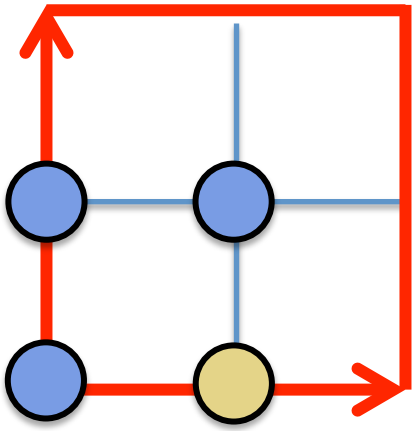
B:

Equivalent configurations:

- Rotation or translation maps occupants







Configuration enumeration

Enumerate symmetrically unique configurations by:

- Loop over possible occupations
- Check all factor group operations that keep the supercell lattice vectors unchanged, in combination with all primitive cell translations within the supercell
- If translation within the supercell alone maps all occupants, then it is a supercell of a smaller primitive configuration

Configuration enumeration

Configurations are enumerated ONLY in supercells that have been enumerated.

You can specify the min and max supercell volumes, or specify a supercell explicitly.

CAUTION: The number of unique configurations that fit in a supercell grows exponentially with supercell volume.

```
Adamant:~/Documents/ZrO$ casm enum -c --all
-- Construct: CASM Project --
from: "/Users/johnct/Documents/ZrO"

-- Load project data --
read: "/Users/johnct/Documents/ZrO/.casm/composition_axes.json"
read: "/Users/johnct/Documents/ZrO/training_data/SCEL"
read: "/Users/johnct/Documents/ZrO/.casm/config_list.json"


**************************

Enumerate all configurations

  Enumerate configurations for SCEL1_1_1_1_0_0_0 ...  3 configs.
  Enumerate configurations for SCEL2_1_2_1_0_0_0 ...  4 configs.
  Enumerate configurations for SCEL2_1_2_1_1_0_0 ...  3 configs.
  Enumerate configurations for SCEL2_1_1_2_0_0_0 ...  3 configs.
  Enumerate configurations for SCEL3_3_1_1_0_0_1 ...  10 configs.
  Enumerate configurations for SCEL3_3_1_1_0_0_2 ...  10 configs.
  Enumerate configurations for SCEL3_1_3_1_2_0_0 ...  16 configs.
  Enumerate configurations for SCEL3_3_1_1_0_2_2 ...  10 configs.
  Enumerate configurations for SCEL3_1_1_3_0_0_0 ...  10 configs.
  Enumerate configurations for SCEL4_1_4_1_0_0_0 ...  27 configs.
  Enumerate configurations for SCEL4_2_2_1_0_0_1 ...  27 configs.
  Enumerate configurations for SCEL4_1_4_1_1_0_0 ...  42 configs.
  Enumerate configurations for SCEL4_4_1_1_0_3_2 ...  24 configs.
  Enumerate configurations for SCEL4_1_4_1_2_0_0 ...  24 configs.
  Enumerate configurations for SCEL4_2_2_1_0_1_1 ...  21 configs.
  Enumerate configurations for SCEL4_2_2_1_0_0_0 ...  15 configs.
  Enumerate configurations for SCEL4_2_2_1_0_1_0 ...  18 configs.
  Enumerate configurations for SCEL4_1_2_2_0_0_0 ...  21 configs.
  Enumerate configurations for SCEL4_1_2_2_1_0_0 ...  24 configs.
  Enumerate configurations for SCEL4_1_1_4_0_0_0 ...  24 configs.
  DONE.

Writing config_list...
  DONE

Adamant:~/Documents/ZrO$
```

Configuration visualization

Set up environment:

```
casm settings --set-view-command 'casm.view VESTA'
```

Try with one or more configuration names:

```
casm view SCEL4_2_2_1_0_0_1/25
```

CASM Composition metrics

Each configuration has a *composition*, and there are various ways to measure it:

atom_frac: atom_frac(Au), atom_frac(Cu), etc.

- Number of Au, Cu, etc., divided by total number of atoms. Vacancies are ignored.

comp_n: comp_n(Au), comp_n(Cu), etc.

- Number of Au, Cu, etc., divided by number of primitive cells (volume). Accounts for vacancies.

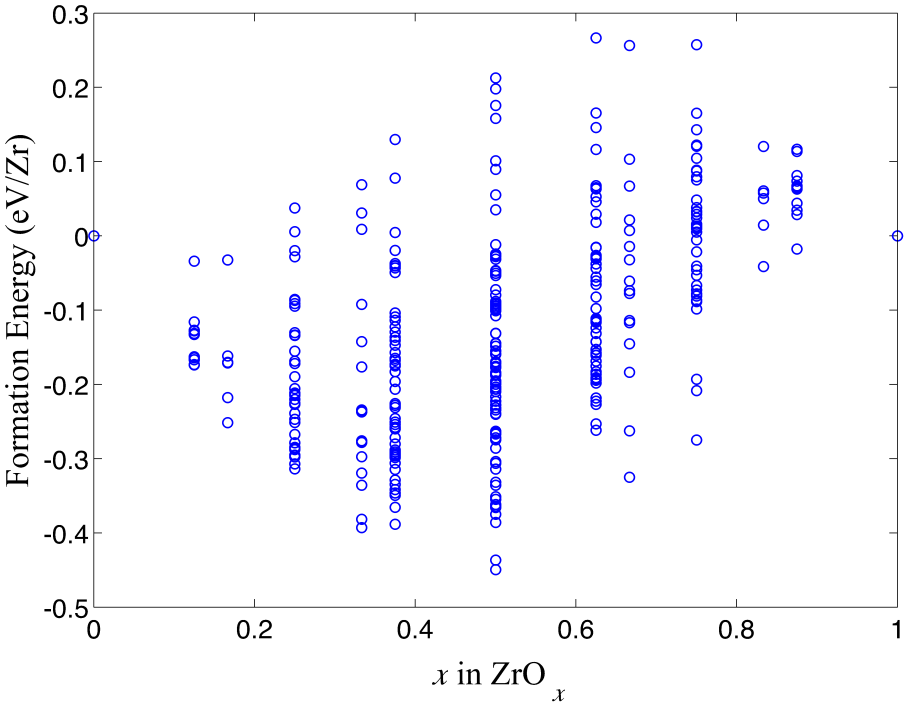
comp: comp(a), comp(b), etc.

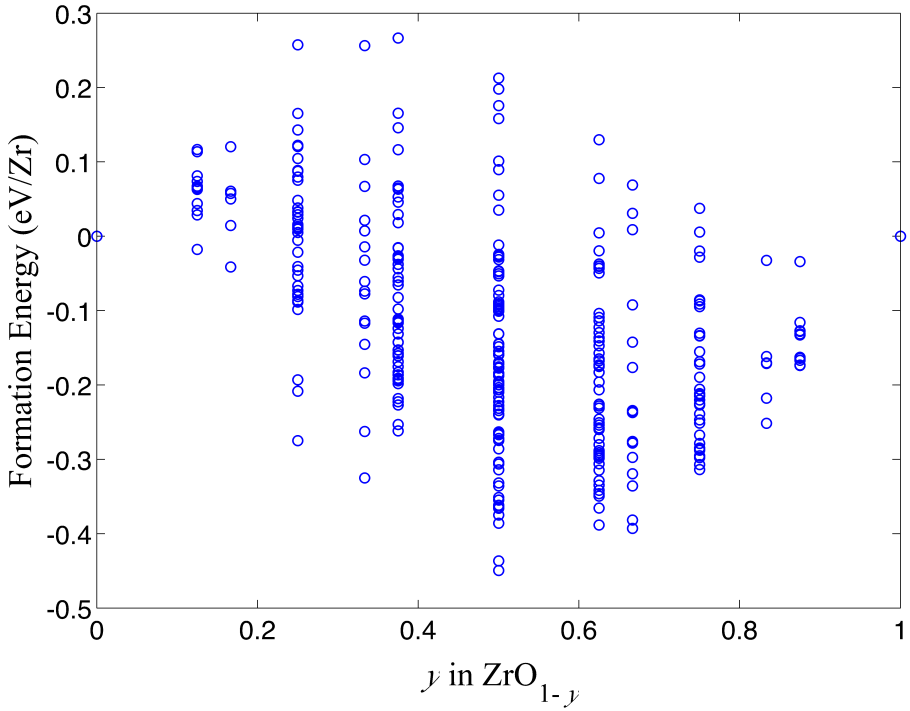
- Primitive-cell-based formula parameters (e.g., $Au_aCu_bPt_{(1-a-b)}$), determined by user-specified composition axes. Accounts for vacancies.

When vacancies are present, 'atom_frac' is qualitatively different from 'comp_n' and 'comp'

Specifying composition axes

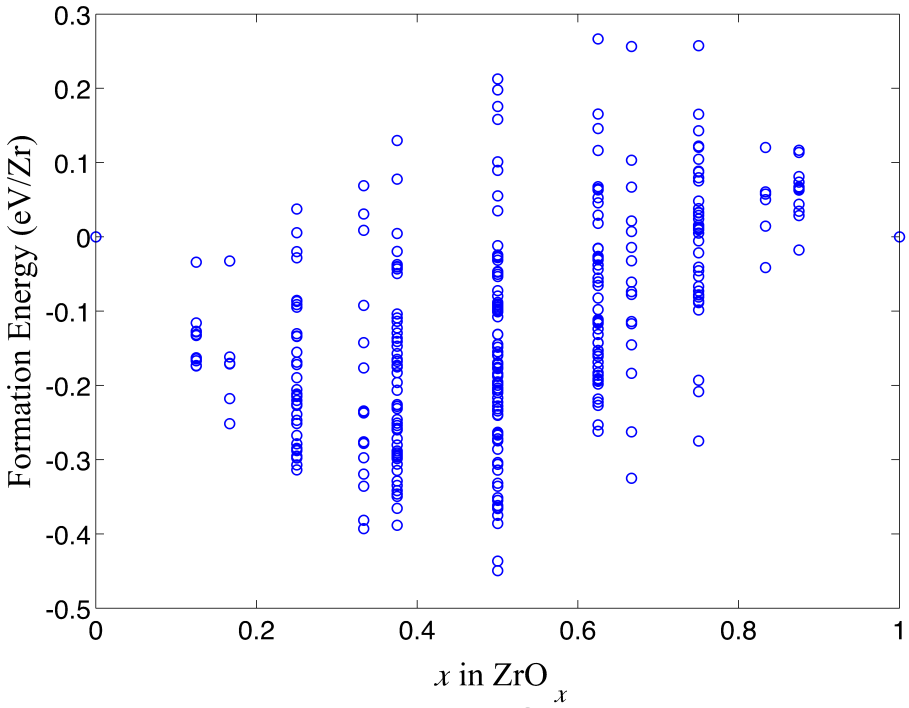
Binary Compound:

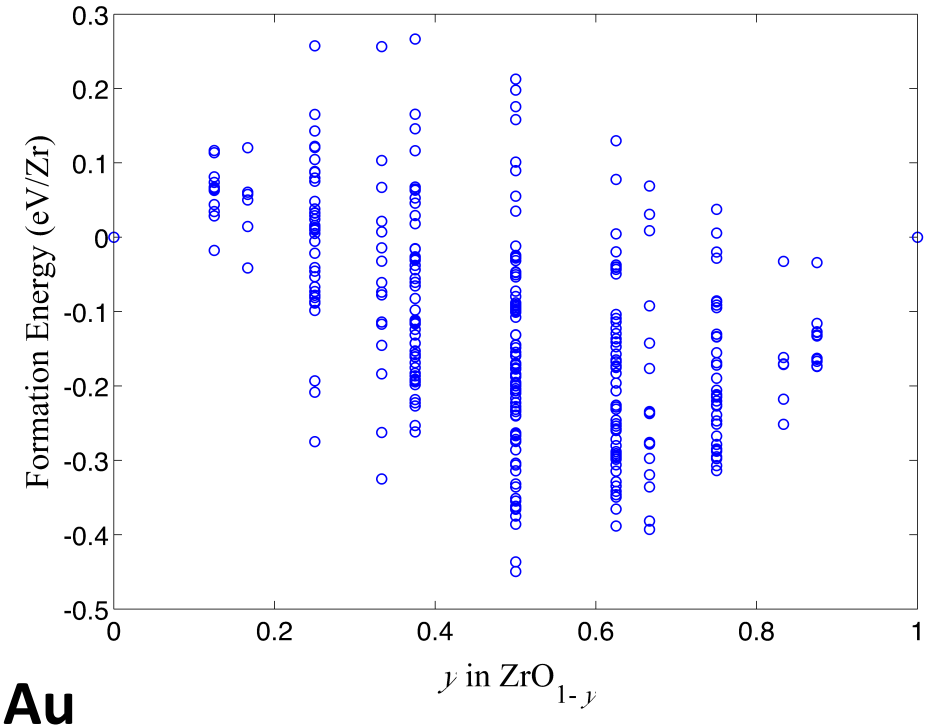




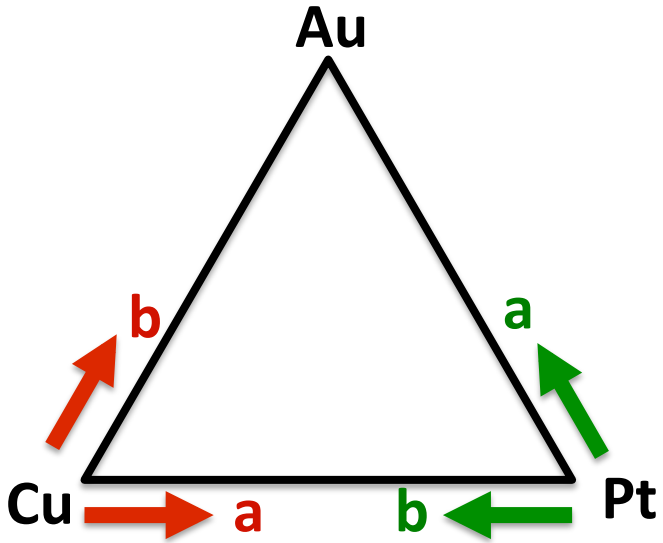
Specifying composition axes

Binary Compound:





Ternary Compound:

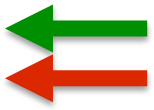


$Cu_{1-a-b}Au_bPt_a$

$Cu_bAu_aPt_{1-a-b}$

Specifying composition axes

```
Adamant:~/Documents/ZrO$ casm composition -d
-- Construct: CASM Project --
from: "/Users/johnct/Documents/ZrO"

-- Load project data --
read: "/Users/johnct/Documents/ZrO/.casm/composition_axes.json"


*****************************

Standard composition axes:

       KEY     ORIGIN          a     GENERAL FORMULA
       ---        ---        ---     ---
         0 Zr(2)Va(2)  Zr(2)O(2)     Zr(2)Va(2-2a)O(2a)
         1  Zr(2)O(2) Zr(2)Va(2)     Zr(2)Va(2a)O(2-2a)


Please use 'casm composition --select' to choose your composition axes.

Adamant:~/Documents/ZrO$
```



$Zr_2O_{2a}[Va]_{2(1-a)}$

$Zr_2O_{2(1-a)}[Va]_{2a}$

NOTE: `prim.json` determines formula unit (2 Zr)

Data Management

Project data is managed according to the parameters, settings, and timeline that it depends on:

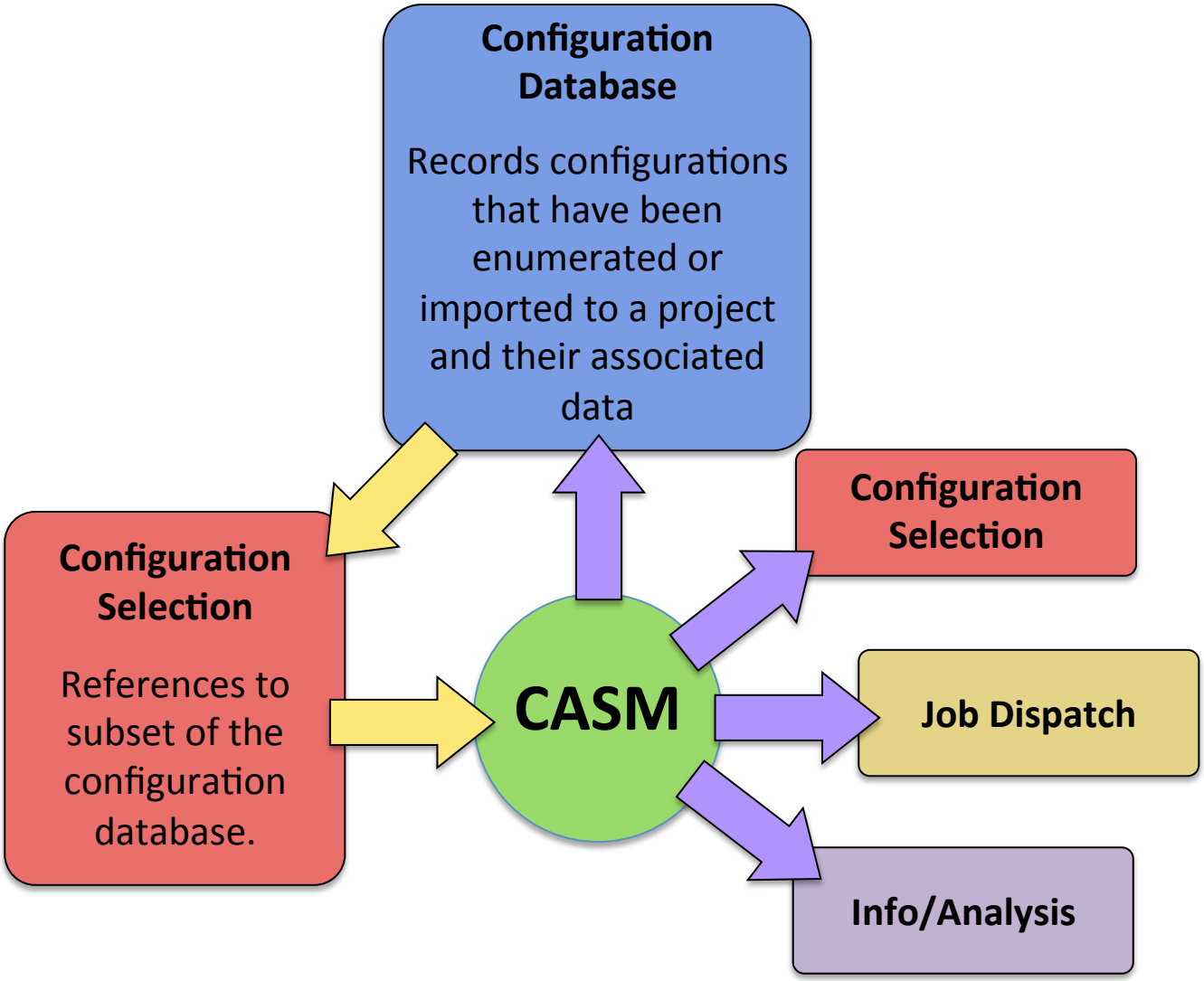
'calctype'	DFT parameters and setting (XC functional, pseudopotential, convergence parameters, etc.)
'property'	The particular quantity being measured (formation_energy, volume_relaxation, etc.)
'ref'	Baseline reference used for measuring quantity (zero-point of energy, reference volume, etc.)
'bset'	Basis functions used to "fingerprint" configurations and in terms of which the property model is expressed .
'eci'	The model parameters (**e**ffective **c**luster **i**nteractions) used to predict a particular property in terms of a particular basis set.

