

PRISMS Center

Center for PRedictive Integrated Structural Materials Science

CASM Tutorial – Section 1

Brian Puchala, John C. Thomas, Anton Van der Ven

Anirudh Natarajan, John Goiri, Min-Hua Chen,
Max Radin, Jonathon Bechtel, Elizabeth Decolvenaere, Anna Belak,
Naga Sri Harsha Gunda, Sanjeev Kolli



TMS ICME 2017

Thurs. May 24, 2017

Website: prisms-center.org



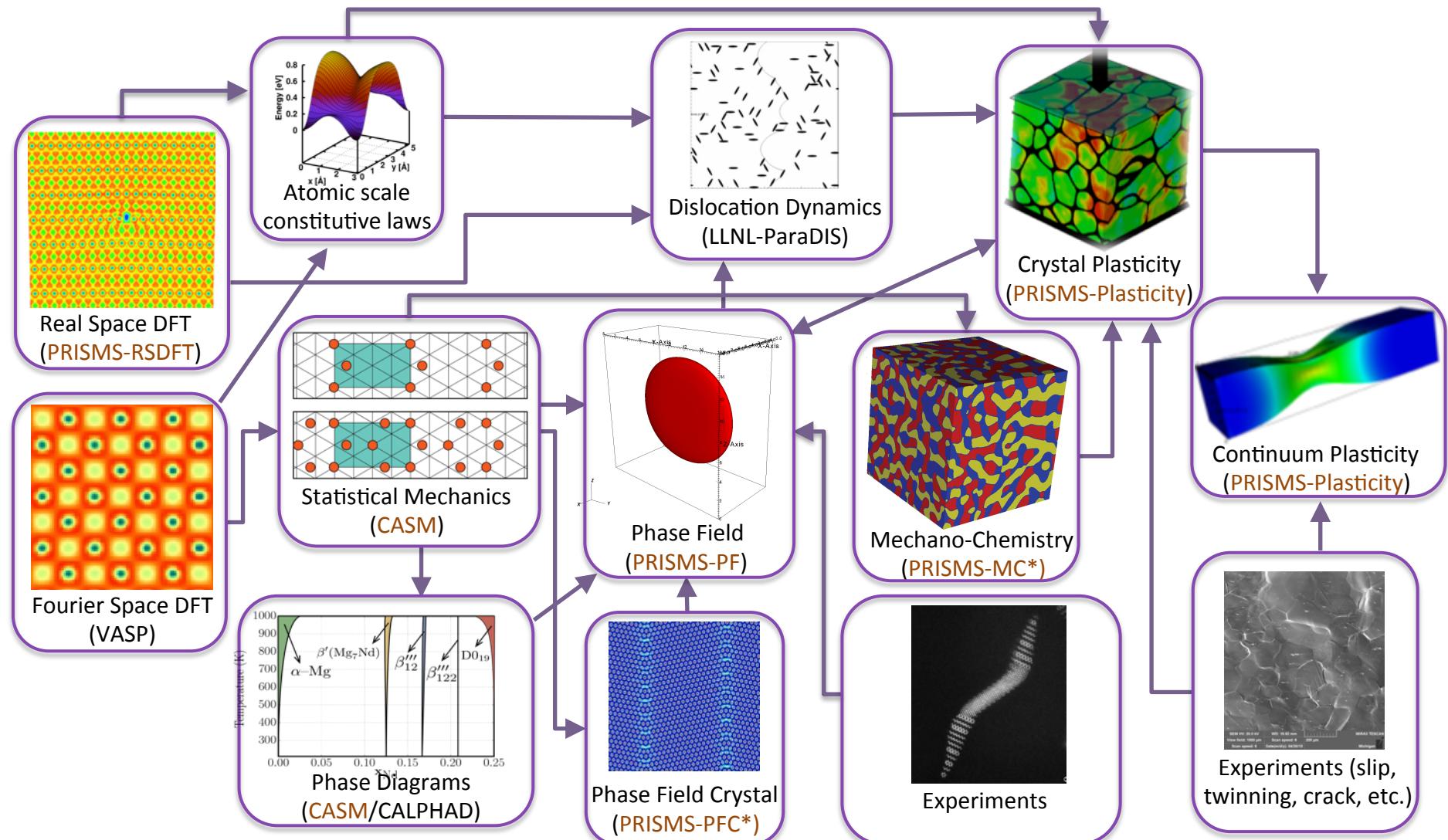
U.S. DEPARTMENT OF
ENERGY

Office of
Science

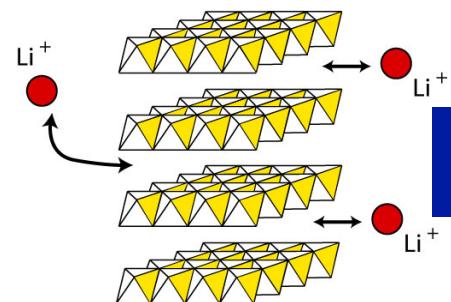
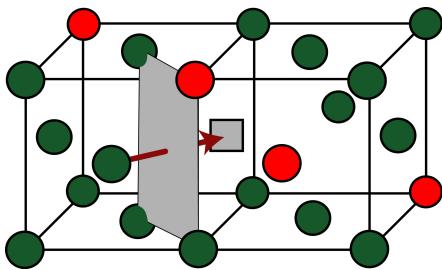
Center for PRedictive Integrated
Structural Materials Science

PRISMS

PRISMS Center Integrated Framework



Electronic structure

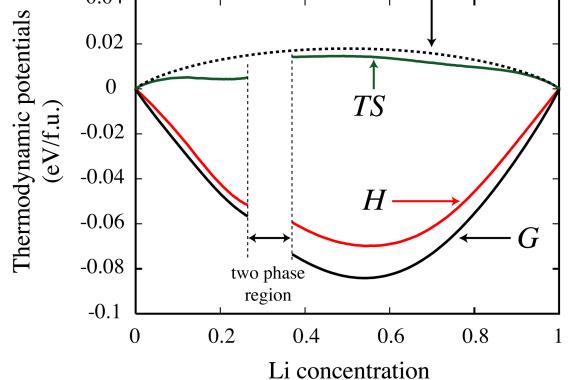


Statistical Mechanics

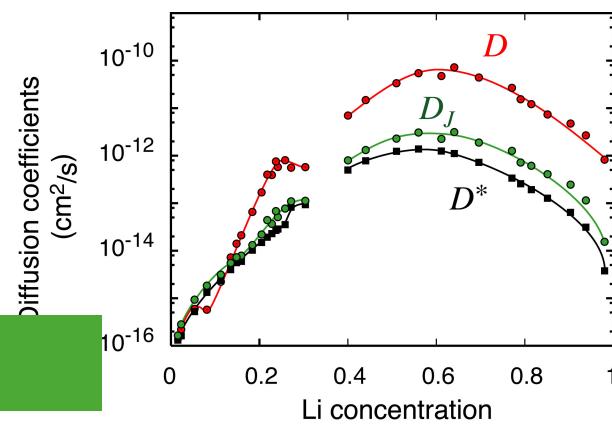
Thermodynamics

Kinetics

Solid Mechanics



Linking atomistic descriptions to phenomenological models of materials



Phenomenological descriptions

Electronic structure

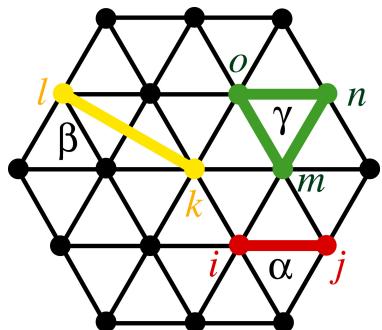
$$\left[-\nabla_i^2 + V_{ion}(r_i) + \int \frac{\rho(r')}{r - r'} dr' + V_{xc}(\rho(r_i)) \right] \phi_i(r_i) = \epsilon_i \phi_i(r_i)$$

Via rigorous and
arbitrarily extendible
cluster expansion
Hamiltonians

Cluster Expansion Hamiltonians

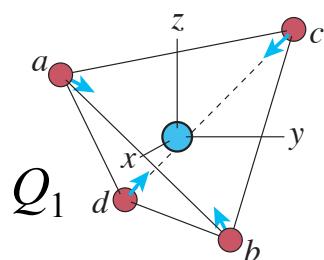
$$E(\vec{\sigma}) = V_o + \sum_i V_i \sigma_i + \sum_{i,j} V_{i,j} \sigma_i \sigma_j + \sum_{i,j,k} V_{i,j,k} \sigma_i \sigma_j \sigma_k + \dots$$

$$E(\vec{u}_1, \dots, \vec{u}_i, \dots, \vec{u}_N) = V_0 + \sum_{\alpha,m} V_m^{(\alpha)} \Phi_m^{(\alpha)}(\vec{Q}^{(\alpha)}) + \sum_{\beta,n} V_n^{(\beta)} \Phi_n^{(\beta)}(\vec{Q}^{(\beta)}) + \sum_{\gamma,p} V_p^{(\gamma)} \Phi_p^{(\gamma)}(\vec{Q}^{(\gamma)}) + \dots$$