Manage and create settings profiles from the 'casm settings' command. A 'default' profile is created during project initialized.

Ex:

- DFT-calculated properties: Depend on 'calctype', 'property', and sometimes 'ref'
- Cluster expansion/physics model: Depends on 'calctype', 'property', 'ref', 'bset'
- Predicted properties and Monte Carlo Data: Depend on 'calctype', 'property', 'ref', 'bset', and 'eci'

Data Management Workflow



Working with selections

CASM query syntax provides simple interface to explore, screen, and select configurations.

Example:

```
casm select --set "eq(scel_size, 4)"
```

will select all configurations the belong to supercells of volume 4 (relative to primitive cell)

Many configuration properties are supported

- 'scel_size' — Integer volume of supercell that contains the configuration
- 'comp(a)', 'comp(b)', etc. — The chemical composition, in terms of composition axes
- 'atom_frac(Au)', etc. — The atomic fraction of a particular specie
- 'is_calculated' — True if DFT data exists for the configuration
- 'formation_energy' — DFT formation energy (if calculated)
- 'volume relaxation — Change in volume due to relaxation (expressed as V/V_0)

… and many others (for full listing see `casm select --help properties`)

Properties can be related and compared via operators

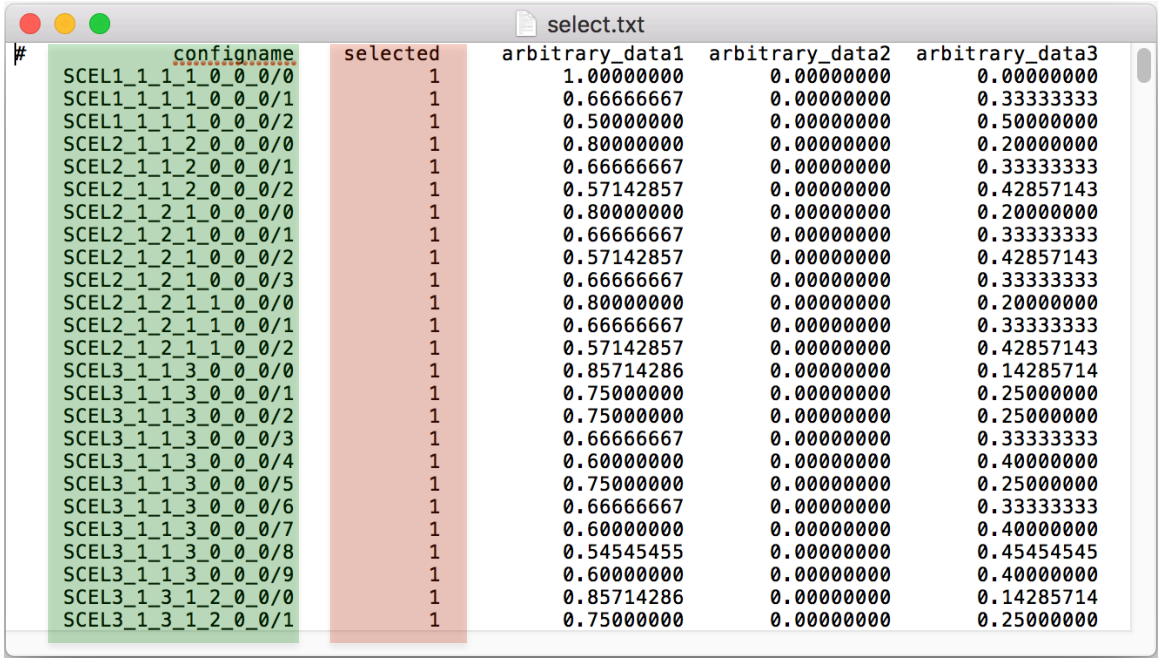
- arithmetic — add, subtract, multiply, divide, sqrt, exp
- comparison — equality, less-than, greater-than, etc.
- logical — AND, OR, NOT, etc.
- ranges — min, max, and others
- text — search and match with regular expressions

for documentation see `casm select --help operators`

Working with selections

Selections can be edited and combined using 'casm select'

The 'MASTER' selection is stored in the configuration database and is the default selection for most operations. A selection can be redirected to a file using `--output <filename>`



select.txt

# configname	selected	arbitrary_data1	arbitrary_data2	arbitrary_data3
SCEL1_1_1_1_0_0_0/0	1	1.00000000	0.00000000	0.00000000
SCEL1_1_1_1_0_0_0/1	1	0.66666667	0.00000000	0.33333333
SCEL1_1_1_1_0_0_0/2	1	0.50000000	0.00000000	0.50000000
SCEL2_1_1_2_0_0_0/0	1	0.80000000	0.00000000	0.20000000
SCEL2_1_1_2_0_0_0/1	1	0.66666667	0.00000000	0.33333333
SCEL2_1_1_2_0_0_0/2	1	0.57142857	0.00000000	0.42857143
SCEL2_1_2_1_0_0_0/0	1	0.80000000	0.00000000	0.20000000
SCEL2_1_2_1_0_0_0/1	1	0.66666667	0.00000000	0.33333333
SCEL2_1_2_1_0_0_0/2	1	0.57142857	0.00000000	0.42857143
SCEL2_1_2_1_0_0_0/3	1	0.66666667	0.00000000	0.33333333
SCEL2_1_2_1_1_0_0/0	1	0.80000000	0.00000000	0.20000000
SCEL2_1_2_1_1_0_0/1	1	0.66666667	0.00000000	0.33333333
SCEL2_1_2_1_1_0_0/2	1	0.57142857	0.00000000	0.42857143
SCEL3_1_1_3_0_0_0/0	1	0.85714286	0.00000000	0.14285714
SCEL3_1_1_3_0_0_0/1	1	0.75000000	0.00000000	0.25000000
SCEL3_1_1_3_0_0_0/2	1	0.75000000	0.00000000	0.25000000
SCEL3_1_1_3_0_0_0/3	1	0.66666667	0.00000000	0.33333333
SCEL3_1_1_3_0_0_0/4	1	0.60000000	0.00000000	0.40000000
SCEL3_1_1_3_0_0_0/5	1	0.75000000	0.00000000	0.25000000
SCEL3_1_1_3_0_0_0/6	1	0.66666667	0.00000000	0.33333333
SCEL3_1_1_3_0_0_0/7	1	0.60000000	0.00000000	0.40000000
SCEL3_1_1_3_0_0_0/8	1	0.54545455	0.00000000	0.45454545
SCEL3_1_1_3_0_0_0/9	1	0.60000000	0.00000000	0.40000000
SCEL3_1_3_1_2_0_0/0	1	0.85714286	0.00000000	0.14285714
SCEL3_1_3_1_2_0_0/1	1	0.75000000	0.00000000	0.25000000

Selection(s) to be operated can be read from a file using `--configs <filename>`

Edit selections using CASM query syntax

- casm select --set <query> Create a selection from scratch
- casm select --set-on <query> Set 'selected' from false to true if <query> is true
- casm select --set-off <query> Set 'selected' from true to false if <query> is true

Combine selections using logical operators (AND, OR, NOT, XOR)

Working with selections

Selections can be edited and combined using 'casm select'

The 'MASTER' selection is stored in the configuration database and is the default selection for most operations. A selection can be redirected to a file using `--output <filename>`

select.txt

# configname	selected	arbitrary_data1	arbitrary_data2	arbitrary_data3
SCEL1_1_1_1_0_0_0/0	1	1.00000000	0.00000000	0.00000000
SCEL1_1_1_1_0_0_0/1	1	0.66666667	0.00000000	0.33333333
SCEL1_1_1_1_0_0_0/2	1	0.50000000	0.00000000	0.50000000
SCEL2_1_1_2_0_0_0/0	1	0.80000000	0.00000000	0.20000000
SCEL2_1_1_2_0_0_0/1	1	0.66666667	0.00000000	0.33333333
SCEL2_1_1_2_0_0_0/2	1	0.57142857	0.00000000	0.42857143
SCEL2_1_2_1_0_0_0/0	1	0.80000000	0.00000000	0.20000000
SCEL2_1_2_1_0_0_0/1	1	0.66666667	0.00000000	0.33333333
SCEL2_1_2_1_0_0_0/2	1	0.57142857	0.00000000	0.42857143
SCEL2_1_2_1_0_0_0/3	1	0.66666667	0.00000000	0.33333333
SCEL2_1_2_1_1_0_0/0	1	0.80000000	0.00000000	0.20000000
SCEL2_1_2_1_1_0_0/1	1	0.66666667	0.00000000	0.33333333
SCEL2_1_2_1_1_0_0/2	1	0.57142857	0.00000000	0.42857143
SCEL3_1_1_3_0_0_0/0	1	0.85714286	0.00000000	0.14285714
SCEL3_1_1_3_0_0_0/1	1	0.75000000	0.00000000	0.25000000
SCEL3_1_1_3_0_0_0/2	1	0.75000000	0.00000000	0.25000000
SCEL3_1_1_3_0_0_0/3	1	0.66666667	0.00000000	0.33333333
SCEL3_1_1_3_0_0_0/4	1	0.60000000	0.00000000	0.40000000
SCEL3_1_1_3_0_0_0/5	1	0.75000000	0.00000000	0.25000000
SCEL3_1_1_3_0_0_0/6	1	0.66666667	0.00000000	0.33333333
SCEL3_1_1_3_0_0_0/7	1	0.60000000	0.00000000	0.40000000
SCEL3_1_1_3_0_0_0/8	1	0.54545455	0.00000000	0.45454545
SCEL3_1_1_3_0_0_0/9	1	0.60000000	0.00000000	0.40000000
SCEL3_1_3_1_2_0_0/0	1	0.85714286	0.00000000	0.14285714
SCEL3_1_3_1_2_0_0/1	1	0.75000000	0.00000000	0.25000000

Predefined selections:

- 'ALL': specifies the total configuration database.
- 'CALCULATED': specifies all configurations for which DFT data exist.
- 'NONE': equivalent to 'ALL' but with all 'selected' bits set to 0

Selection(s) to be operated can be read from a file using `--configs <filename>`

Edit selections using CASM query syntax

casm select --set <query> Create a selection from scratch

casm select --set-on <query> Set 'selected' from false to true if <query> is true

casm select --set-off <query> Set 'selected' from true to false if <query> is true

Combine selections using logical operators (AND, OR, NOT, XOR)

CASM Query Syntax, Part II

CASM query syntax provides simple interface to **explore**, **screen**, and select configurations.

Use 'casm query' to print properties of selected configurations (supports JSON and CSV)
Example:

```
casm query --columns "comp(a)" "formation_energy" --configs select.csv
```

will print 'casm select'-style CSV output to the screen:

# configname	selected	comp(a)	formation_energy
SCEL1_1_1_1_0_0_0/0	1	0.00000000	-0.23245234
SCEL1_1_1_1_0_0_0/1	1	0.50000000	0.10088099
SCEL1_1_1_1_0_0_0/2	1	1.00000000	0.26754766
SCEL2_1_1_2_0_0_0/0	1	0.25000000	-0.03245234
SCEL2_1_1_2_0_0_0/1	1	0.50000000	0.10088099
SCEL2_1_1_2_0_0_0/2	1	0.75000000	0.19611909
SCEL2_1_2_1_0_0_0/0	1	0.25000000	-0.03245234
SCEL2_1_2_1_0_0_0/1	1	0.50000000	0.10088099
SCEL2_1_2_1_0_0_0/2	1	0.75000000	0.19611909
SCEL2_1_2_1_0_0_0/3	1	0.50000000	0.10088099
SCEL2_1_2_1_1_0_0/0	1	0.25000000	-0.03245234
SCEL2_1_2_1_1_0_0/1	1	0.50000000	0.10088099
SCEL2_1_2_1_1_0_0/2	1	0.75000000	0.19611909
SCEL3_1_1_3_0_0_0/0	1	0.16666667	-0.08959520
SCEL3_1_1_3_0_0_0/1	1	0.33333333	0.01754766

output can be redirected to a file with `--output <filename>`

Query syntax can also be used to screen configurations at the enumeration stage, so they don't pollute the configuration database.
Example:

```
casm enum --configurations --all --filter "lt(atom_frac(O),0.25)"
```

will only enumerate configurations that are dilute in oxygen (having O atomic fraction less than 0.25)

DFT calculations

Provide DFT code input files and parameters

- Pseudopotentials, k-point density, energy cut-off, and settings files

Use 'casm select' to identify a selection of configurations to calculate

casm run –e vasp.relax --configs my_selection.csv

- vasp.relax is python script that automates job submission for each configuration. It submits the DFT calculation, handles common errors, ensures that results are converged and relaxed, and writes results to a format CASM understands (properties.calc.json).
- Uses 'pbs' python module to automate job submission (https://github.com/prisms-center/pbs)

Current support for VASP, support for other codes is planned. Contributions from the user community are encouraged!

properties.calc.json files

`$ROOT/training_data/$SCEL/$CONFIGID/calctype.$CALCTYPE/`

CASM interprets calculation data from properties.calc.json file

Generated by ‘vasp.relax’ or manually using ‘vasp.relax.report’

properties.calc.json


```
{
    "atom_type": [
        "O",
        "Zr"
    ],
    "atoms_per_type": [
        1,
        4
    ],
    "coord_mode": "direct",
    "relaxed_basis": [
        [0.3333333, 0.6666666, 0.125],
        [0.0, 0.0, 0.99549145],
        [0.0, 0.0, 0.49846384],
        [0.6666667, 0.3333334, 0.25450855],
        [0.6666667, 0.3333334, 0.75153616]
    ],
    "relaxed_energy": -44.06613533,
    "relaxed_forces": [
        [-0.0, -0.0, -0.0],
        [0.0, -0.0, -0.01503973],
        [0.0, -0.0, -0.00709864],
        [-0.0, 0.0, 0.01503973],
        [-0.0, 0.0, 0.00709864]
    ],
    "relaxed_lattice": [
        [3.26993078, 0.0, 0.0],
        [-1.63496539, 2.83184311, 0.0],
        [0.0, 0.0, 10.46480612]
    ]
}
```

Relaxations and Mapping/Import

casm update:

Read all new property files, and check relaxation

lattice_deformation (LD):

Degree of homogeneous strain (per atom)

basis_deformation (BD):

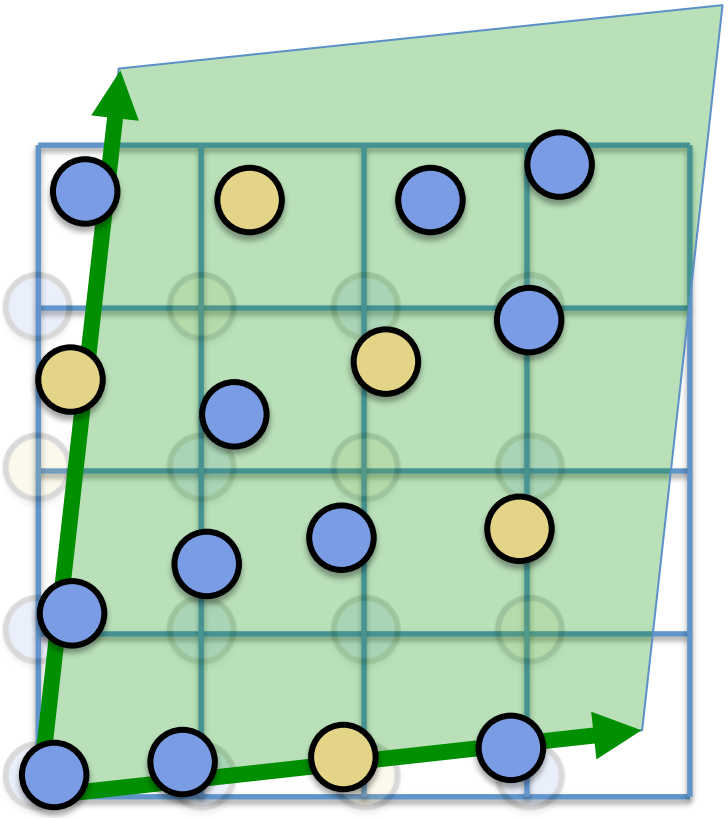
Mean square displacement

total deformation cost:

w*LD+(1-w)*BD , 0<w<1

casm import:

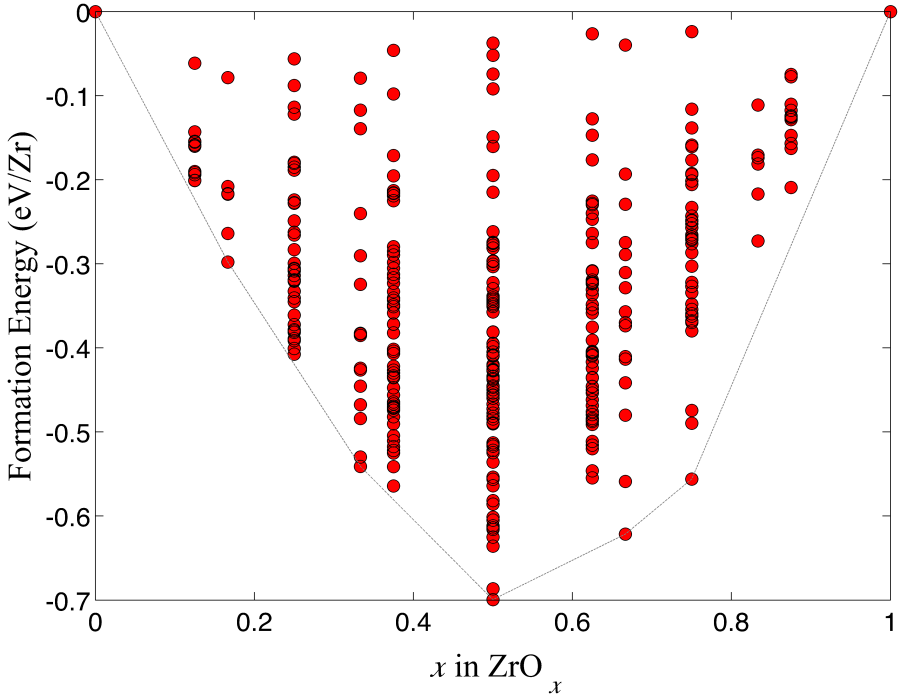
Use same algorithms as casm update to add ideal or relaxed configurations to project (even if not enumerated)



Very large relaxations may change the configuration. Conflicts can be resolved using either min-energy or min-deformation rule