Statistical Mechanics

$$F = -k_B T \ln Z \quad Z = \sum_s \exp\left(-\frac{E_s}{k_B T}\right)$$



Phenomenological descriptions

CASM Tutorial Outline

1:30 – 3:00:

- 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

3:30 – 5:00:

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

CASM: Clusters Approach to Statistical Mechanics

First-principles energies of a “few” excitations

$$H = \sum_{i=1}^{N_e} \left(-\nabla_i^2 + V_{nuc}(r_i) \right) + \frac{1}{2} \sum_{i \neq j} \frac{1}{|r_i - r_j|} + E_{nuc}(\{R\})$$



Lattice Model Hamiltonian

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



Monte Carlo

$$Z = \sum_{\vec{\sigma}} \exp\left(-\frac{E(\vec{\sigma})}{k_B T}\right)$$

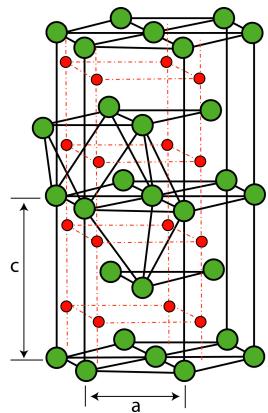


$$F = -k_B T \ln(Z)$$

Thermodynamics

Extrapolate to
all other excitations

Tutorial Project: Zr-O binary system



Zr:

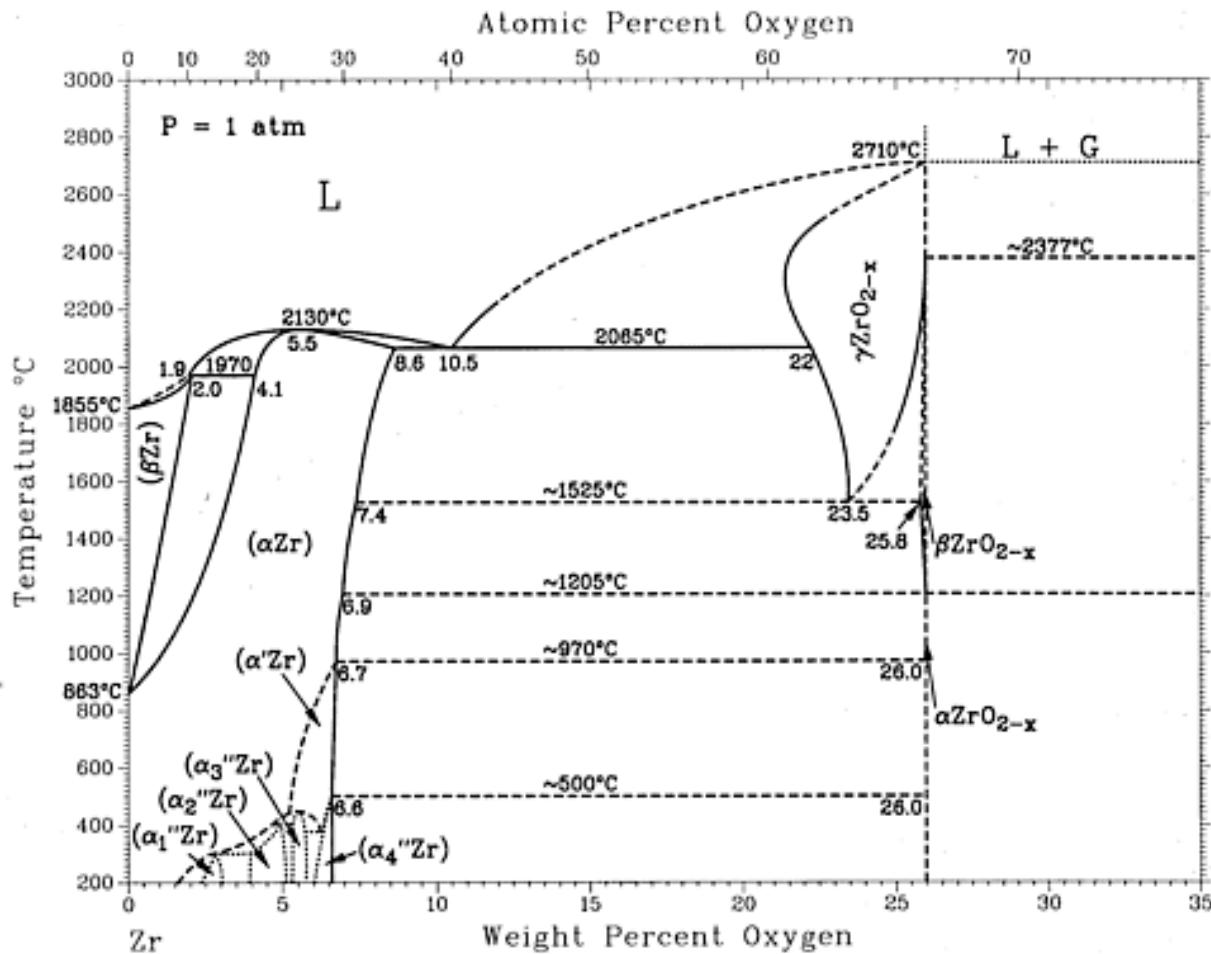
- Very high O solubility

ZrO_x

- Series of suboxide phases

ZrO_{2-x}

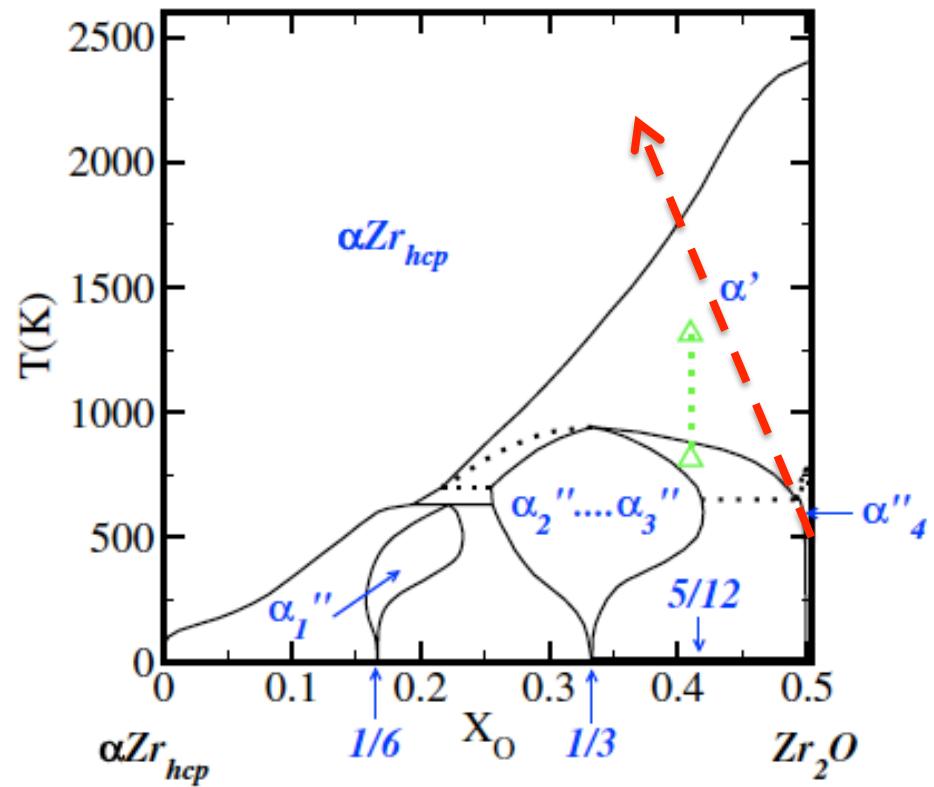
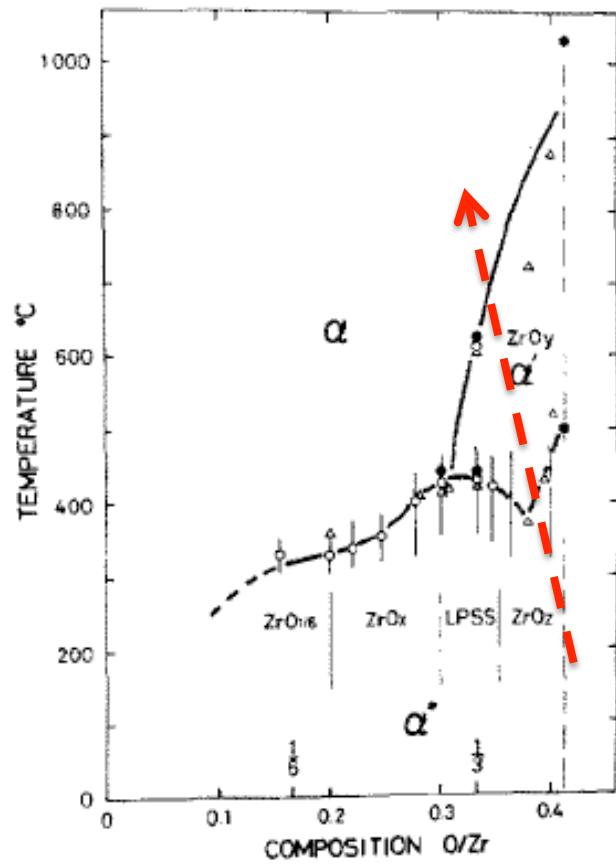
- Oxide



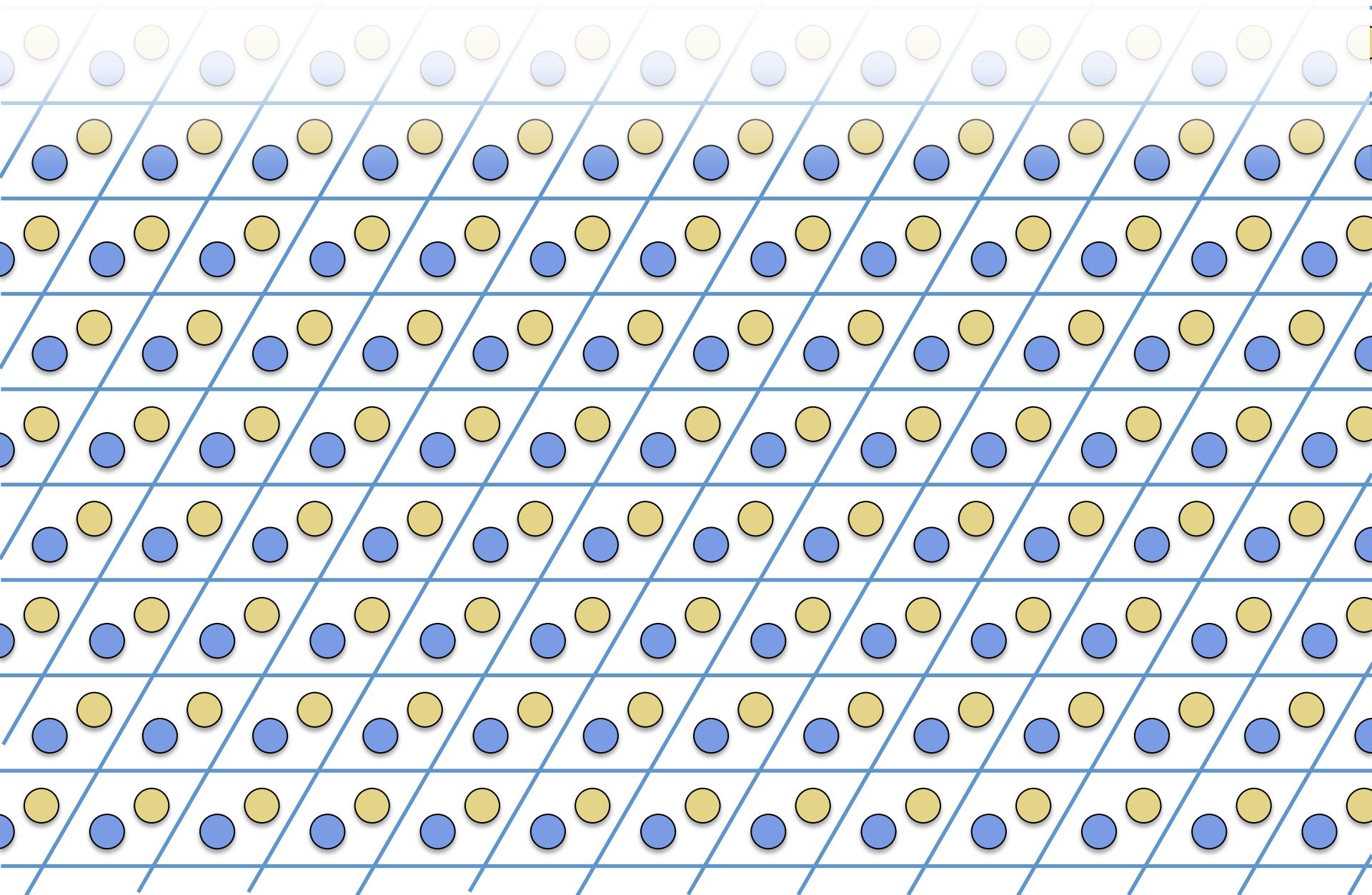
J.P. Abriata, J. Garcés, and R. Versaci, Bulletin of Alloy Phase Diagrams Vol. 7 No. 2 1986.

Nature of $\text{ZrO}_{1/2}$ disordering transitions?

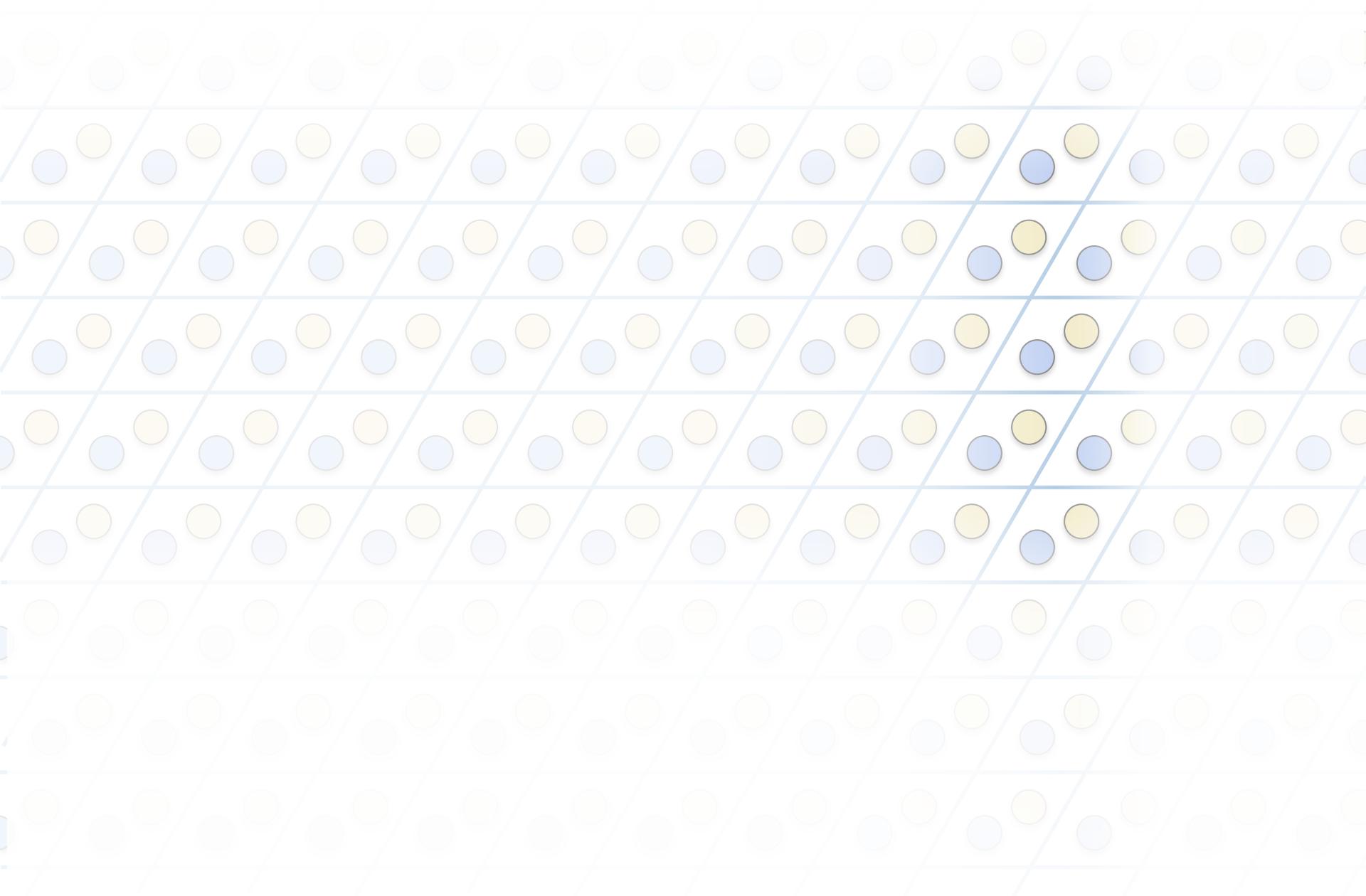
“the rearrangement of the oxygen atoms from uneven to even distribution on the neighboring interstice planes, and then [to a] random distribution in each plane.” or is low temperature transition first order?