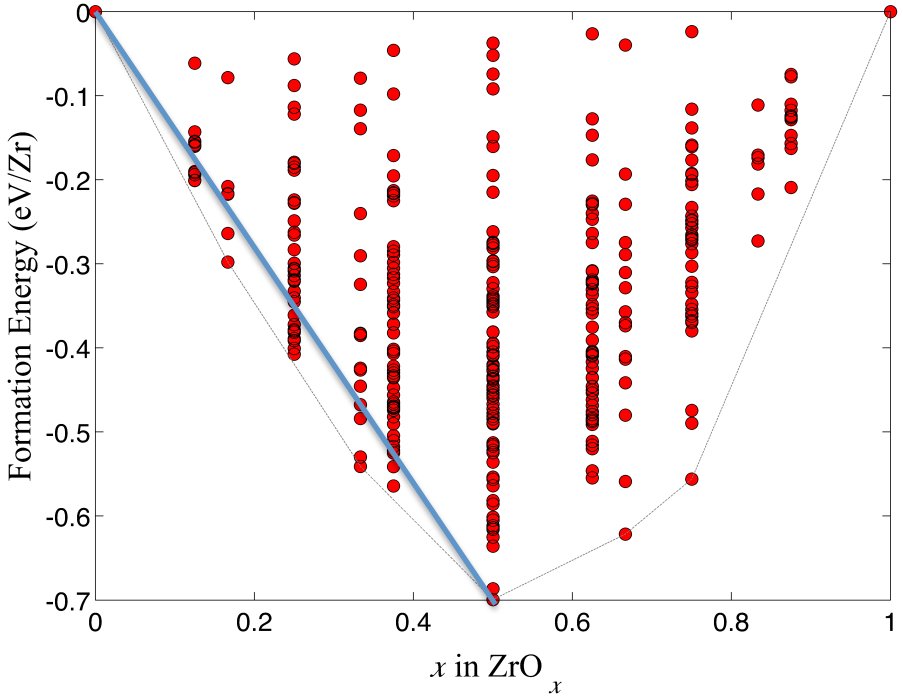
Setting energy reference

Chemical energy can only be measured relative to reference compounds



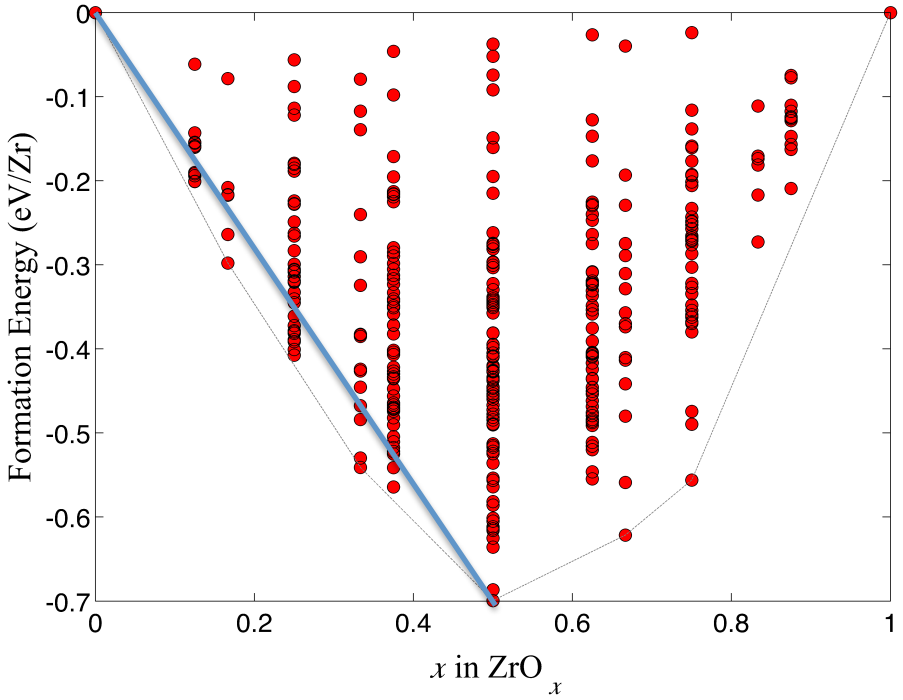
Setting energy reference

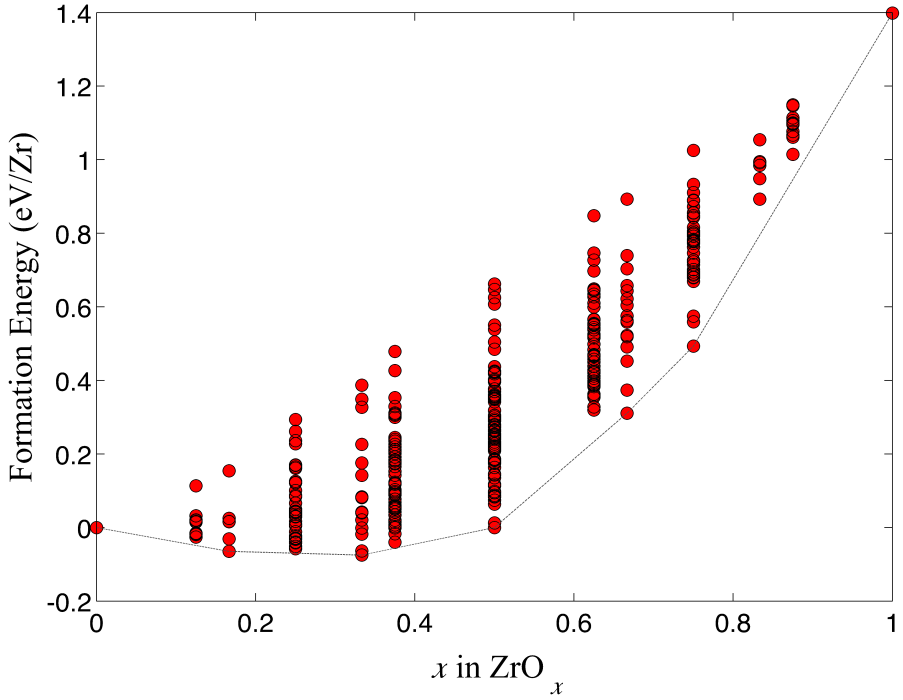
Chemical energy can only be measured relative to reference compounds



Setting energy reference

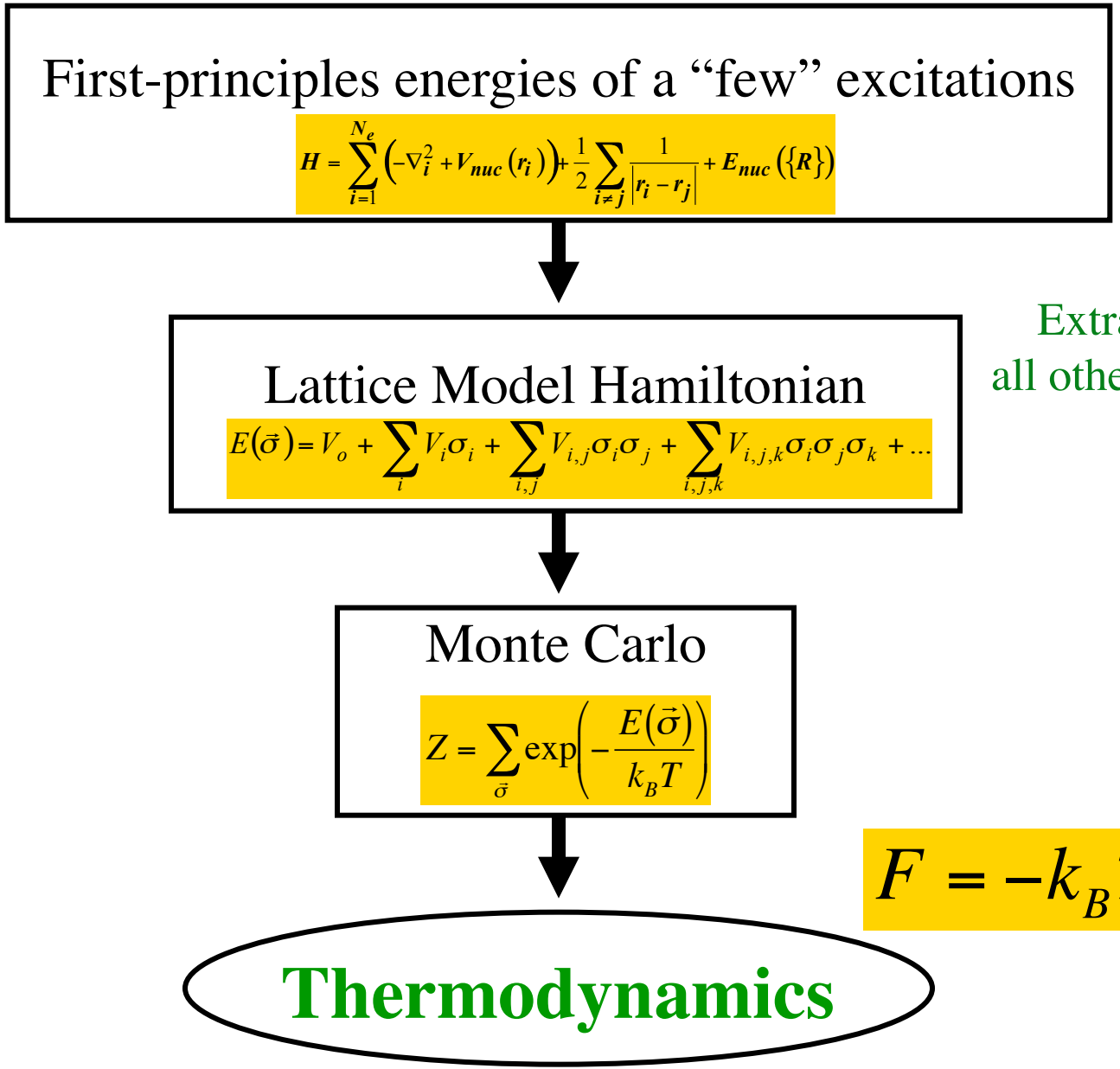
Chemical energy can only be measured relative to reference compounds





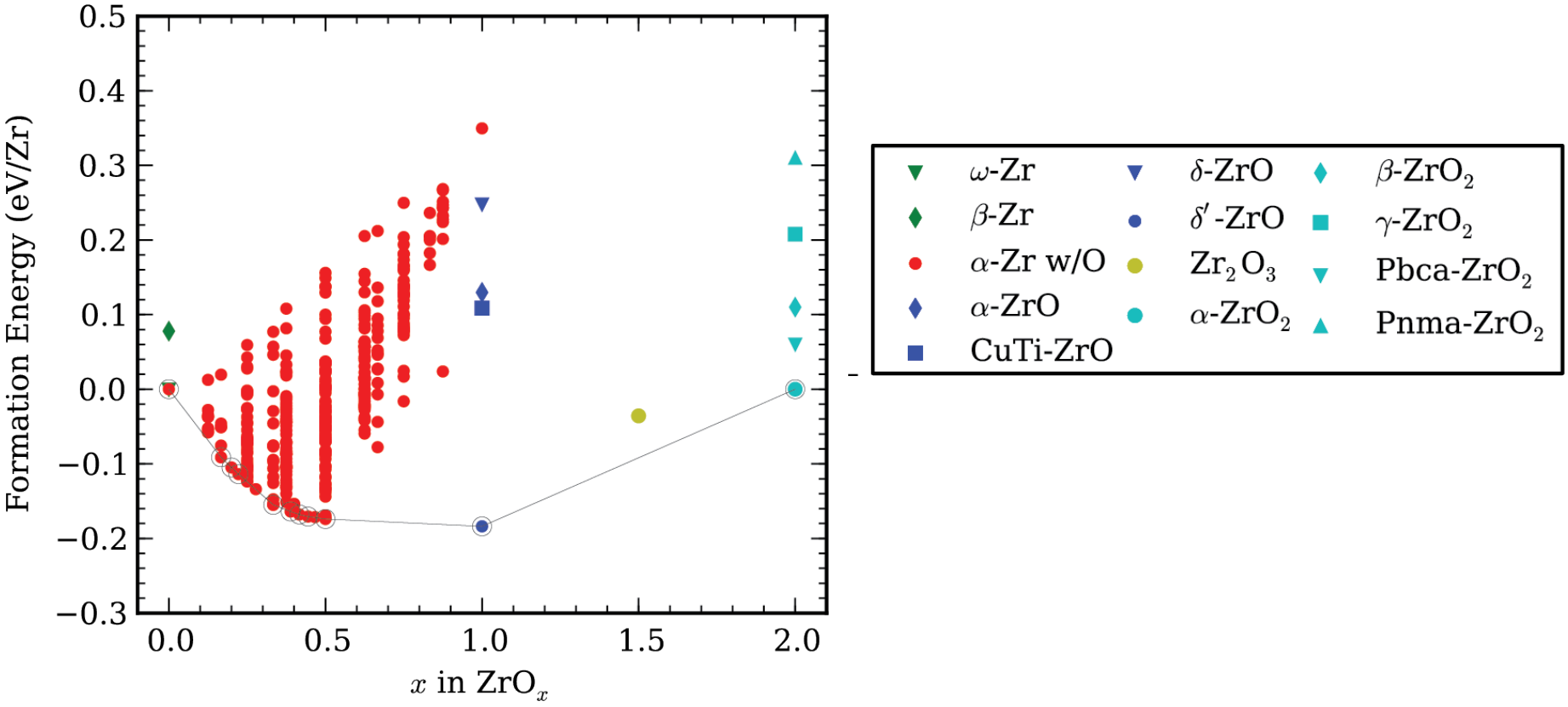
`casm ref --set-auto` will attempt to set energy references as near to extreme stoichiometries as possible (left case)

CASM: Clusters Approach to Statistical Mechanics



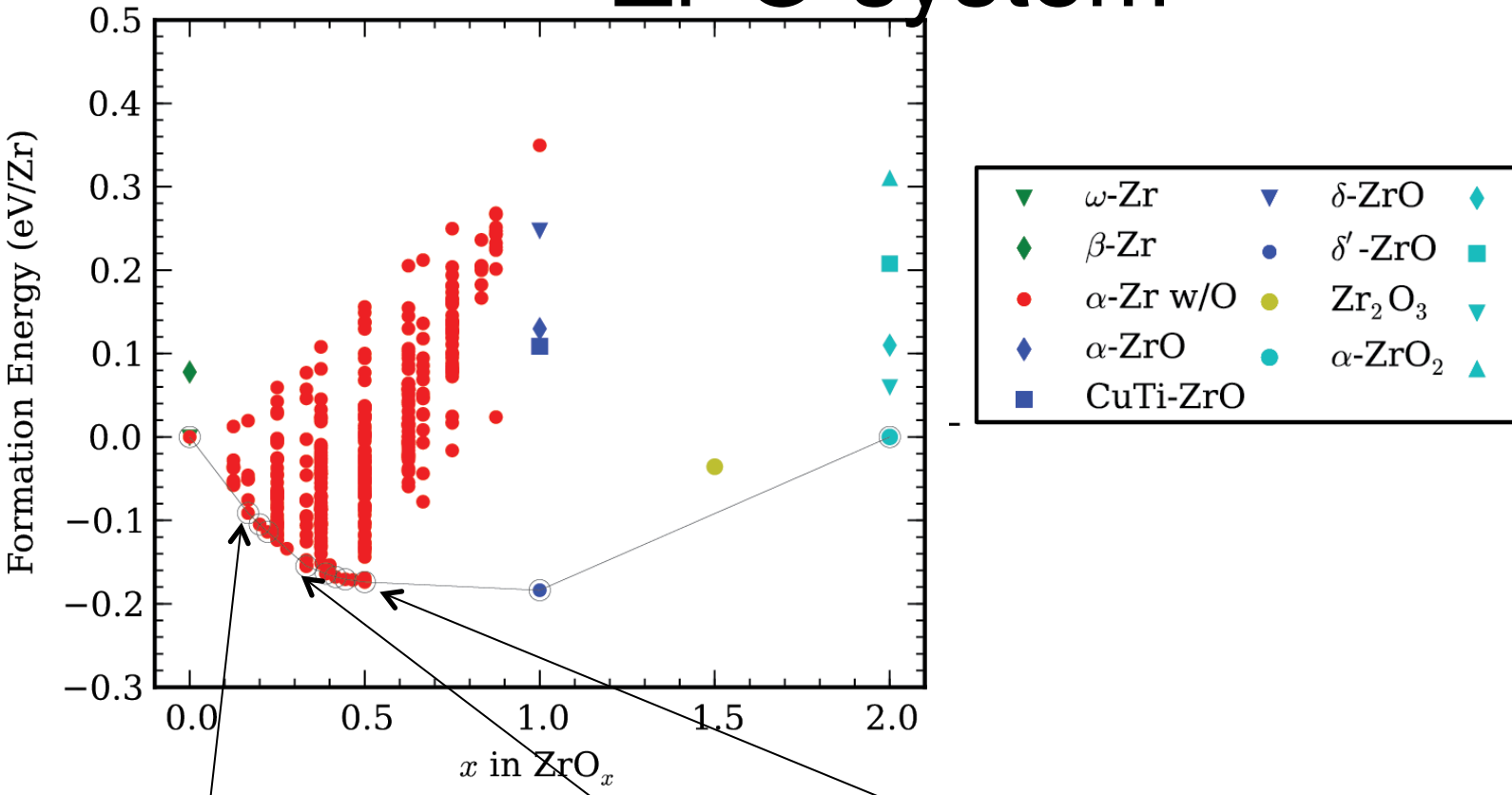
Zr-O system

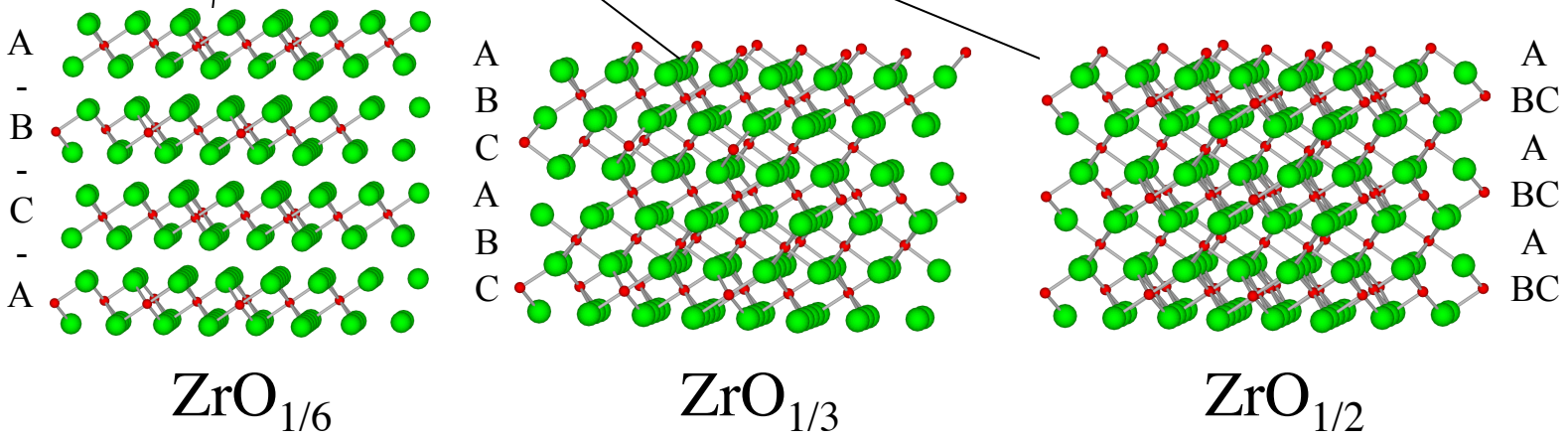
Each red circle specifies belongs to a specific configuration of HCP-based ZrO_x



Puchala and Van der Ven, Phys. Rev. B **88** 094108 (2013).

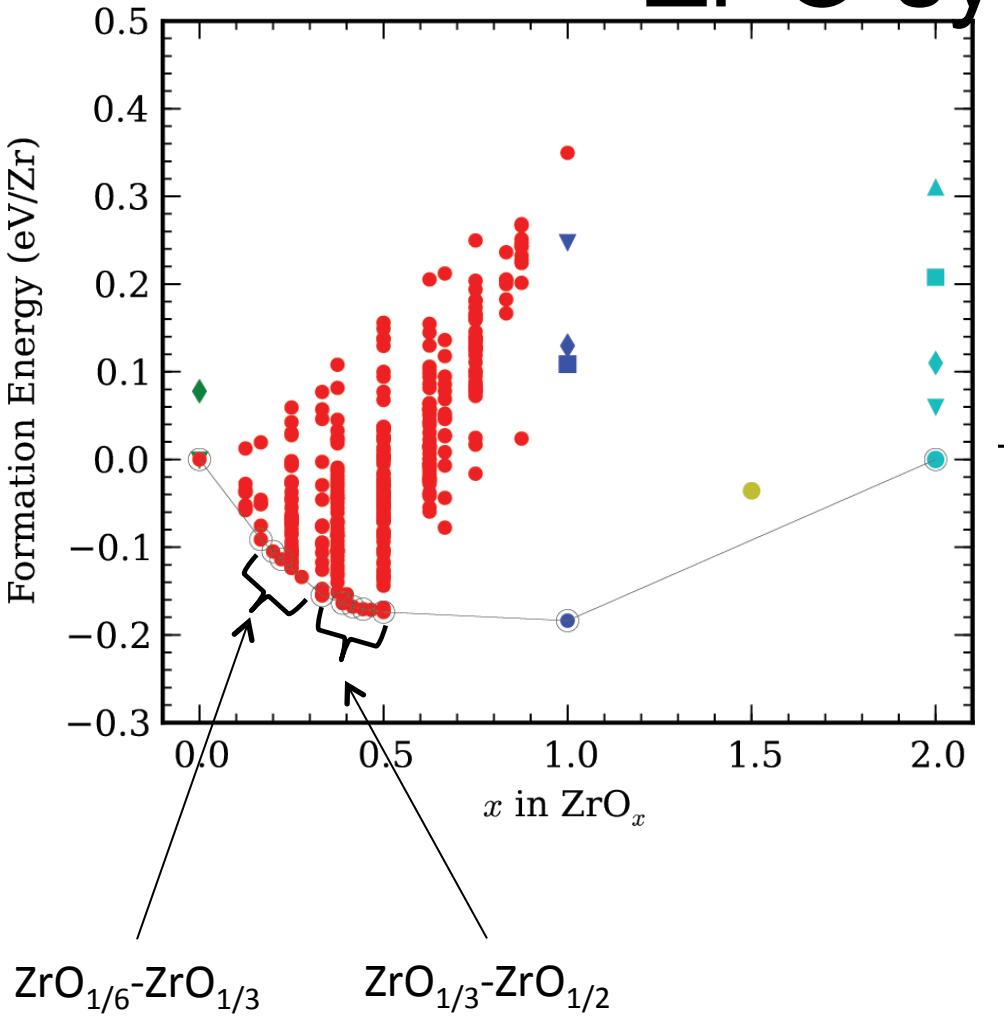
Zr-O system



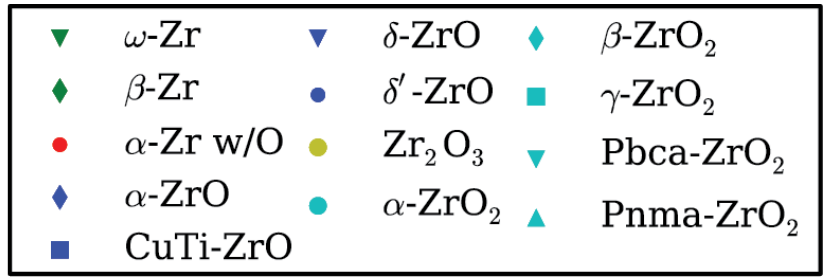


Puchala and Van der Ven, Phys. Rev. B **88** 094108 (2013).

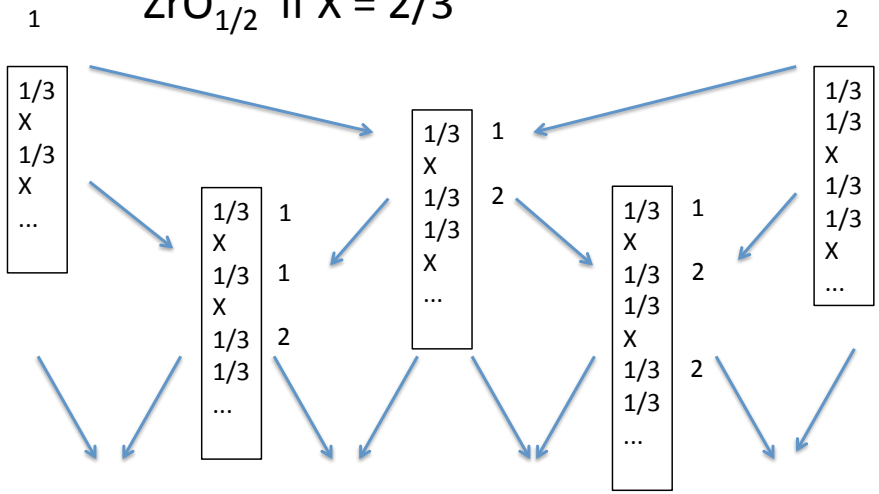
Zr-O system



O stacking sequences



$ZrO_{1/6}$ if X = 0
$ZrO_{1/2}$ if X = 2/3

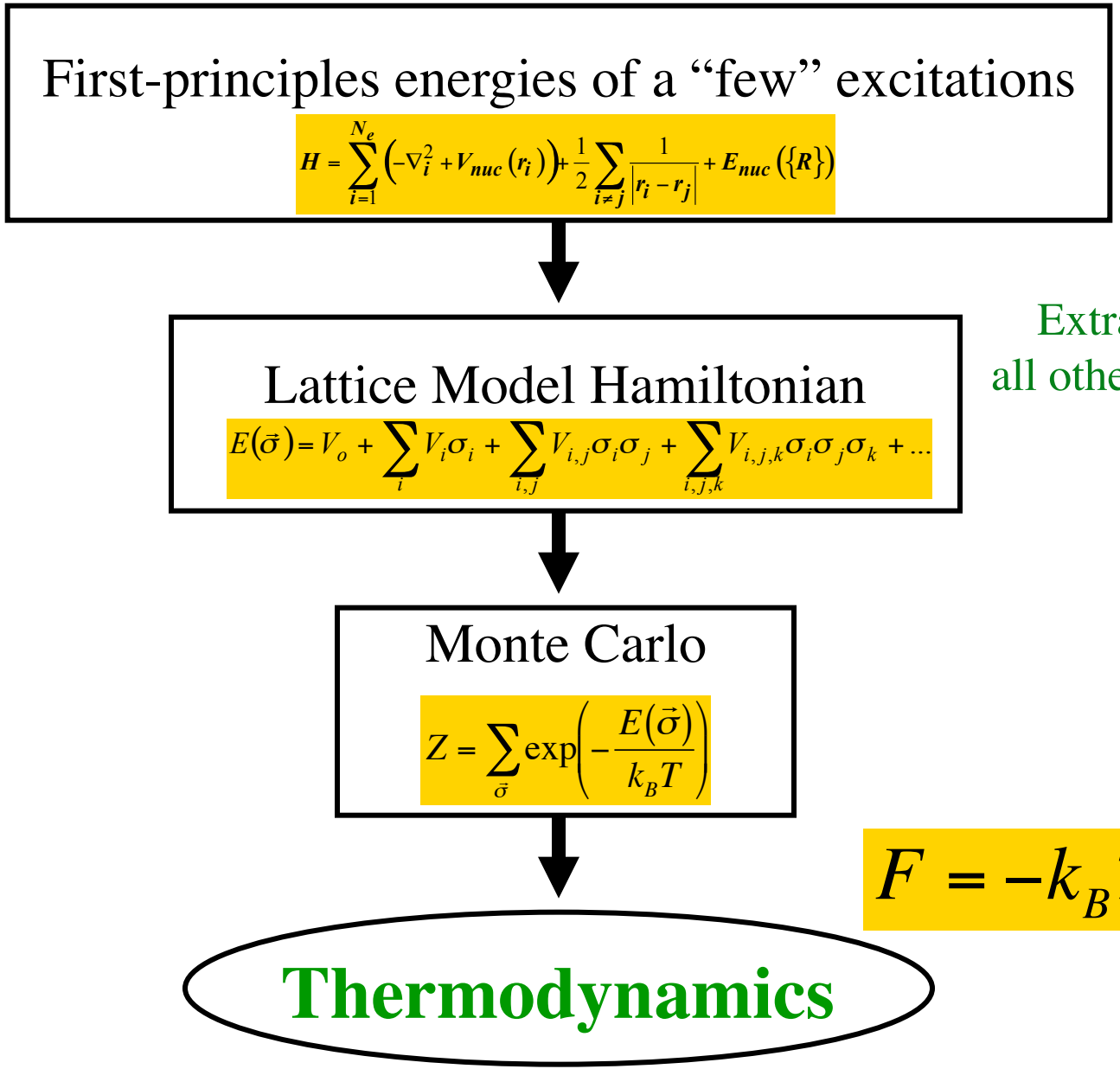


... can repeat for more "generations"

Puchala and Van der Ven, Phys. Rev. B **88** 094108 (2013).

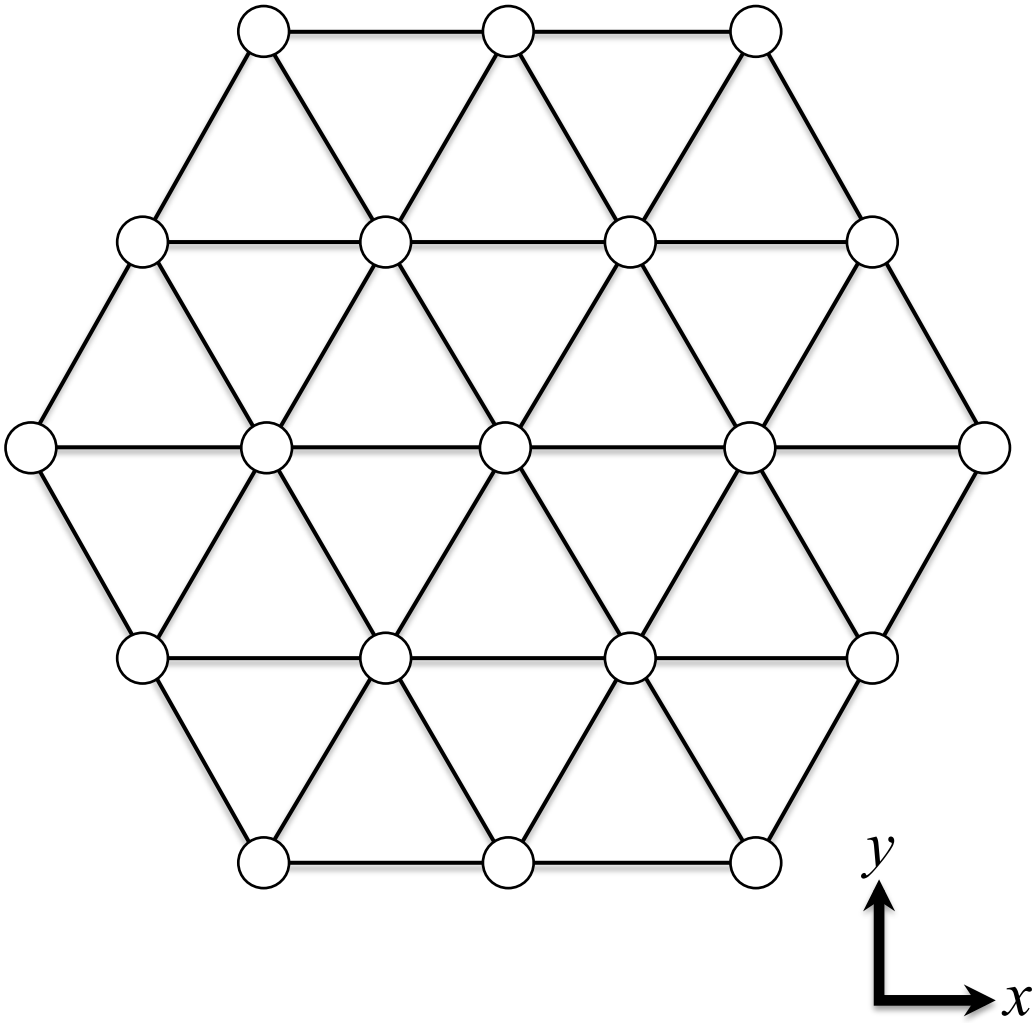
de Fontaine, Nature 343, 544 (1990).

CASM: Clusters Approach to Statistical Mechanics



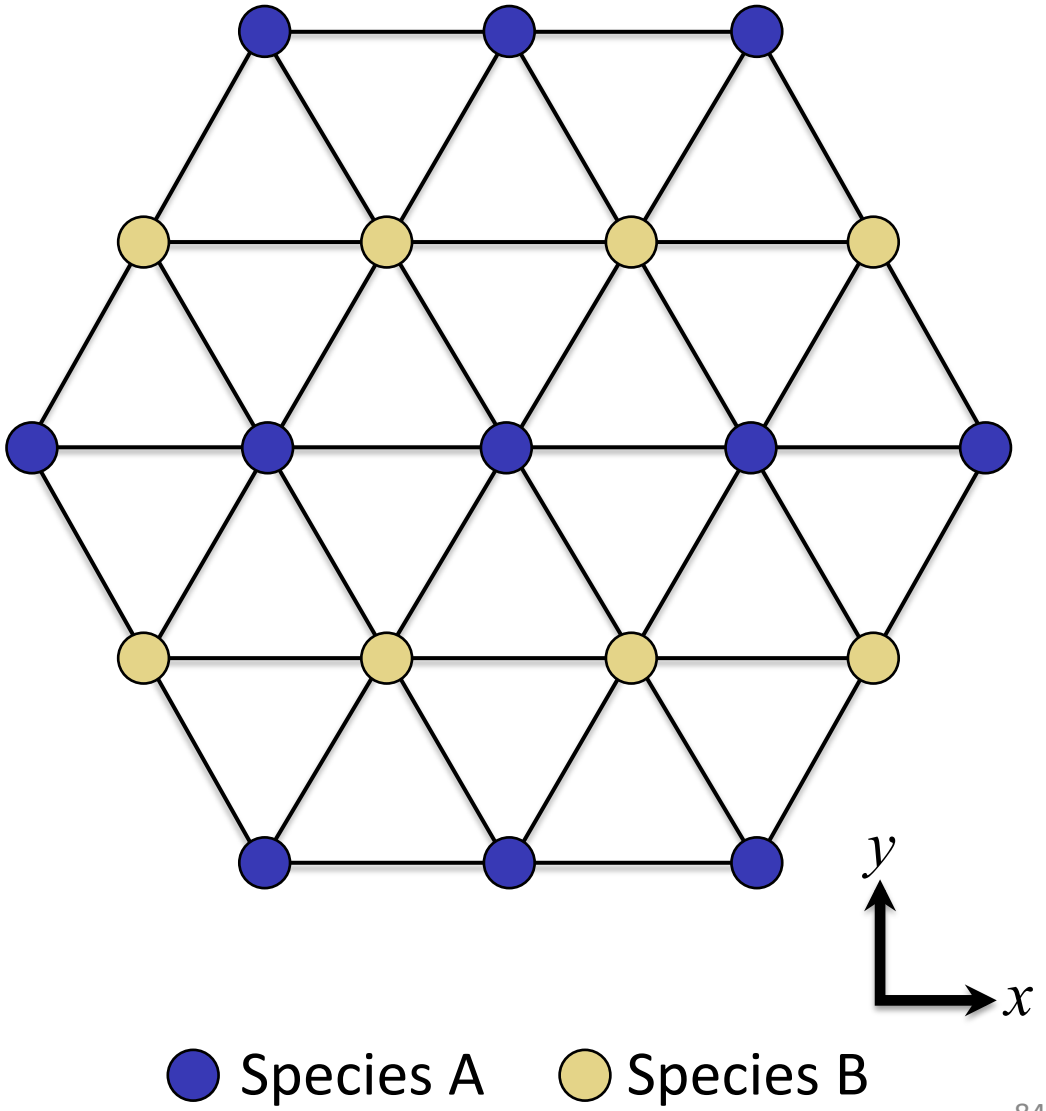
The **Clusters** Approach to Statistical Mechanics

Is there a compact way to approximate the energy of any configuration?



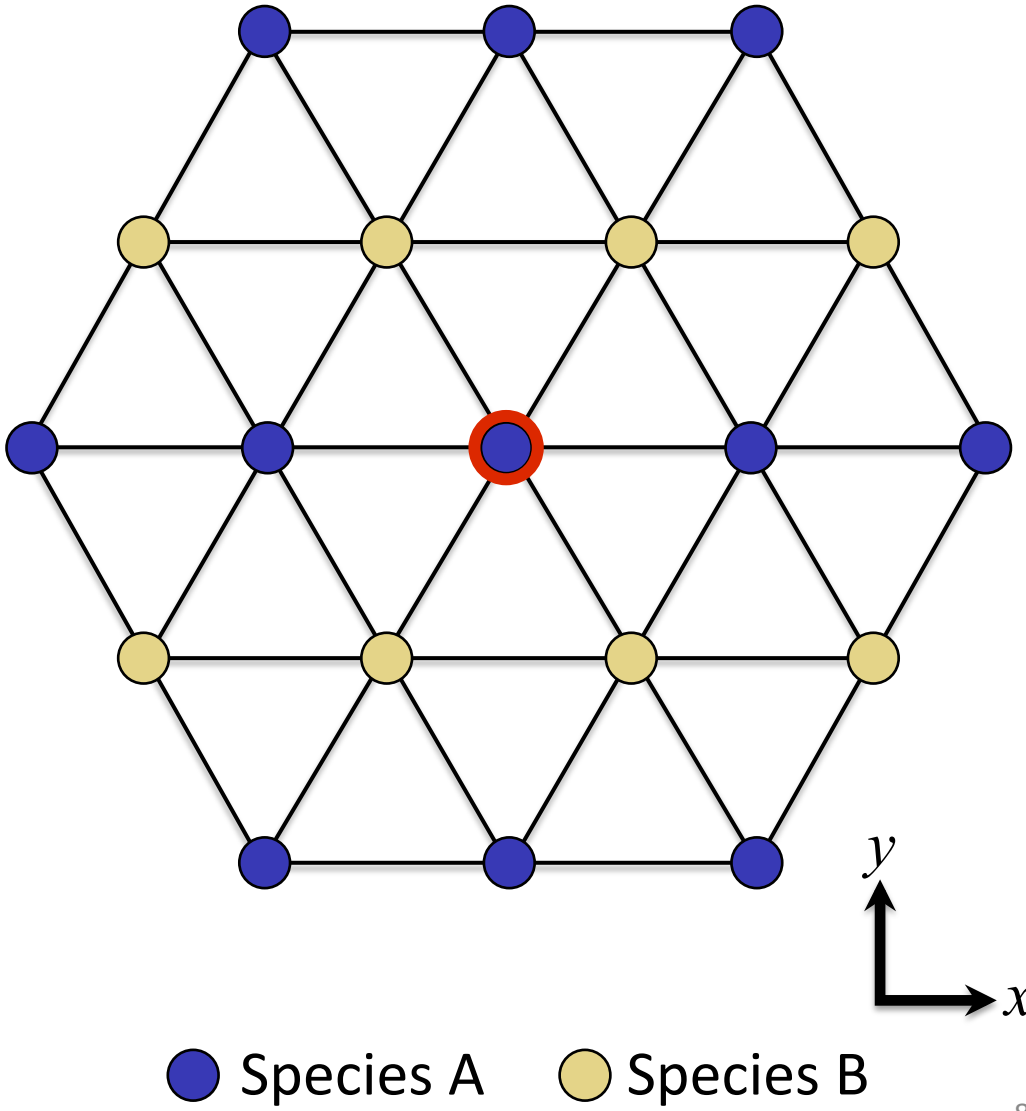
The **Clusters** Approach to Statistical Mechanics

Is there a compact way to approximate the energy of any configuration?