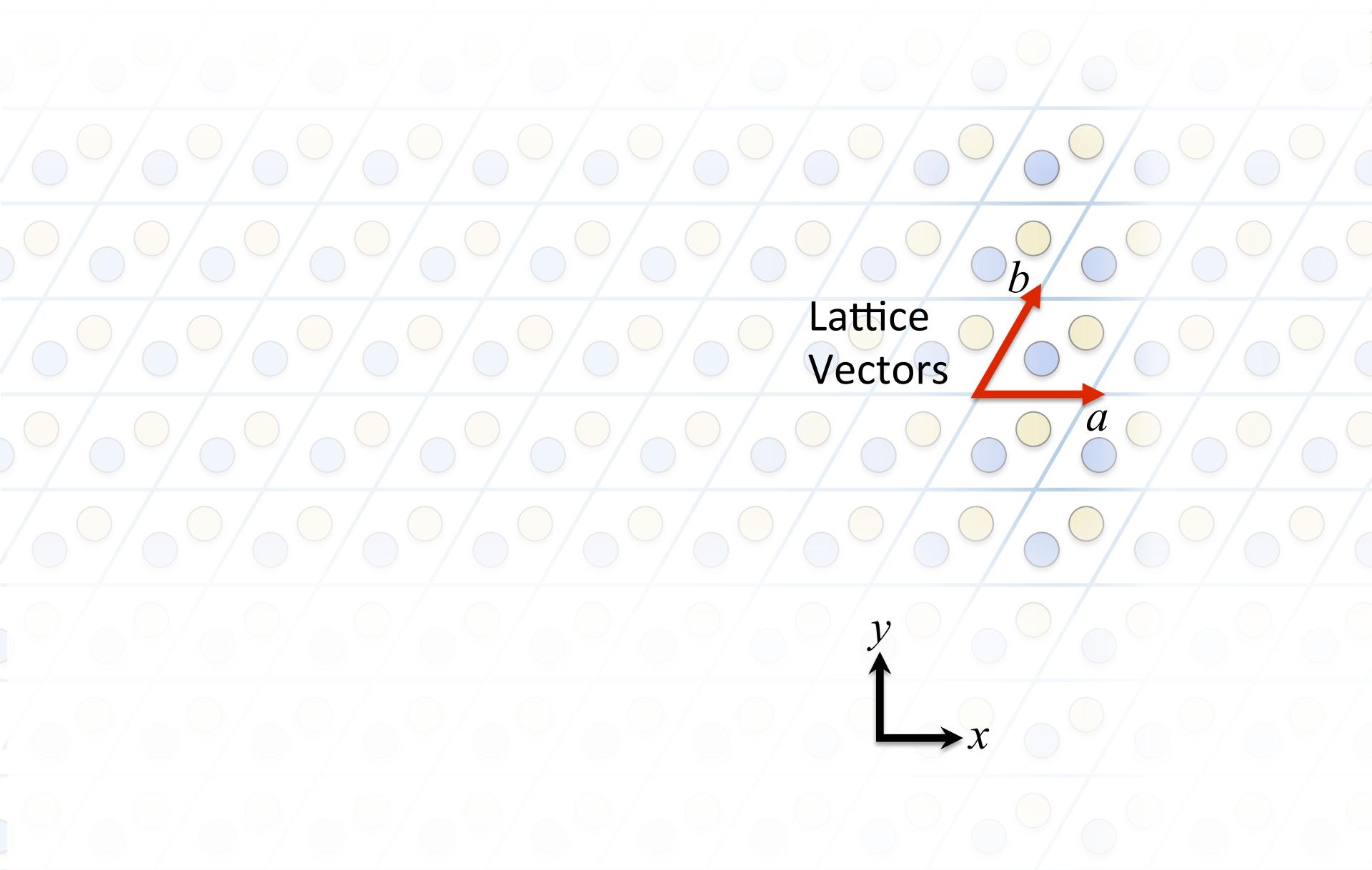
Problem Definition



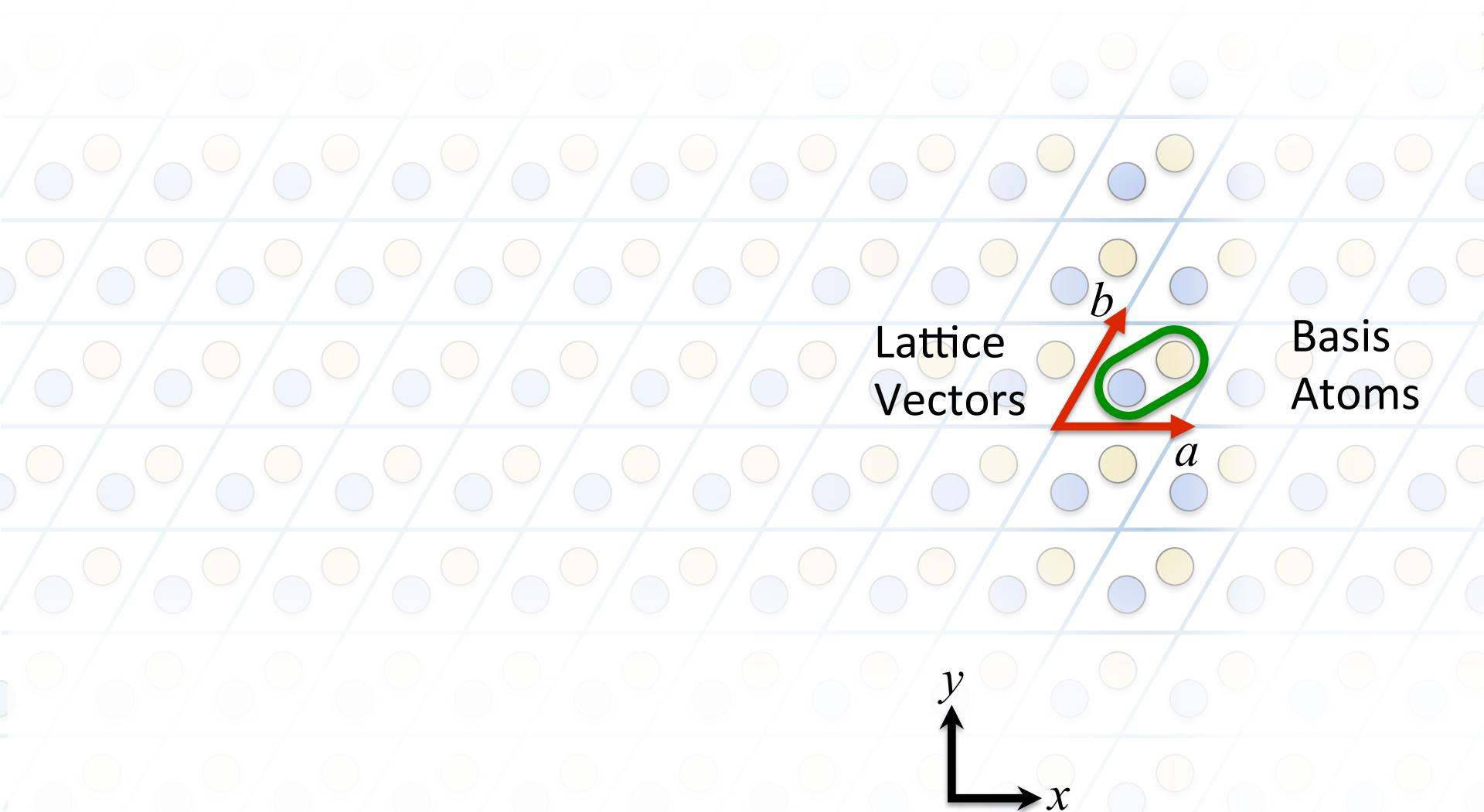
Problem Definition



Problem Definition



Problem Definition

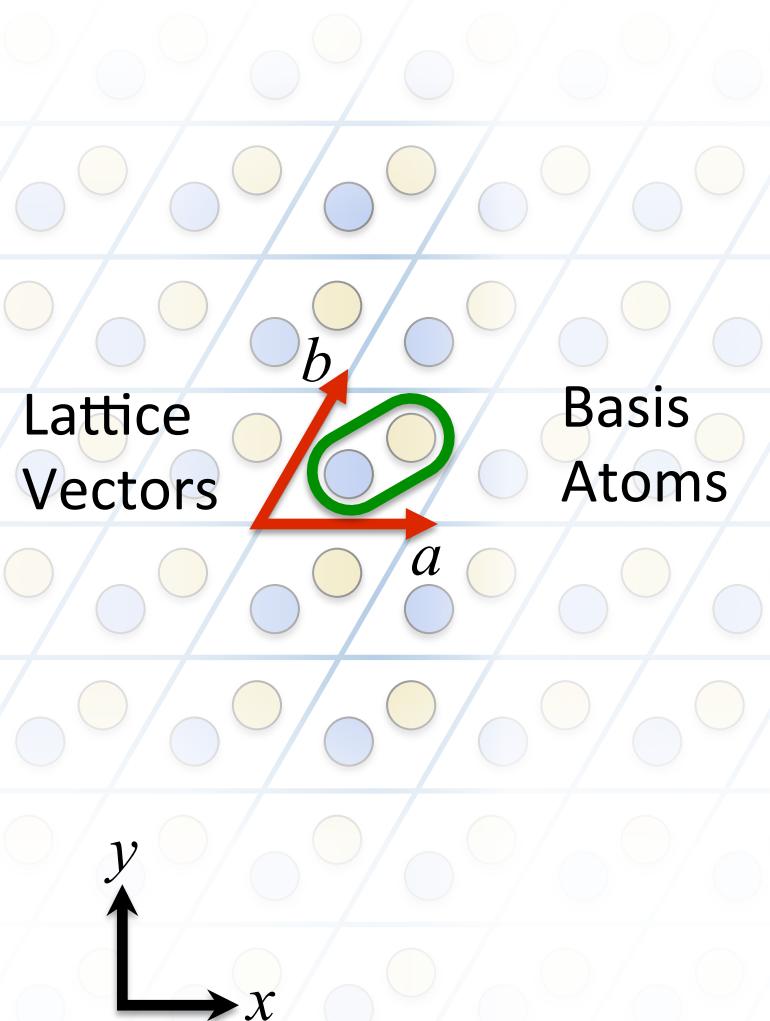


Problem Definition

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" ]  

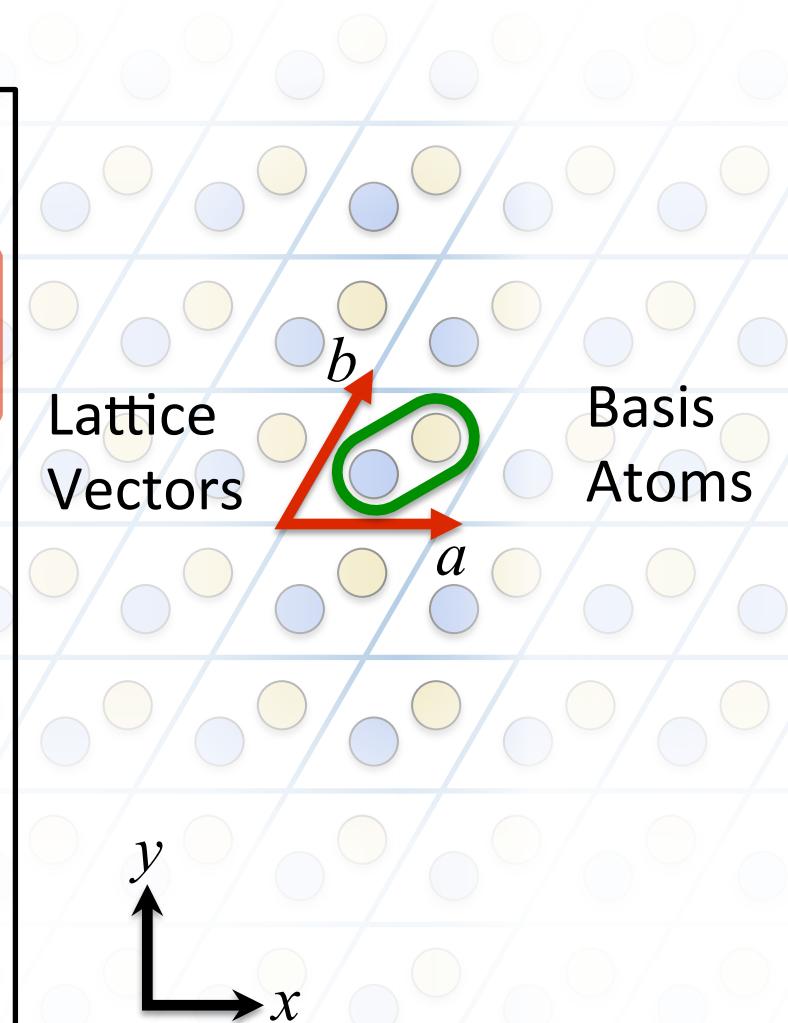
```



Problem Definition

prim.json:

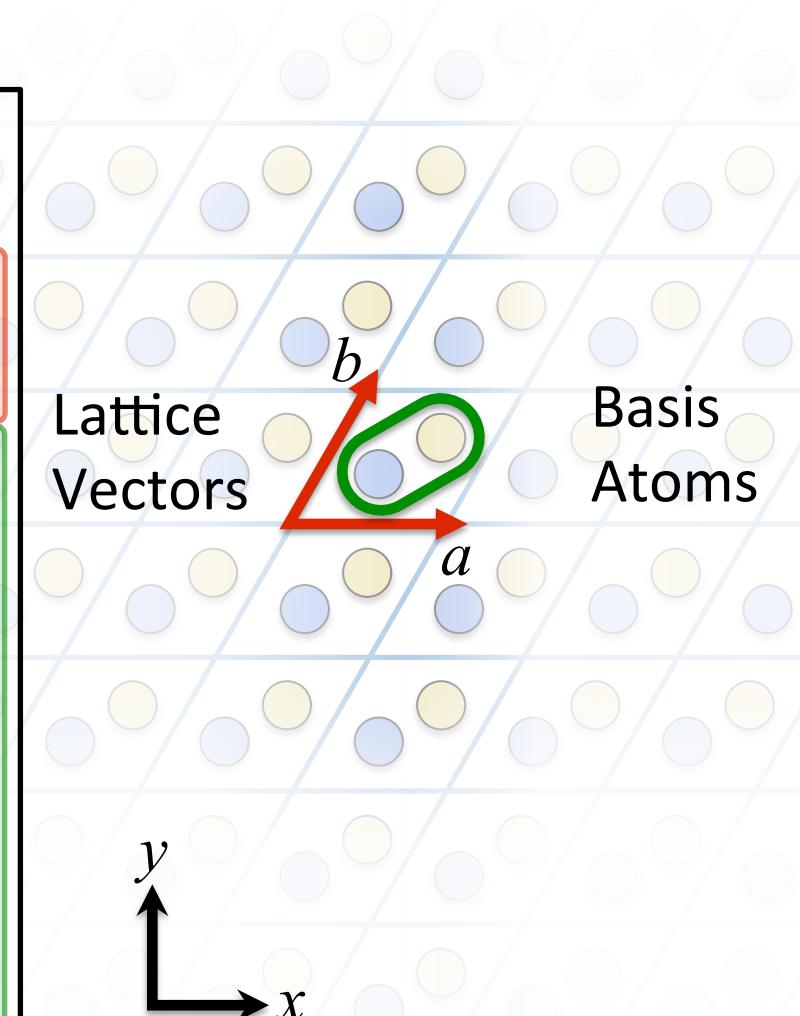
```
{  
  "title" : "ZrO",  
  "coordinate_mode" : "Fractional",  
  "description" : "HCP Zr with oct (O)",  
  "lattice_vectors" : [  
    a [ 3.233986860000, 0.000000000000, 0.000000000000 ],  
    b [ -1.616993430000, 2.800714770000, 0.000000000000 ],  
    c [ -0.000000000000, 0.000000000000, 5.168678340000 ]  
  ],  
  "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:

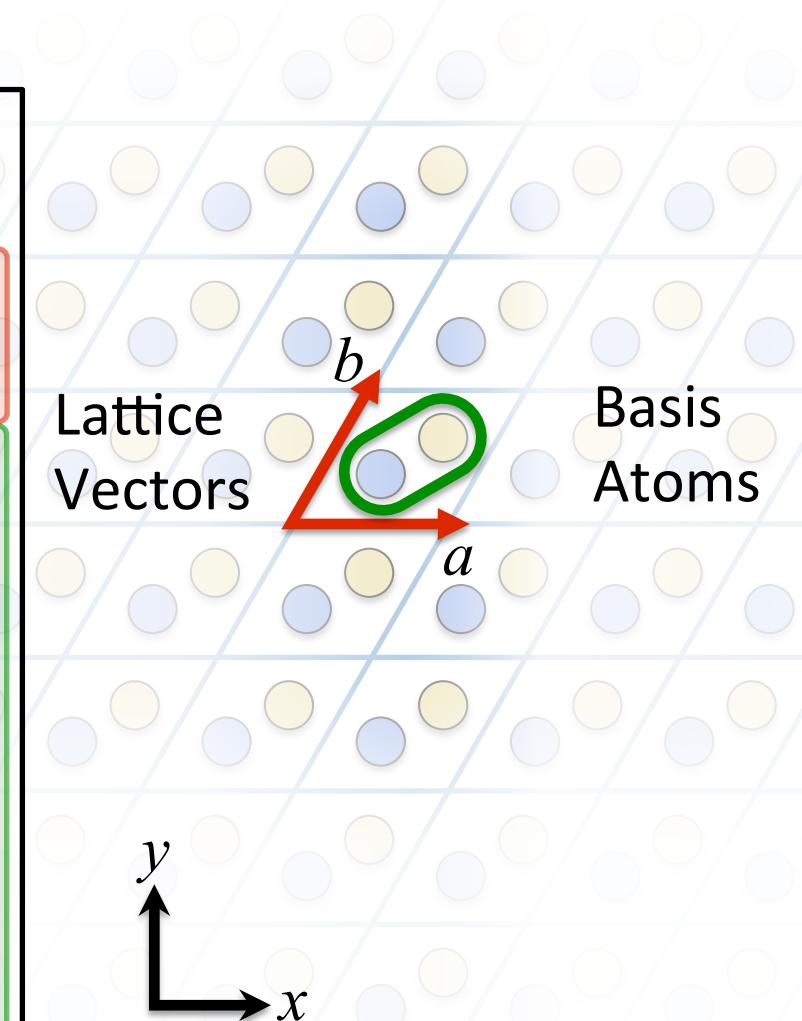
```
{  
  "title" : "ZrO",  
  "coordinate_mode" : "Fractional",  
  "description" : "HCP Zr with oct (O)",  
  "lattice_vectors" : [  
    a [ 3.233986860000, 0.000000000000, 0.000000000000 ],  
    b [ -1.616993430000, 2.800714770000, 0.000000000000 ],  
    c [ -0.000000000000, 0.000000000000, 5.168678340000 ]  
  ],  
  "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 ]  
  ],  
  "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

CASM: Getting started

```
[Adamant:~$ cd ~/Documents/Zr0
[Adamant:~/Documents/Zr0$ ls
prim.json
[Adamant:~/Documents/Zr0$ cat prim.json
{
    "title" : "Zr0",
    "coordinate_mode" : "Fractional",
    "description" : "HCP Zr with oct (0)",
    "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", "0" ]
        },
        {
            "coordinate" : [ 0.333333300000, 0.666666600000, 0.750000000000 ],
            "occupant_dof" : [ "Va", "0" ]
        }
    ]
}
[Adamant:~/Documents/Zr0$ casm init
*****
Initializing CASM project 'Zr0'
DONE

[Adamant:~/Documents/Zr0$ ls
basis_sets      cluster_expansions prim.json          symmetry
Adamant:~/Documents/Zr0$ █
```

Material definition

Initialization

Generated directories

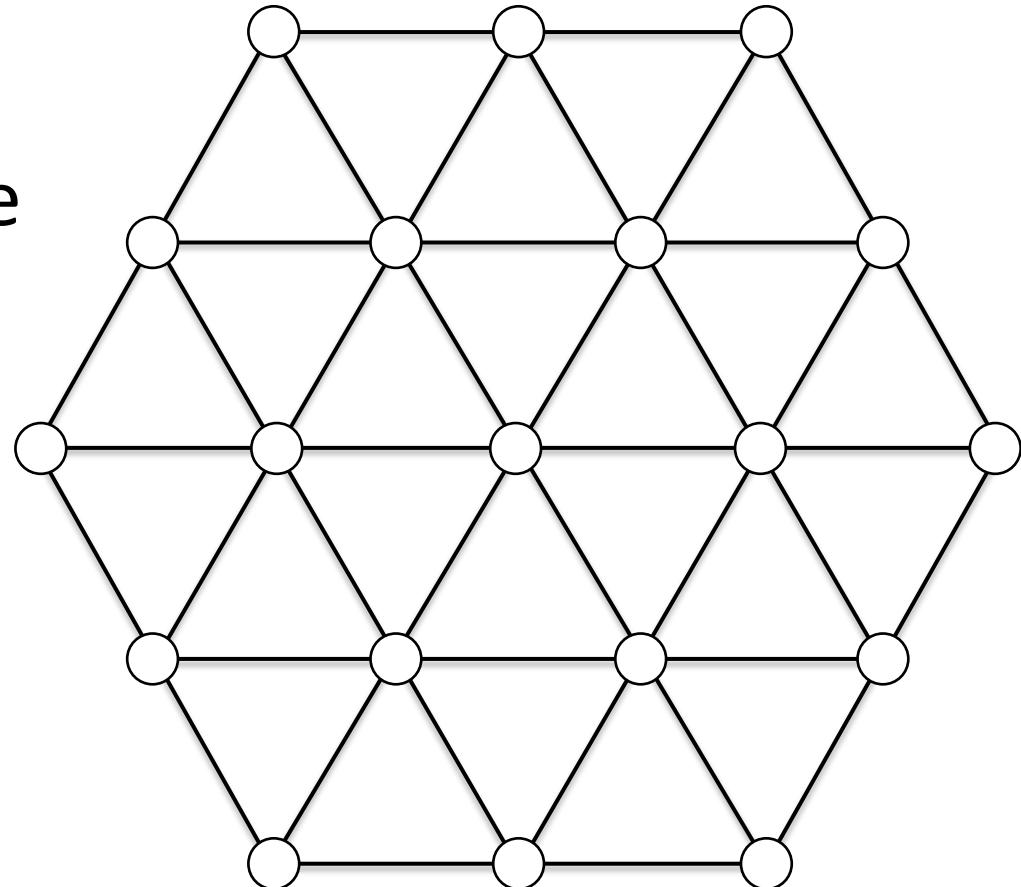
training_data

CASM: Getting started

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

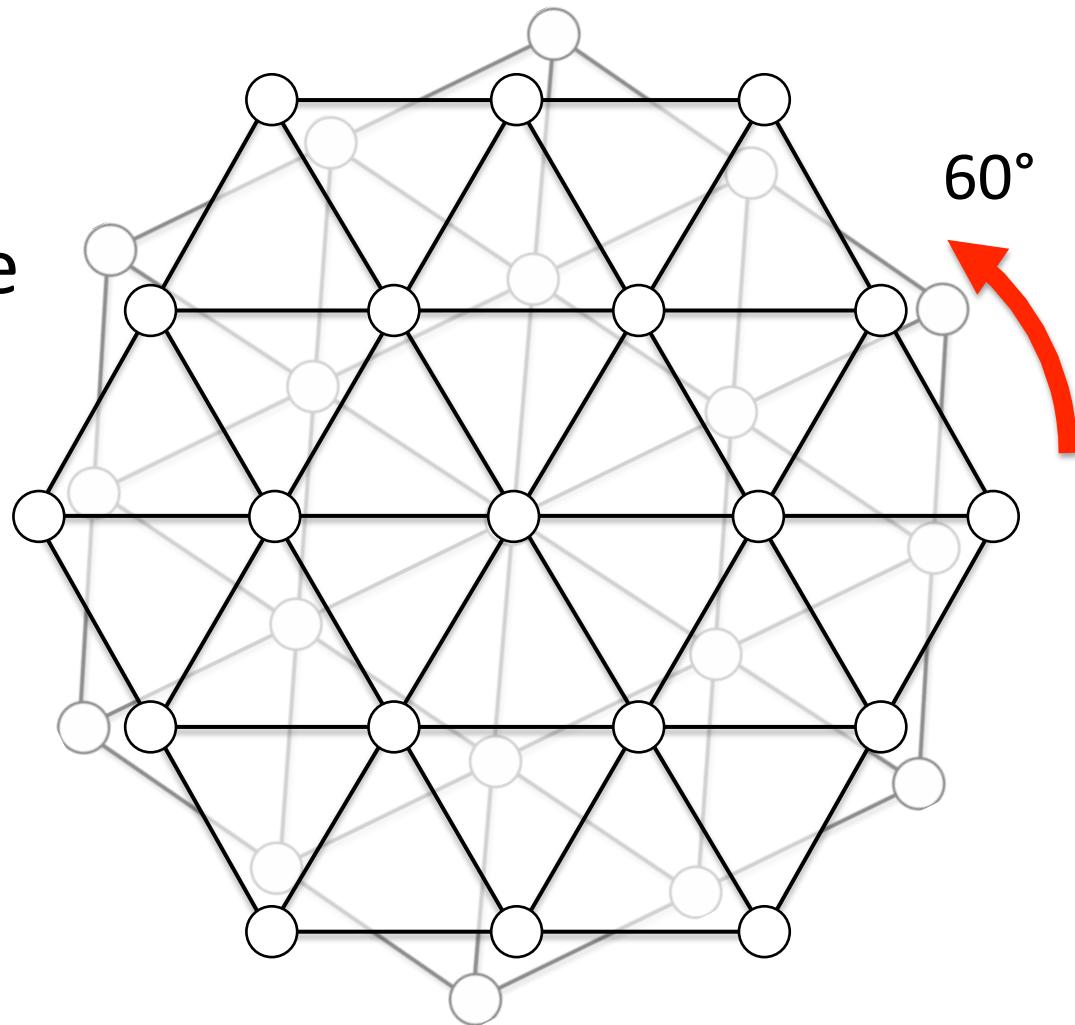
Crystal Symmetry

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



Crystal Symmetry

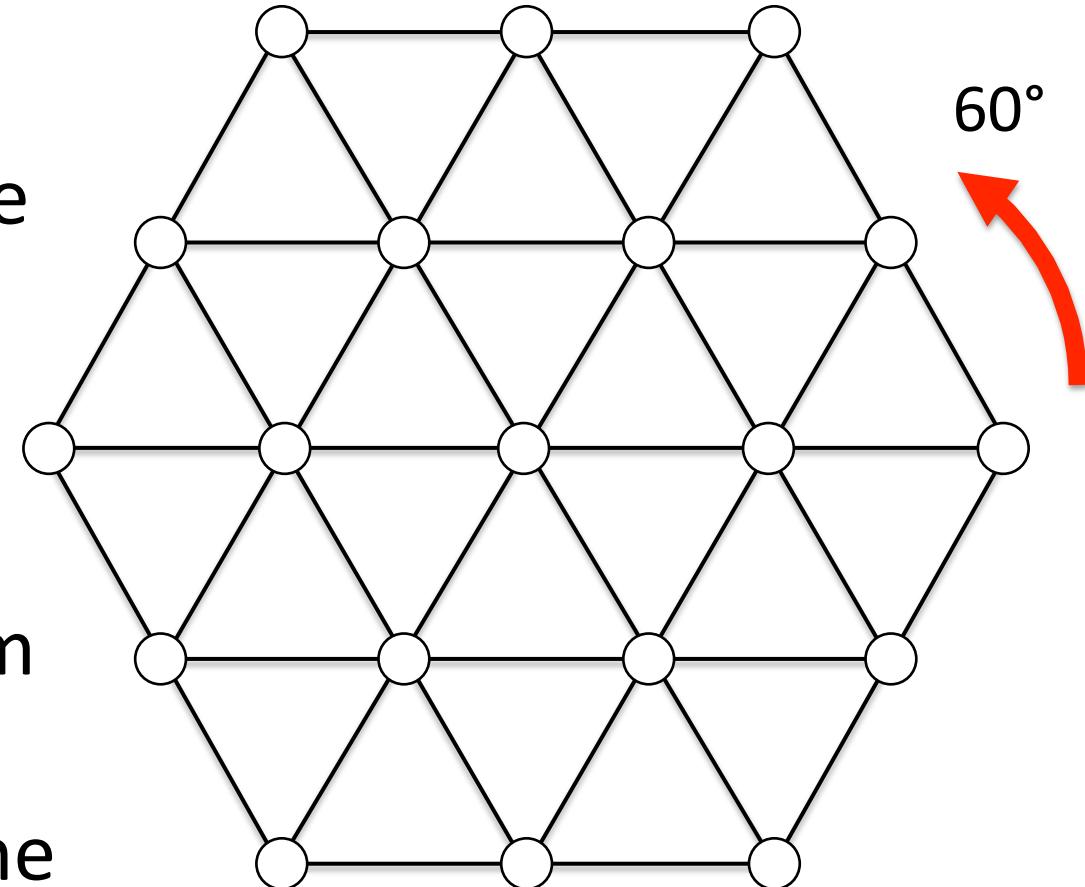
Apply rotation,
reflection, inversion,
and/or rotation to the
crystal.



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/Zr0$ ls symmetry
crystal_point_group.json
factor_group.json
lattice_point_group.json
[Adamant:~/Documents/Zr0$ casm sym
Generating lattice point group.
```