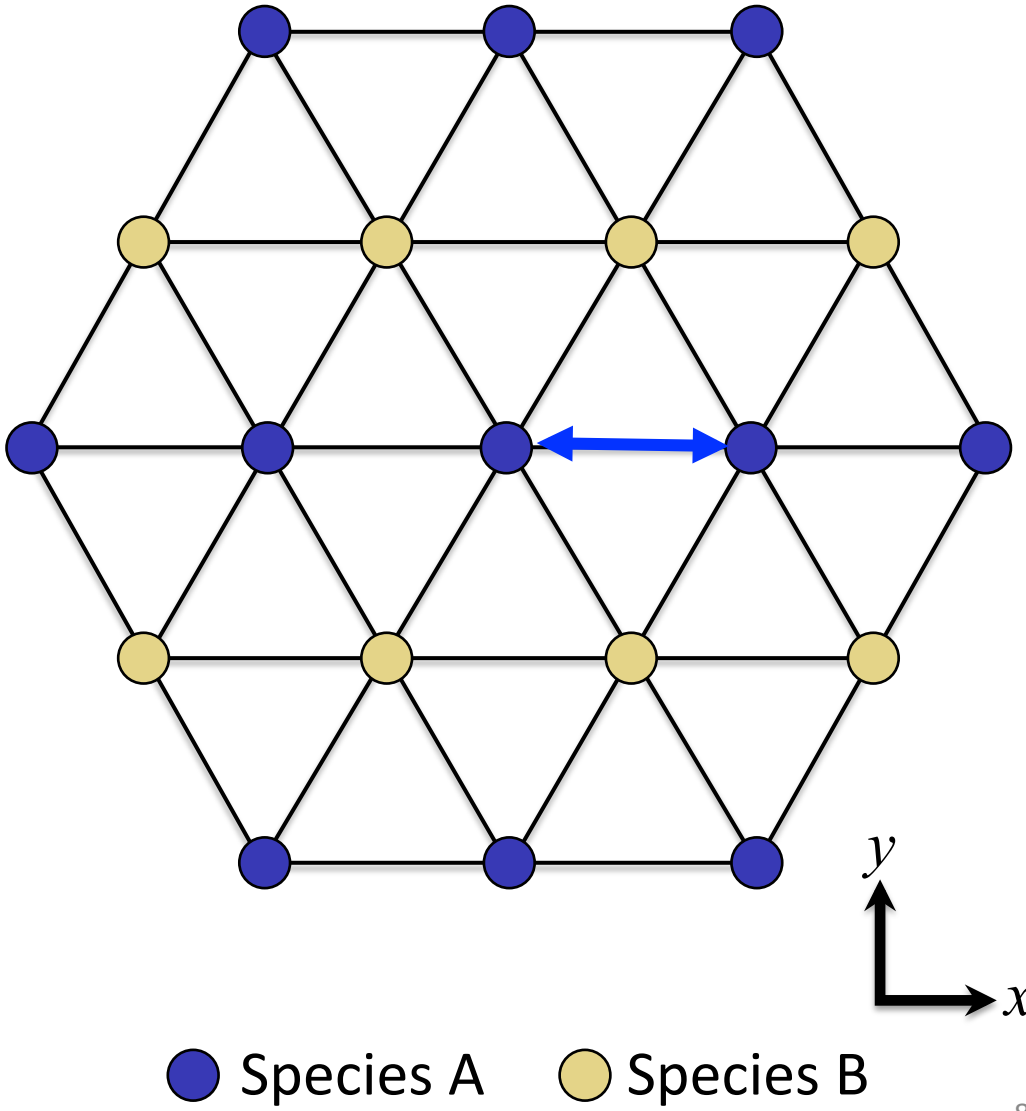
The **Clusters** Approach to Statistical Mechanics

Is there a compact way to approximate the energy of any configuration?



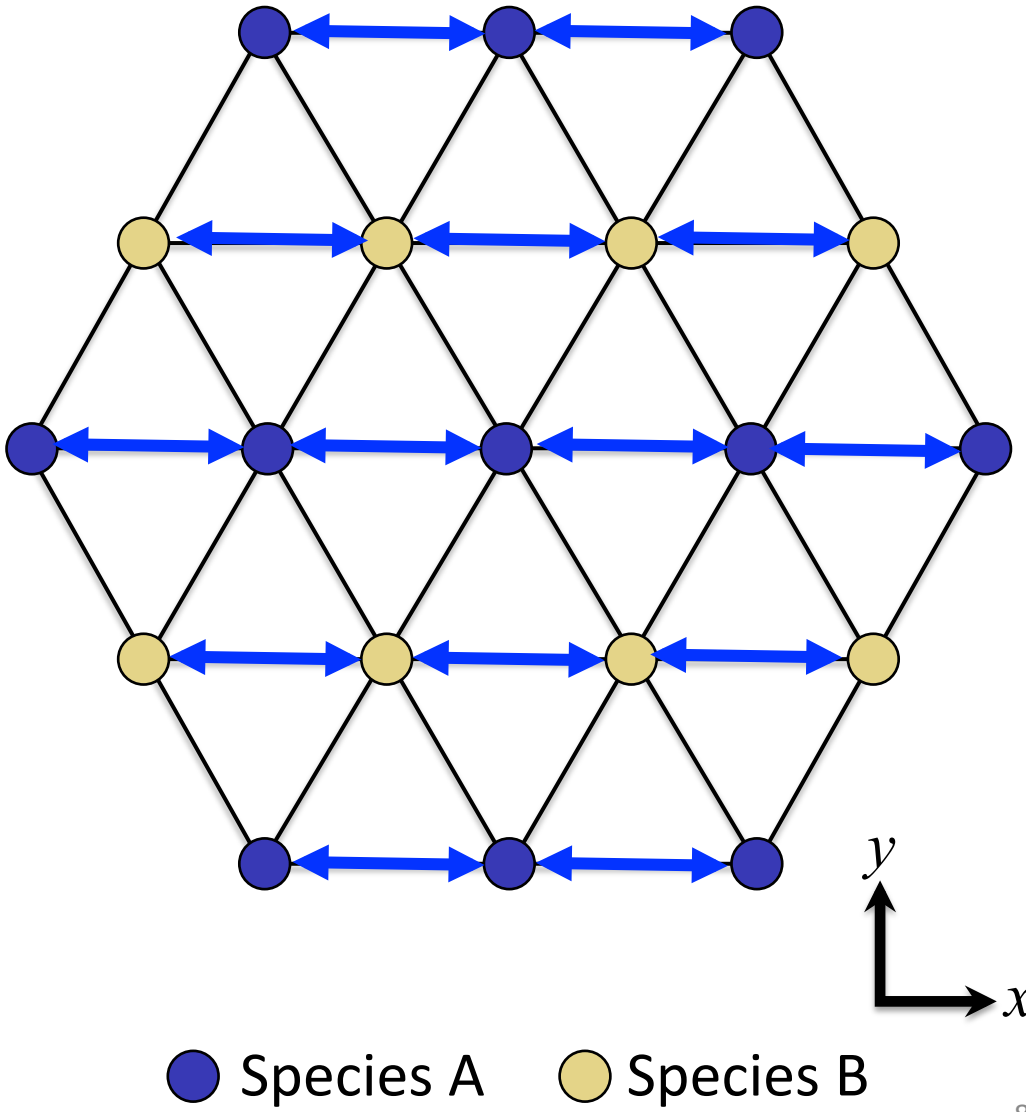
The **Clusters** Approach to Statistical Mechanics

Is there a compact way to approximate the energy of any configuration?



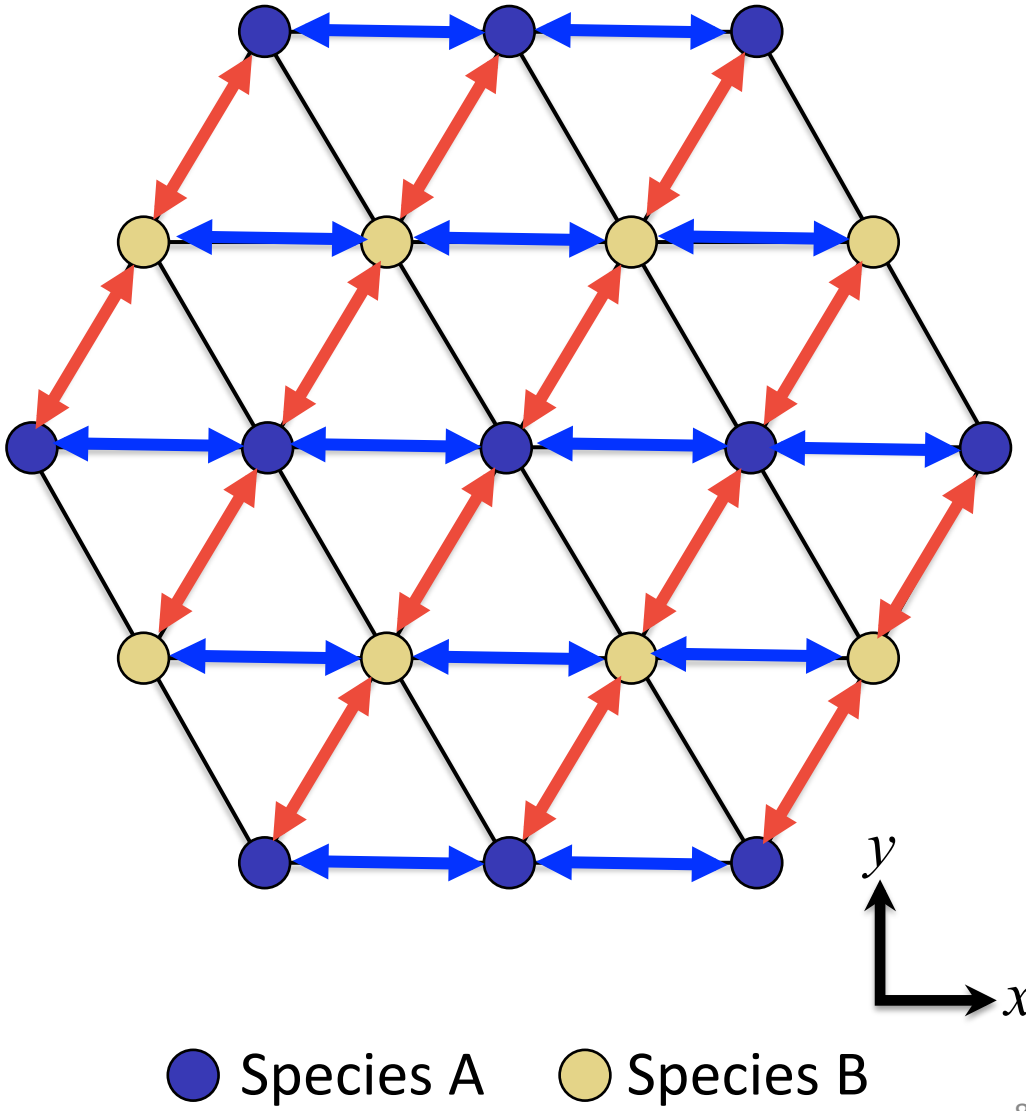
The **Clusters** Approach to Statistical Mechanics

Is there a compact way to approximate the energy of any configuration?



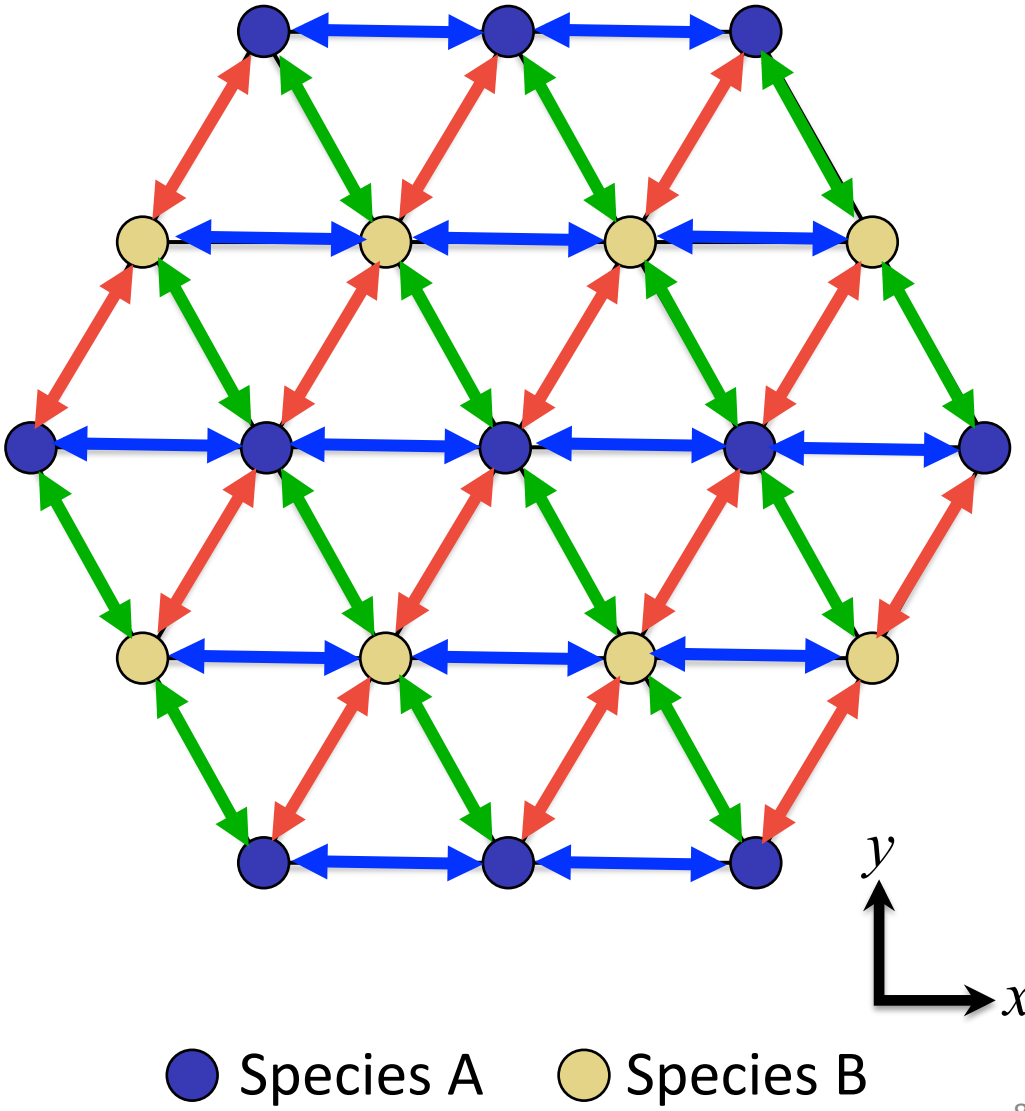
The **Clusters** Approach to Statistical Mechanics

Is there a compact way to approximate the energy of any configuration?



The **Clusters** Approach to Statistical Mechanics

Is there a compact way to approximate the energy of any configuration?

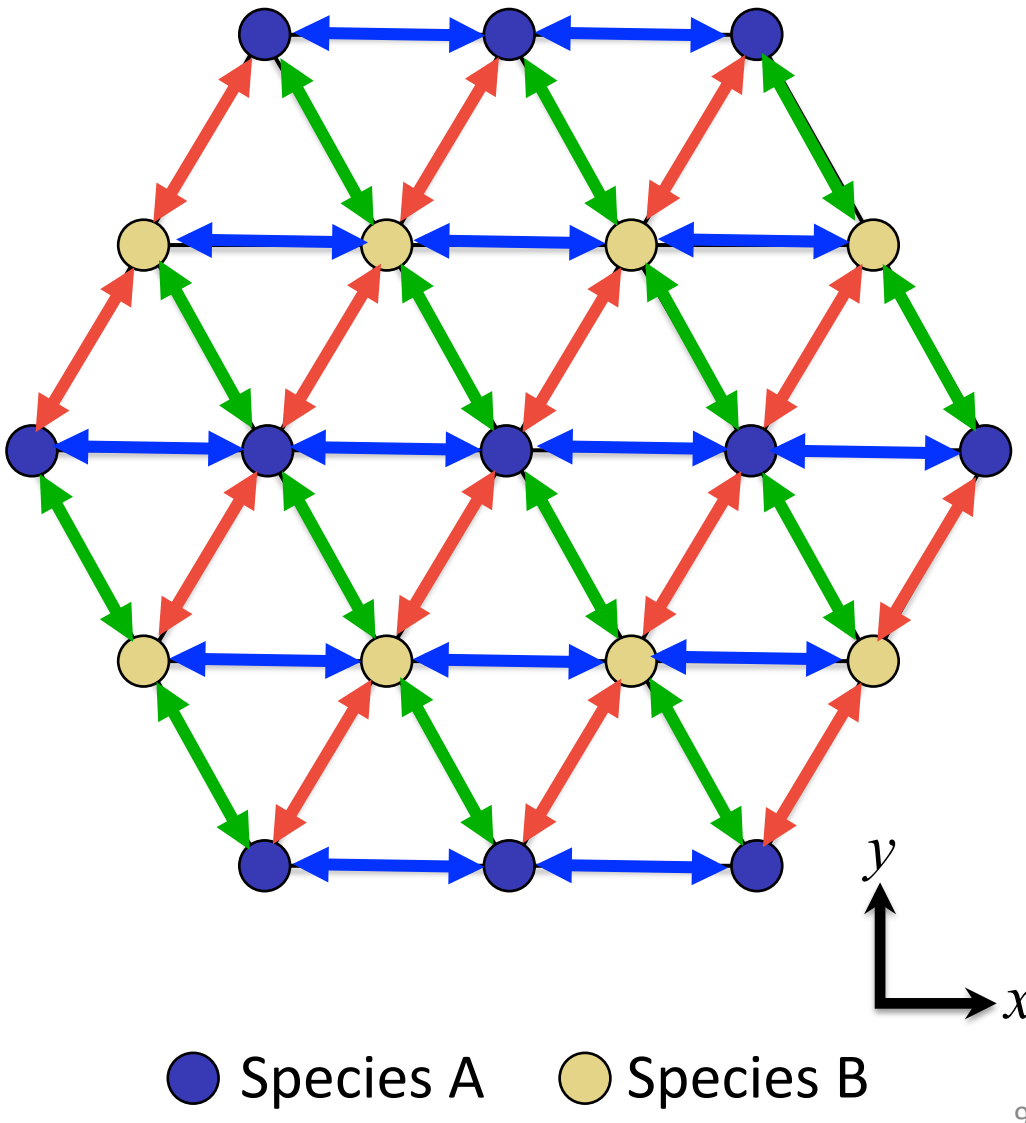


The **Clusters** Approach to Statistical Mechanics

Is there a compact way to approximate the energy of any configuration?

Symmetric averages of measurements of cluster occupation states.

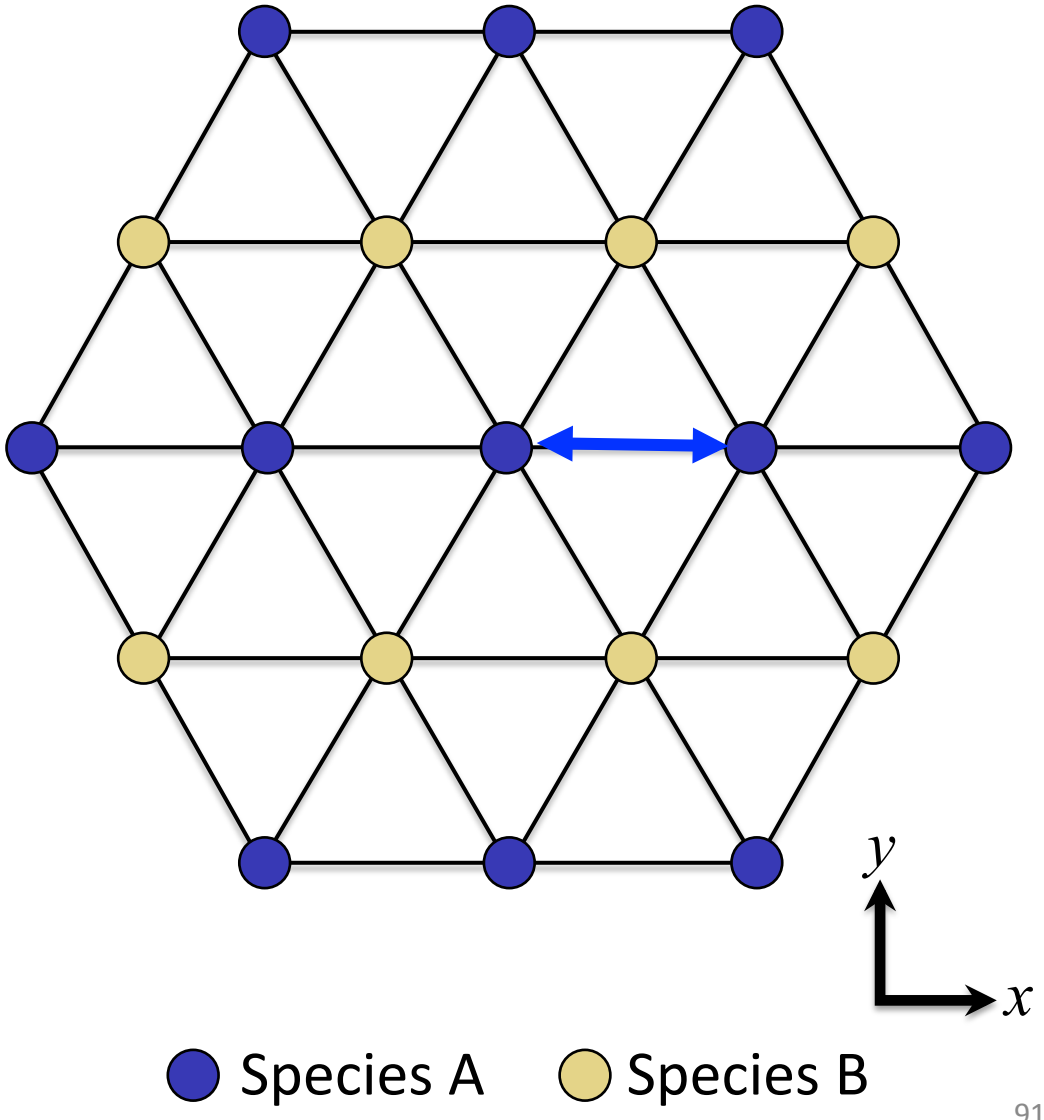
The **orbit** of the the NN pair



The **Clusters** Approach to Statistical Mechanics

Is there a compact way to approximate the energy of any configuration?

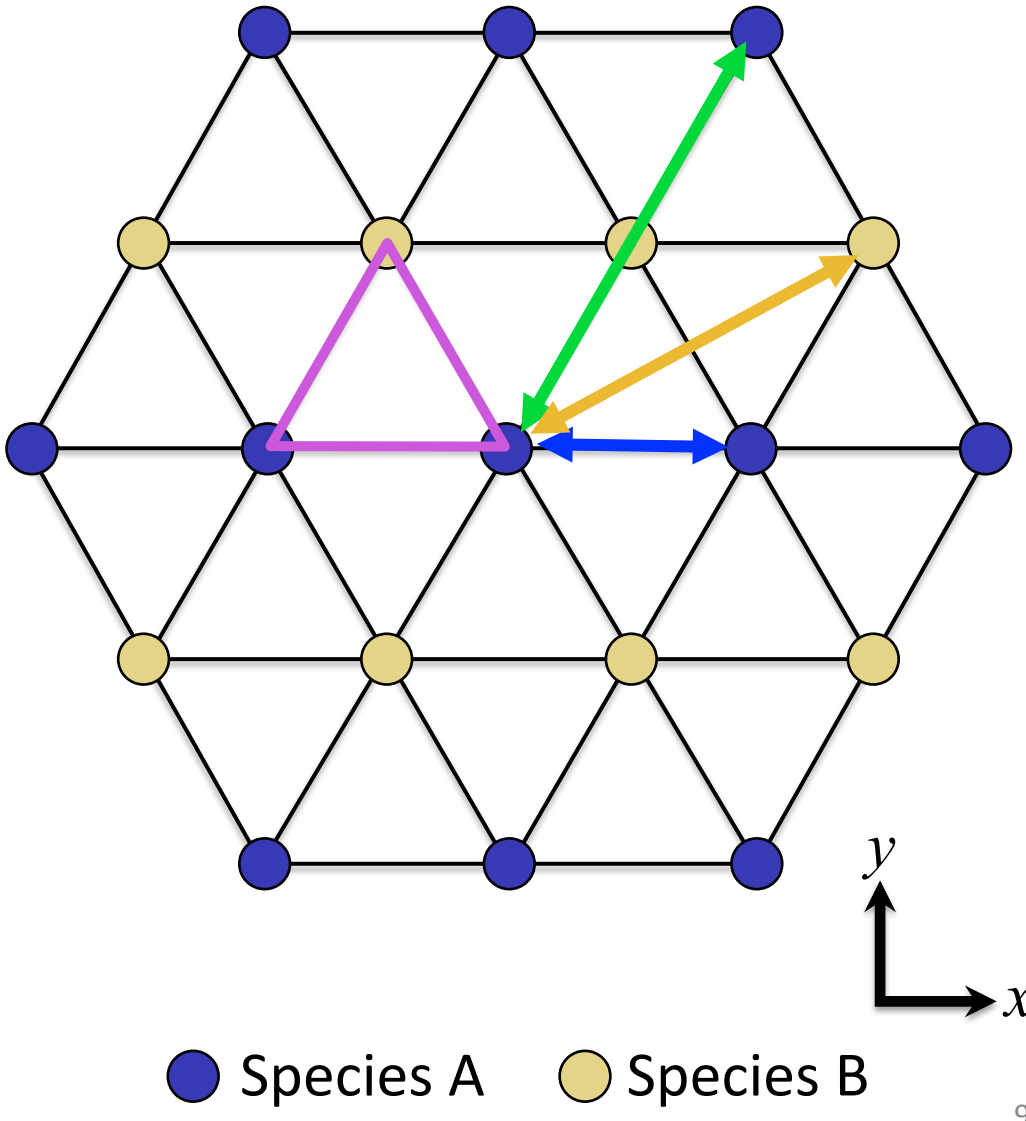
Symmetric averages of measurements of cluster occupation states.



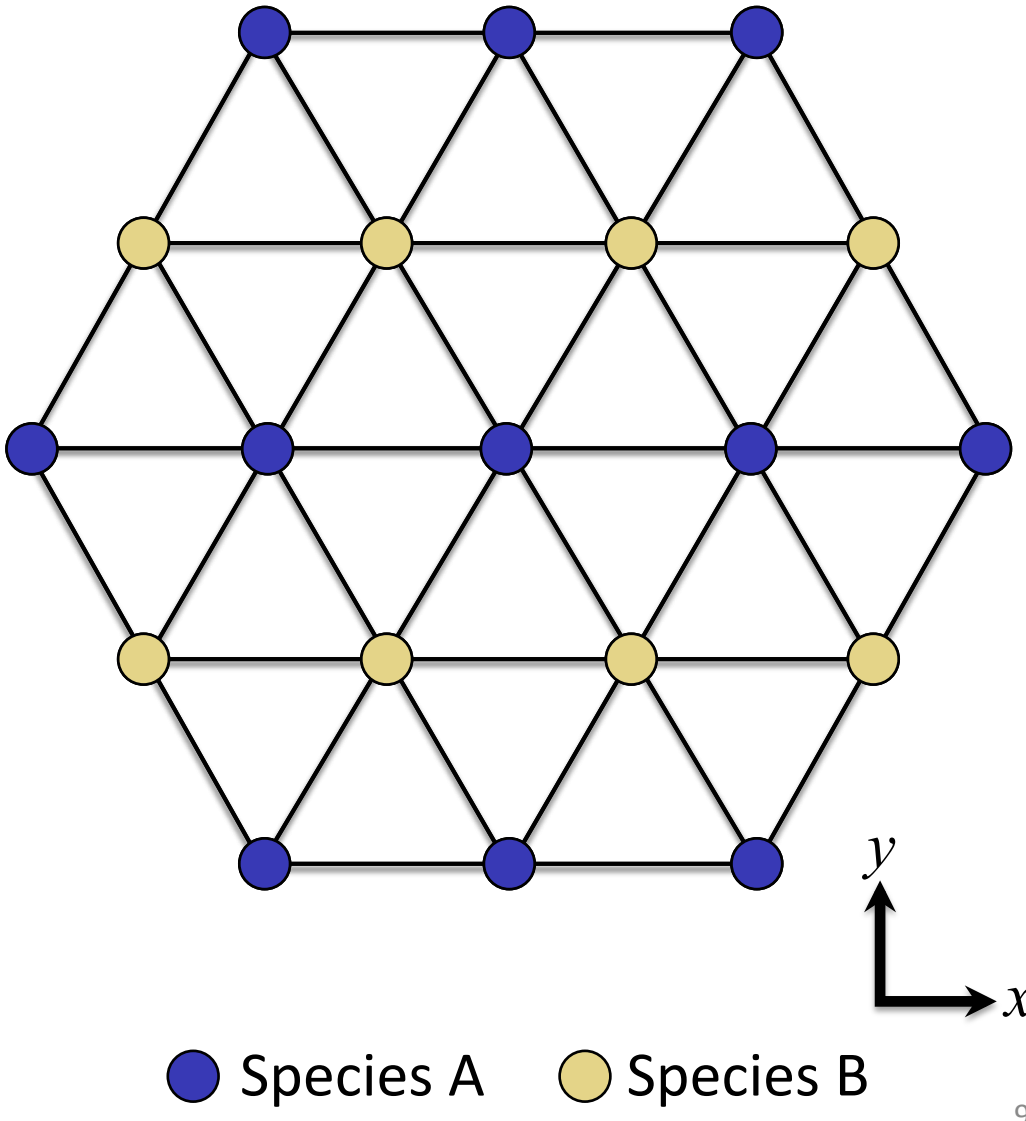
The **Clusters** Approach to Statistical Mechanics

Is there a compact way to approximate the energy of any configuration?

Symmetric averages of measurements of cluster occupation states.



The Cluster Expansion

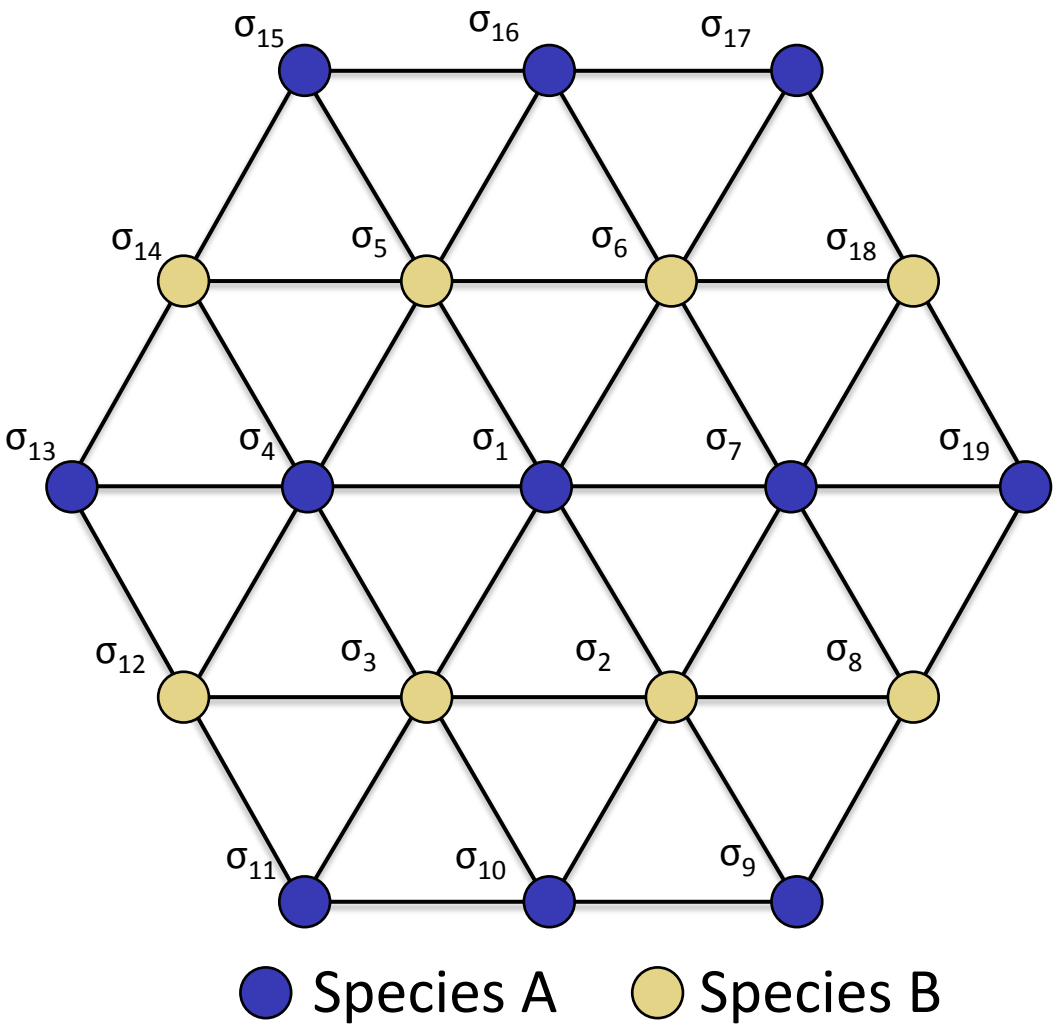


The Cluster Expansion

The occupant of lattice site *i* is described by the variable σ_i :

$\sigma_i = +1$ if species A

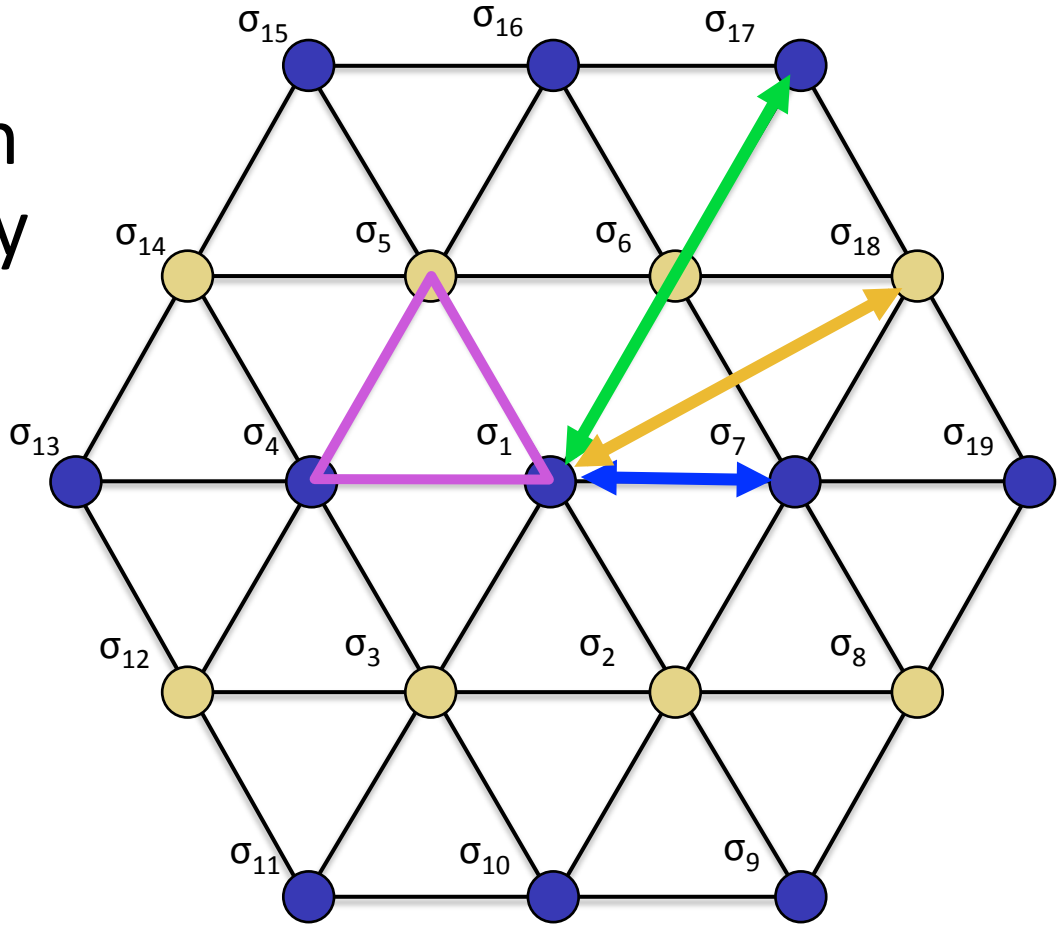
$\sigma_i = -1$ if species B



The Cluster Expansion

Interactions between sites are described by products of σ_i.