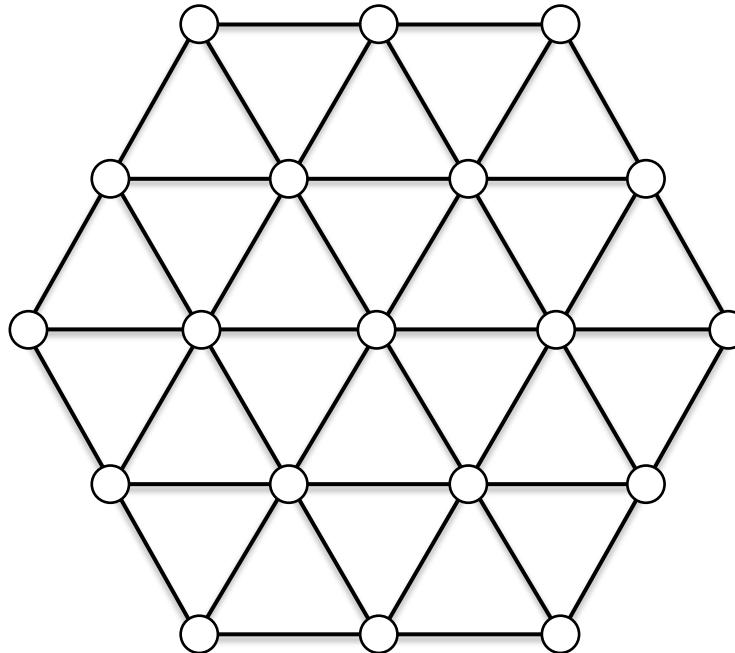
```
Lattice point group size: 24
Lattice point group is: D6h
```

```
Generating factor group.
```

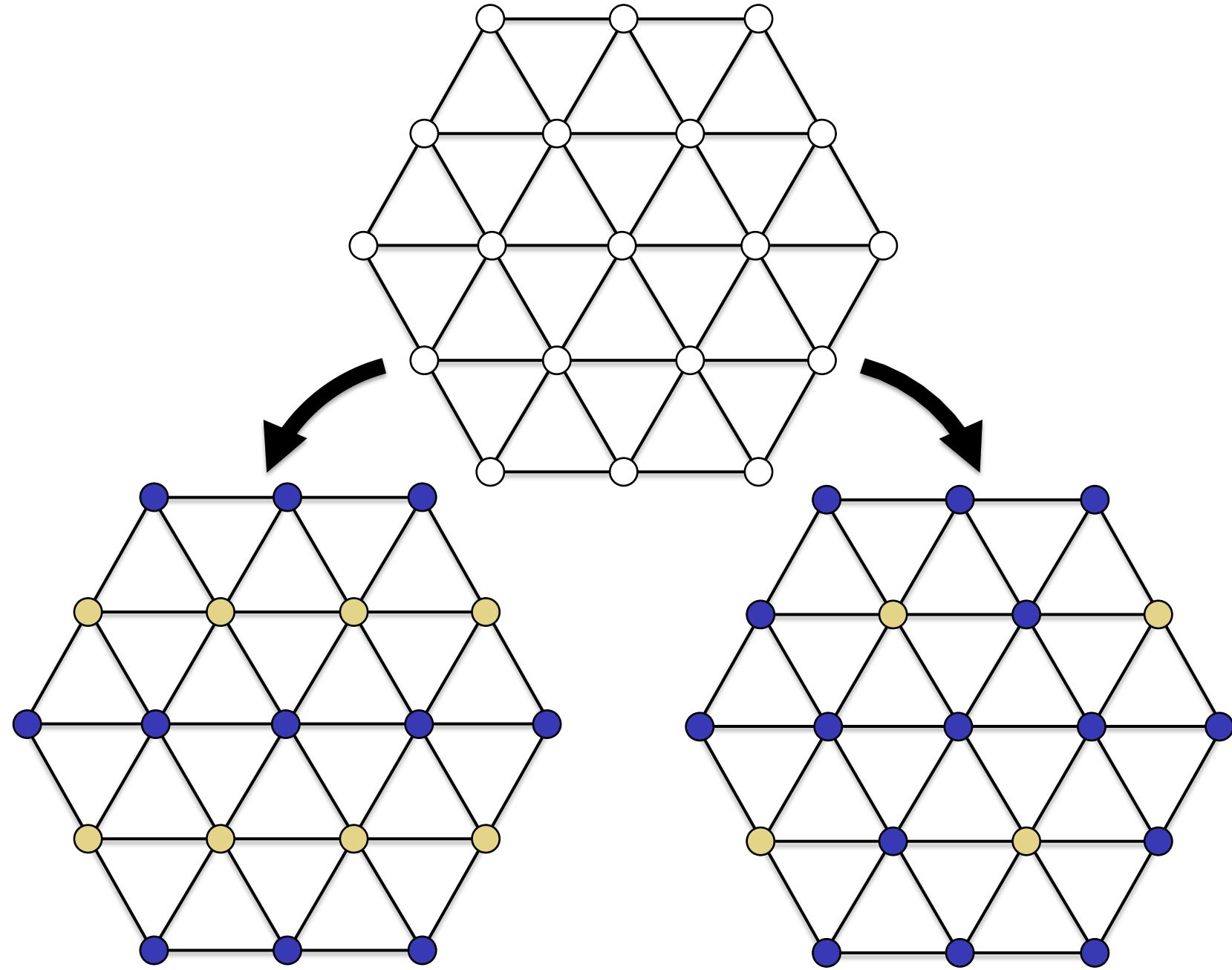
```
Factor group size: 24
Crystal point group is: D6h
```

```
Adamant:~/Documents/Zr0$ █
```