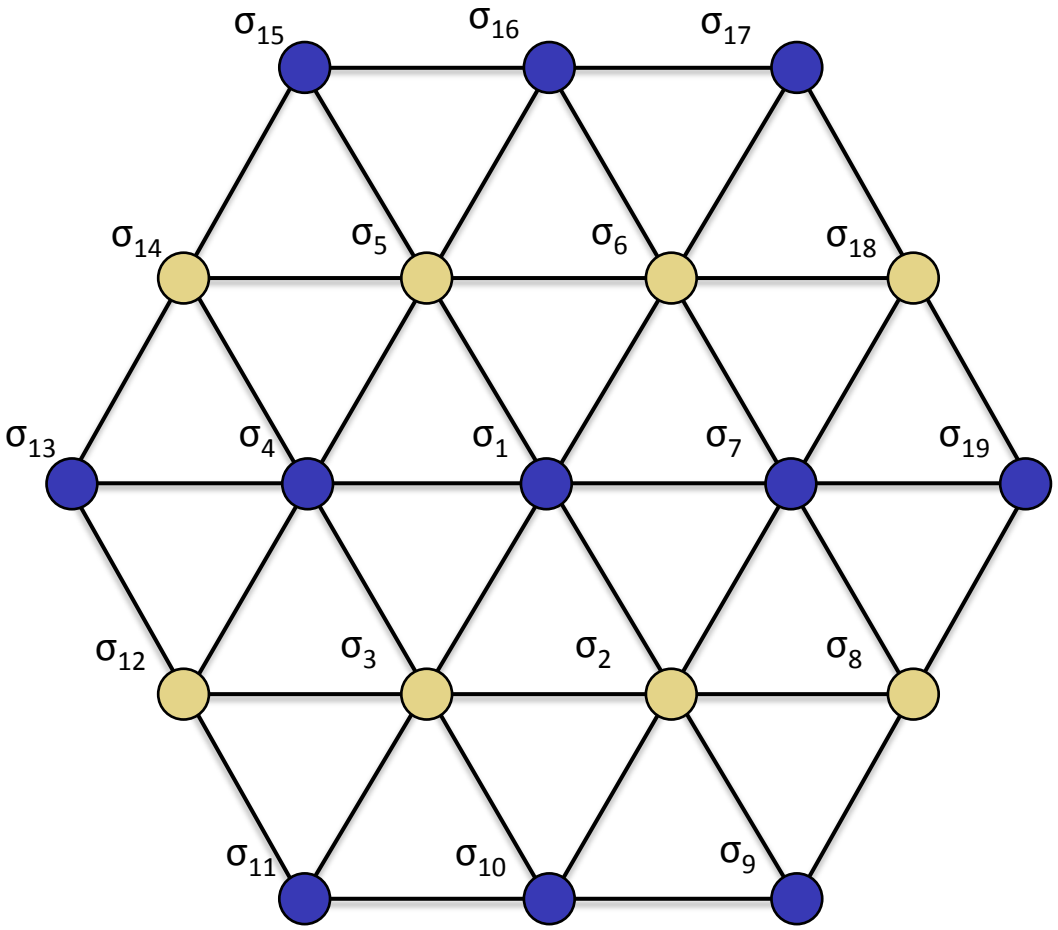
Fit interaction parameters to a database of DFT energies.



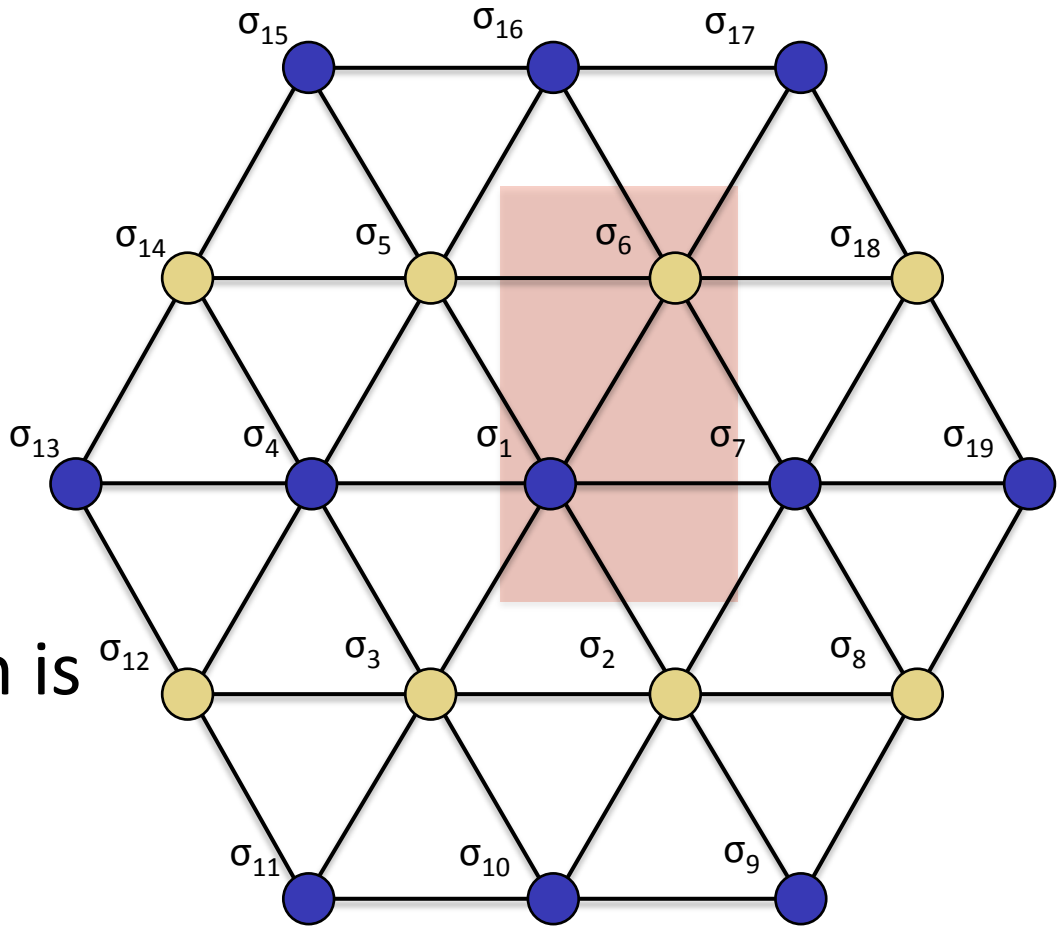
$$E = J_1\sigma_1 + J_{1,7}\sigma_1\sigma_7 + J_{1,18}\sigma_1\sigma_{18} + J_{1,17}\sigma_1\sigma_{17} + J_{1,4,5}\sigma_1\sigma_4\sigma_5 + \ldots$$

Then use symmetry to average over the entire crystal

Cluster Expansion Example



Cluster Expansion Example



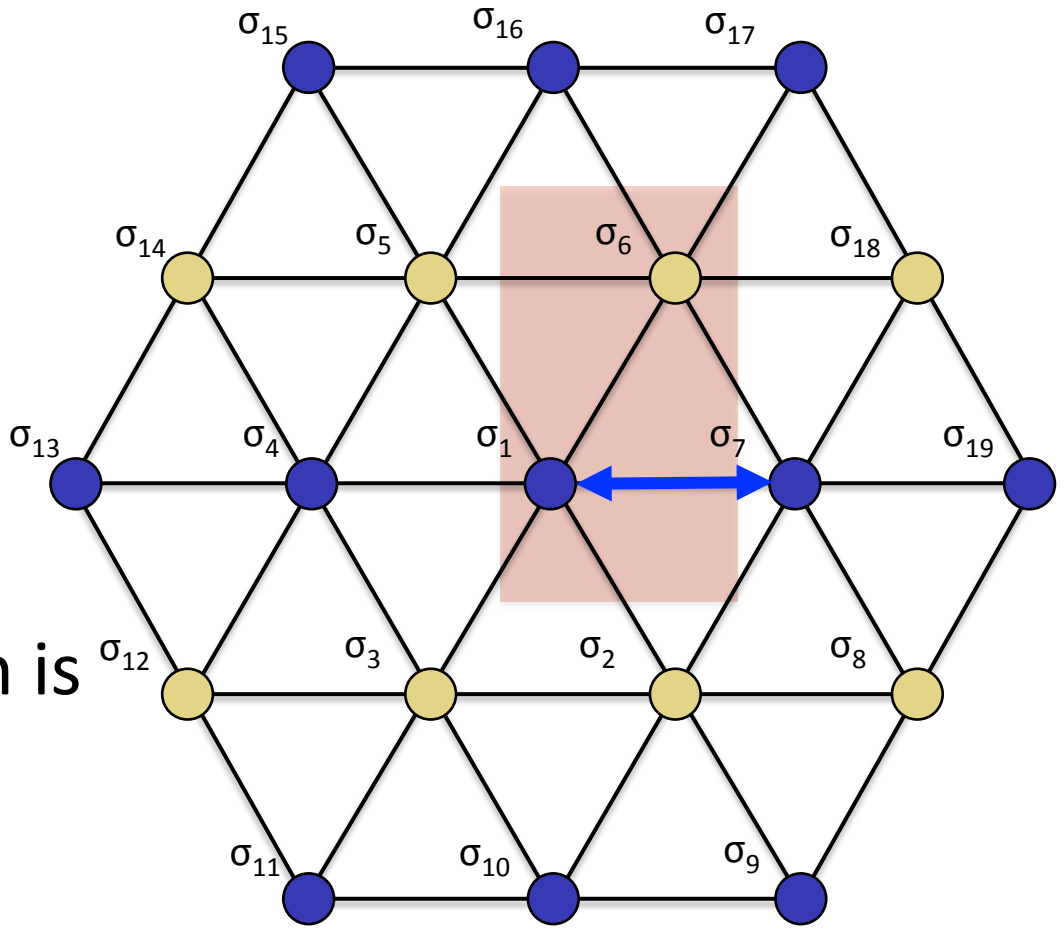
Average site occupation is

$\Gamma_1=(\sigma_1+\sigma_6)/2=0$

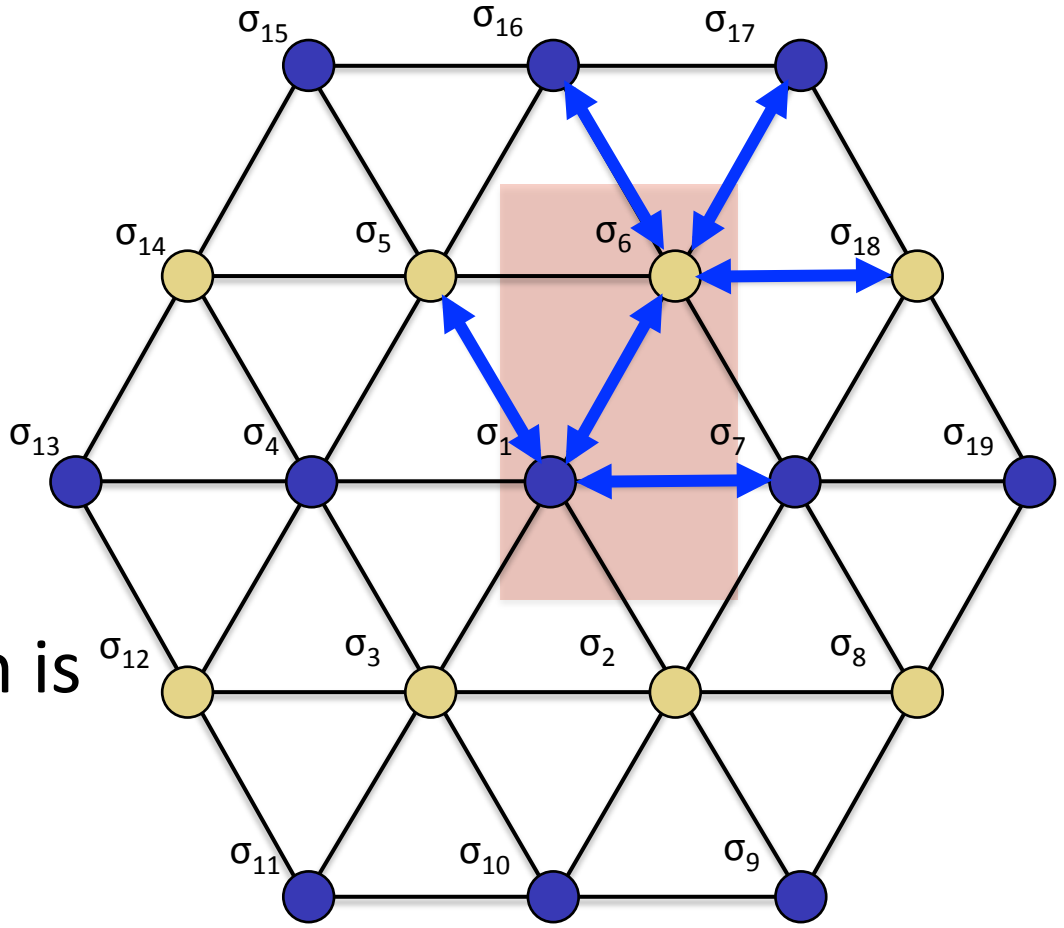
Cluster Expansion Example

Average site occupation is

$\Gamma_1=(\sigma_1+\sigma_6)/2=0$



Cluster Expansion Example



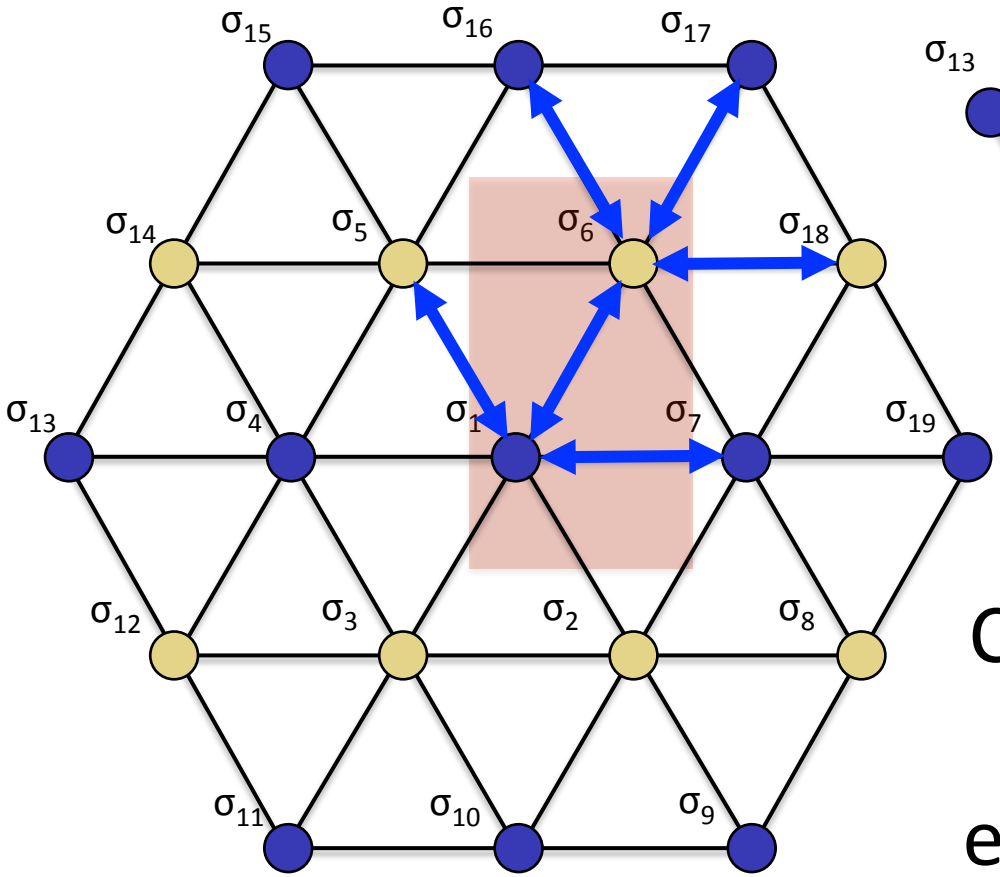
Average site occupation is

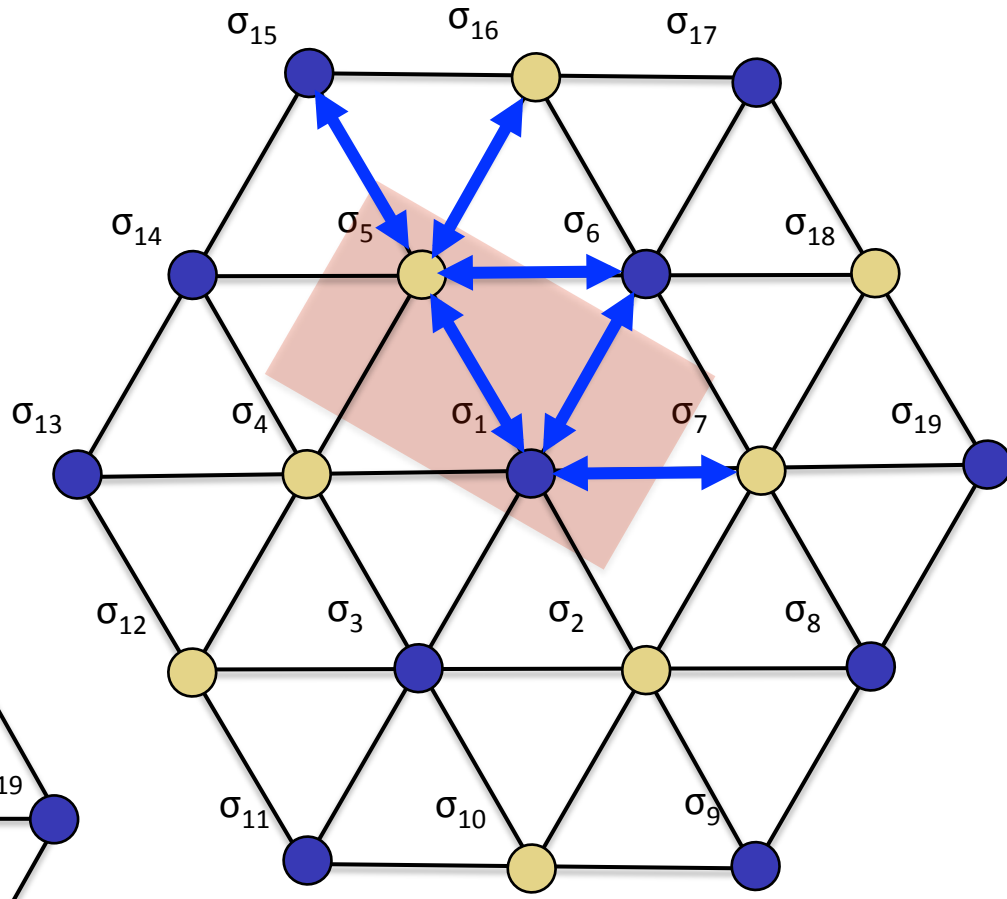
$\Gamma_1=(\sigma_1+\sigma_6)/2=0$

Average NN pair occupation is

$\Gamma_2=(\sigma_1\sigma_7+\sigma_1\sigma_6+\sigma_1\sigma_5+\sigma_6\sigma_{18}+\sigma_6\sigma_{17}+\sigma_6\sigma_{16})/6 = 1/3$

Cluster Expansion Example





Correlations are identical for symmetrically equivalent configurations

The bspecs.json file

Specifications for generating clusters and the cluster expansion basis functions

bspecs.json

```
{
  "basis_functions" : {
    "site_basis_functions" : "occupation"
  },
  "orbit_branch_specs" : {
    "2" : {"max_length" : 4.01},
    "3" : {"max_length" : 3.01}
  },
  "orbit_specs" : [
    {
      "coordinate_mode" : "Direct",
      "prototype" : [
        [ 0.000000000000, 0.000000000000, 0.000000000000 ],
        [ 1.000000000000, 0.000000000000, 0.000000000000 ],
        [ 2.000000000000, 0.000000000000, 0.000000000000 ],
        [ 3.000000000000, 0.000000000000, 0.000000000000 ]
      ],
      "include_subclusters" : true
    },
    {
      "coordinate_mode" : "Direct",
      "prototype" : [
        [ 0.000000000000, 0.000000000000, 0.000000000000 ],
        [ 0.000000000000, 1.000000000000, 0.000000000000 ],
        [ 0.000000000000, 0.000000000000, 1.000000000000 ],
        [ 1.000000000000, 1.000000000000, 1.000000000000 ]
      ],
      "include_subclusters" : true
    }
  ]
}
```

The clust.json file

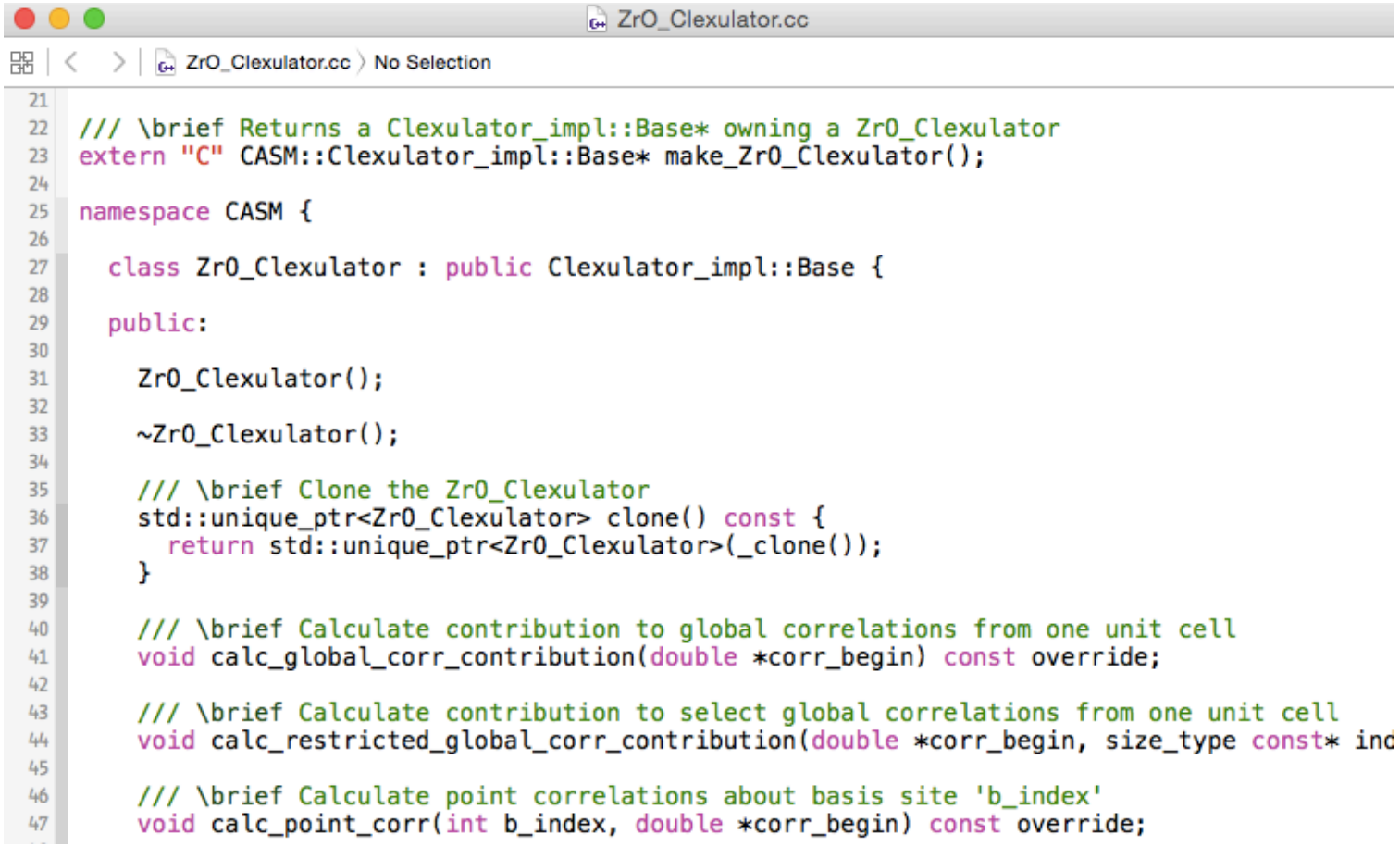
Lists one example (“prototype”) of each cluster type

clust.json


```
{
  "branches" : [
    {
      "orbits" : [
        {
          "prototype" : {
            "max_length" : 0.000000000000,
            "min_length" : 0.000000000000,
            "sites" : [ ]
          }
        }
      ]
    },
    {
      "orbits" : [
        {
          "prototype" : {
            "max_length" : 0.000000000000,
            "min_length" : 0.000000000000,
            "sites" : [
              [ 3, 0, 0, 0 ]
            ]
          }
        }
      ]
    },
    {
      "orbits" : [
        {
          "prototype" : {
            "max_length" : 2.584339170000,
            "min_length" : 2.584339170000,
            "sites" : [
              [ 3, 0, 0, 0 ],
              [ 2, 0, 0, 0 ]
            ]
          }
```

The clexulator (ZrO_Clexulator.cc)

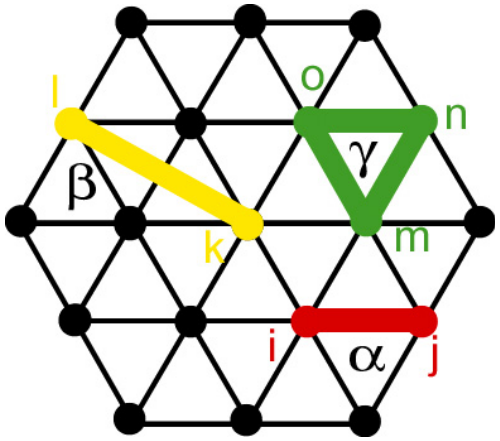
Generated code for evaluating basis functions




```
21
22  /// \brief Returns a Clexulator_impl::Base* owning a ZrO_Clexulator
23  extern "C" CASM::Clexulator_impl::Base* make_ZrO_Clexulator();
24
25  namespace CASM {
26
27    class ZrO_Clexulator : public Clexulator_impl::Base {
28
29    public:
30
31      ZrO_Clexulator();
32
33      ~ZrO_Clexulator();
34
35      /// \brief Clone the ZrO_Clexulator
36      std::unique_ptr<ZrO_Clexulator> clone() const {
37        return std::unique_ptr<ZrO_Clexulator>(_clone());
38      }
39
40      /// \brief Calculate contribution to global correlations from one unit cell
41      void calc_global_corr_contribution(double *corr_begin) const override;
42
43      /// \brief Calculate contribution to select global correlations from one unit cell
44      void calc_restricted_global_corr_contribution(double *corr_begin, size_type const* inc
45
46      /// \brief Calculate point correlations about basis site 'b_index'
47      void calc_point_corr(int b_index, double *corr_begin) const override;
```

Generated code for evaluating basis functions

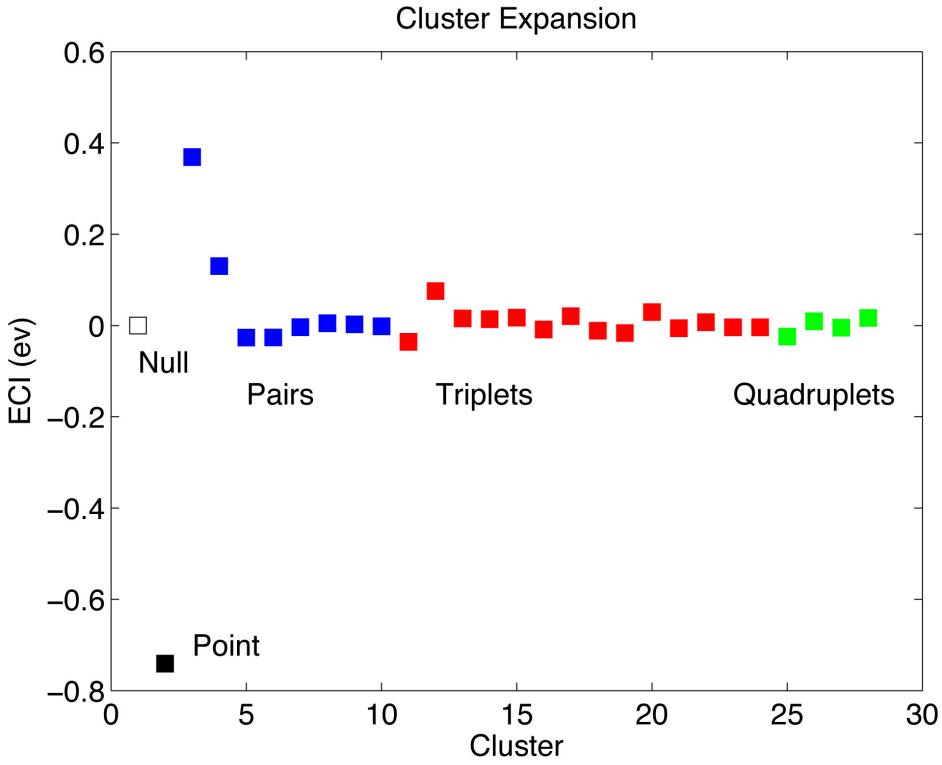
Cluster expansion for hcp Zr-O



$$\Phi_\alpha = \sigma_i \sigma_j$$

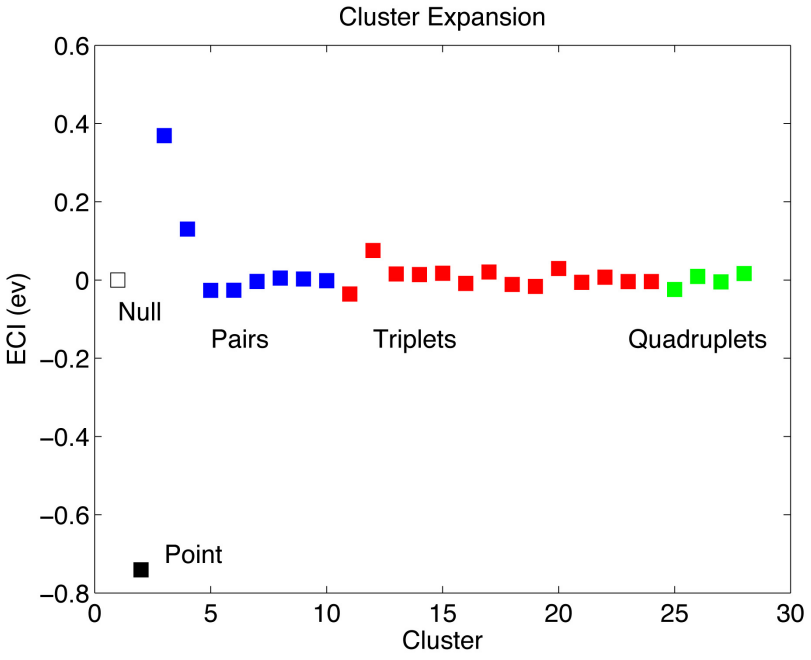
$$\Phi_\beta = \sigma_l \sigma_k$$

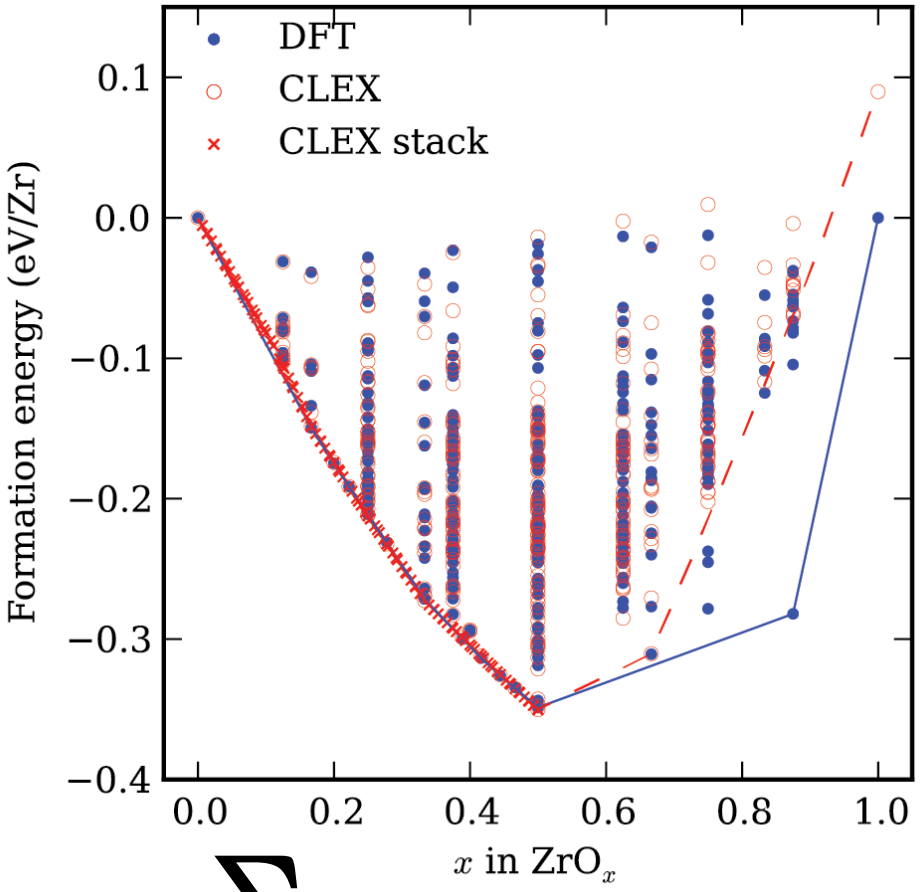
$$\Phi_\gamma = \sigma_m \sigma_n \sigma_o$$



$$E(\vec{\sigma}) = V_o + \sum_i V_i \sigma_i + \sum_{i,j} V_{i,j} \sigma_i \sigma_j + \sum_{i,j,k} V_{i,j,k} \sigma_i \sigma_j \sigma_k + \ldots$$

Cluster expansion for hcp Zr-O

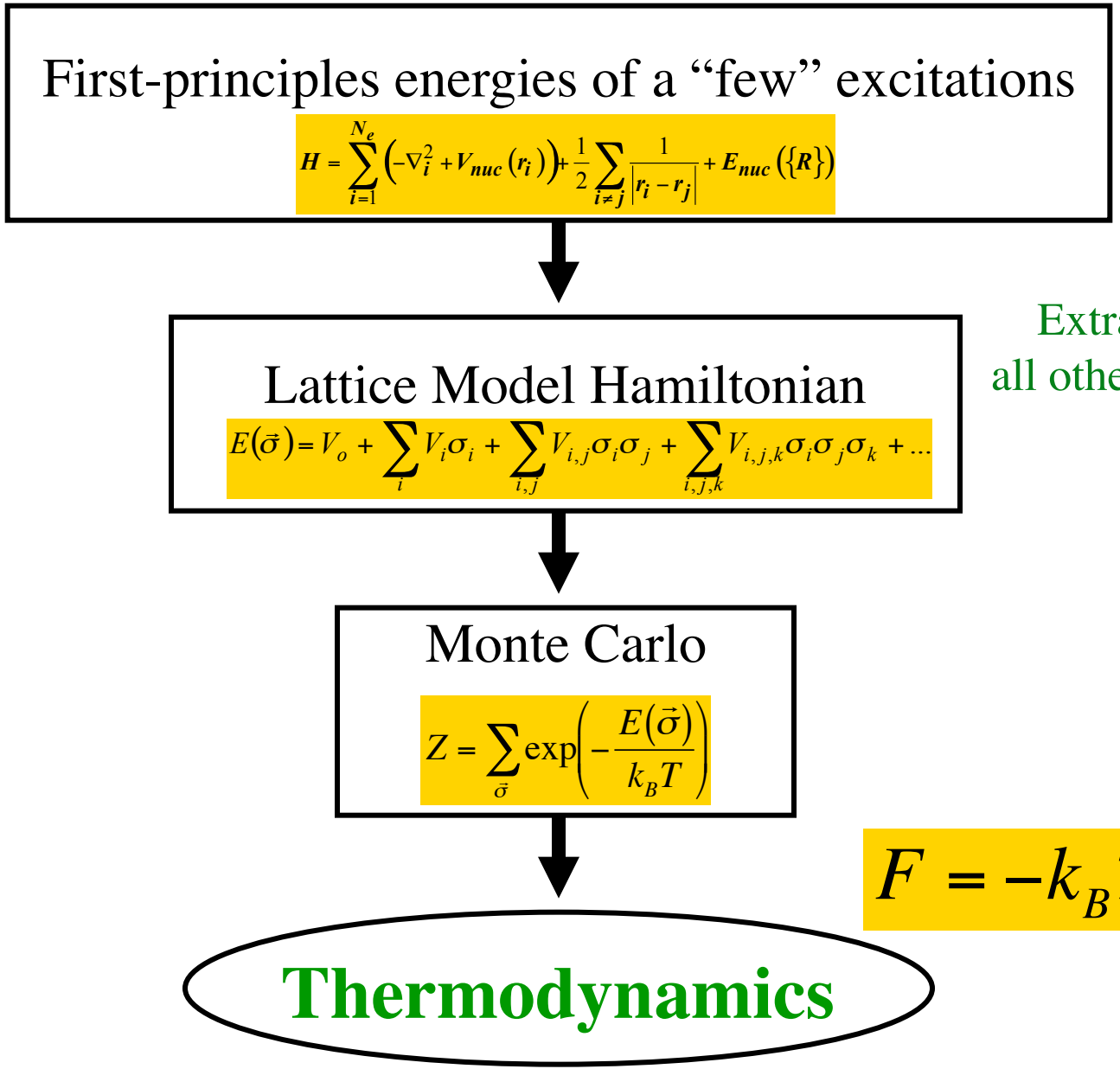




$$E(\vec{\sigma}) = V_o + \sum_i V_i \sigma_i + \sum_{i,j} V_{i,j} \sigma_i \sigma_j + \sum_{i,j,k} V_{i,j,k} \sigma_i \sigma_j \sigma_k + \ldots$$

Puchala and Van der Ven, Phys. Rev. B **88** 094108 (2013).

CASM: Clusters Approach to Statistical Mechanics



Summary

- Crystal structure identification & specification
- Symmetry analysis
- Using CASM to manage data
  - Configuration selections and query syntax
  - Maintaining sanity with cluster expansion profiles
- Cluster expansion basics
  - Configuration enumeration
  - Cluster expansion basis sets & correlation evaluation

Tomorrow (with Brian Puchala):

- Parameterizing cluster expansions
- Predicting thermodynamic properties with grand canonical Monte Carlo
- Free energy integration and phase diagram construction