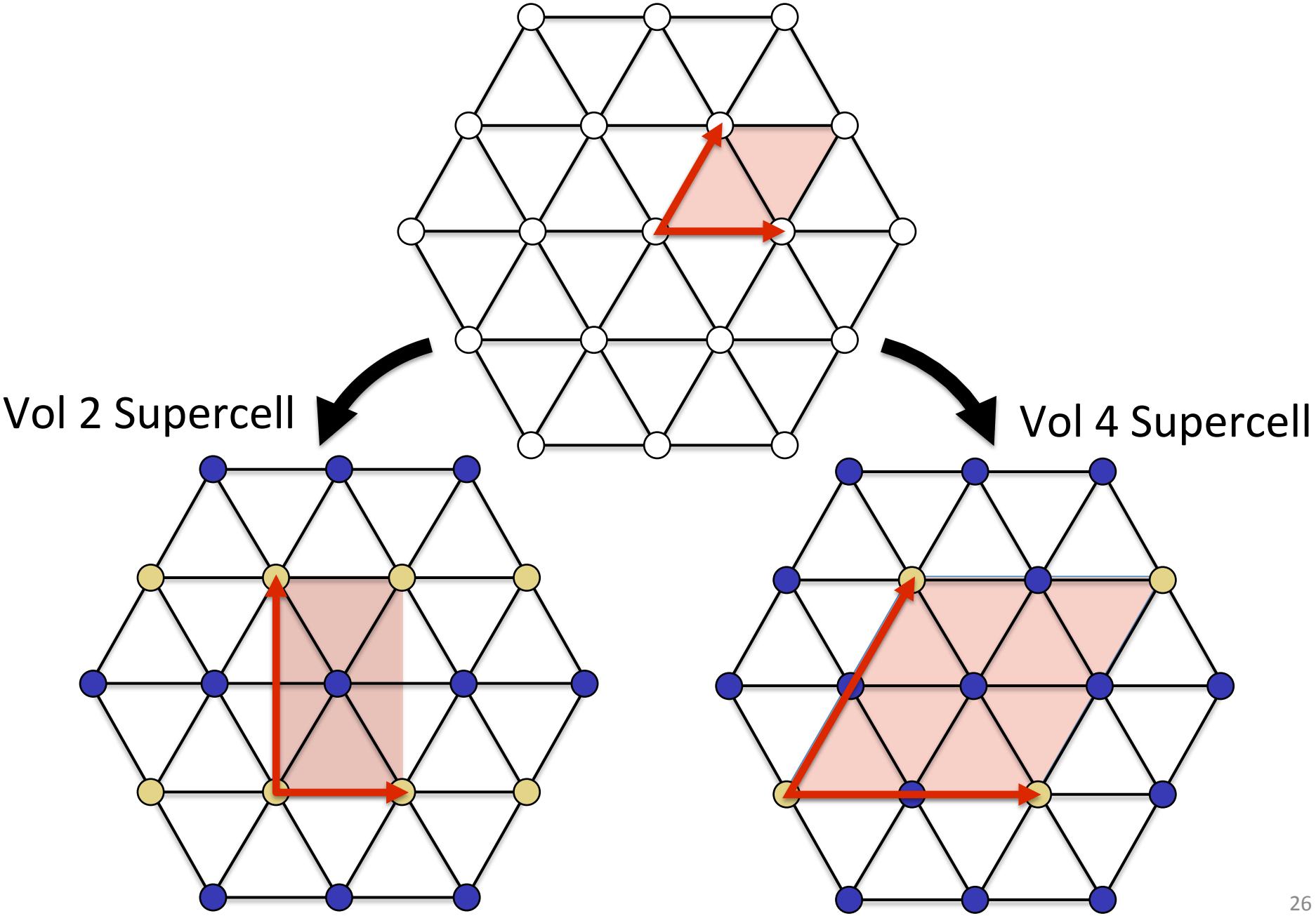
Crystal Configurations



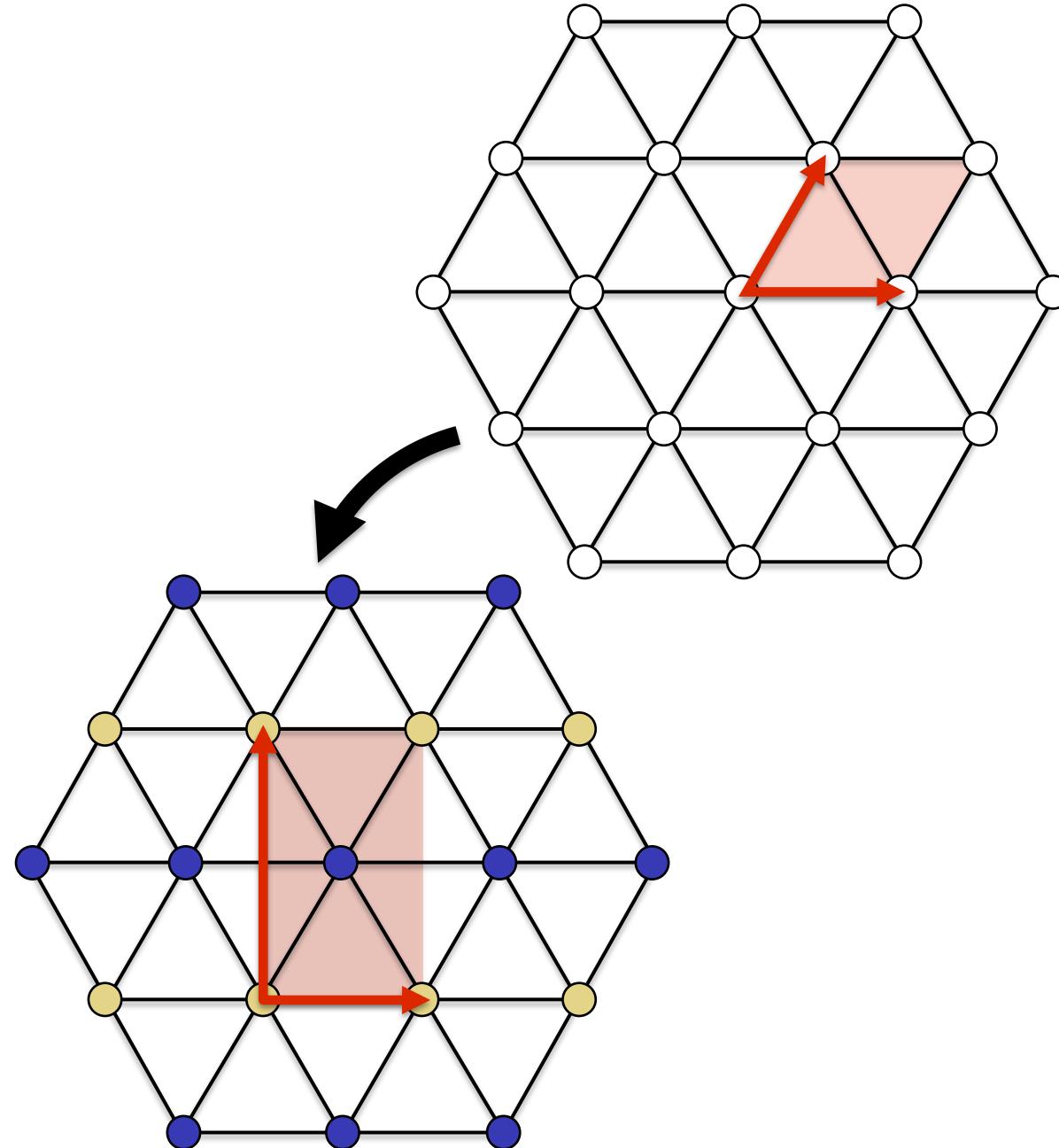
Crystal Configurations



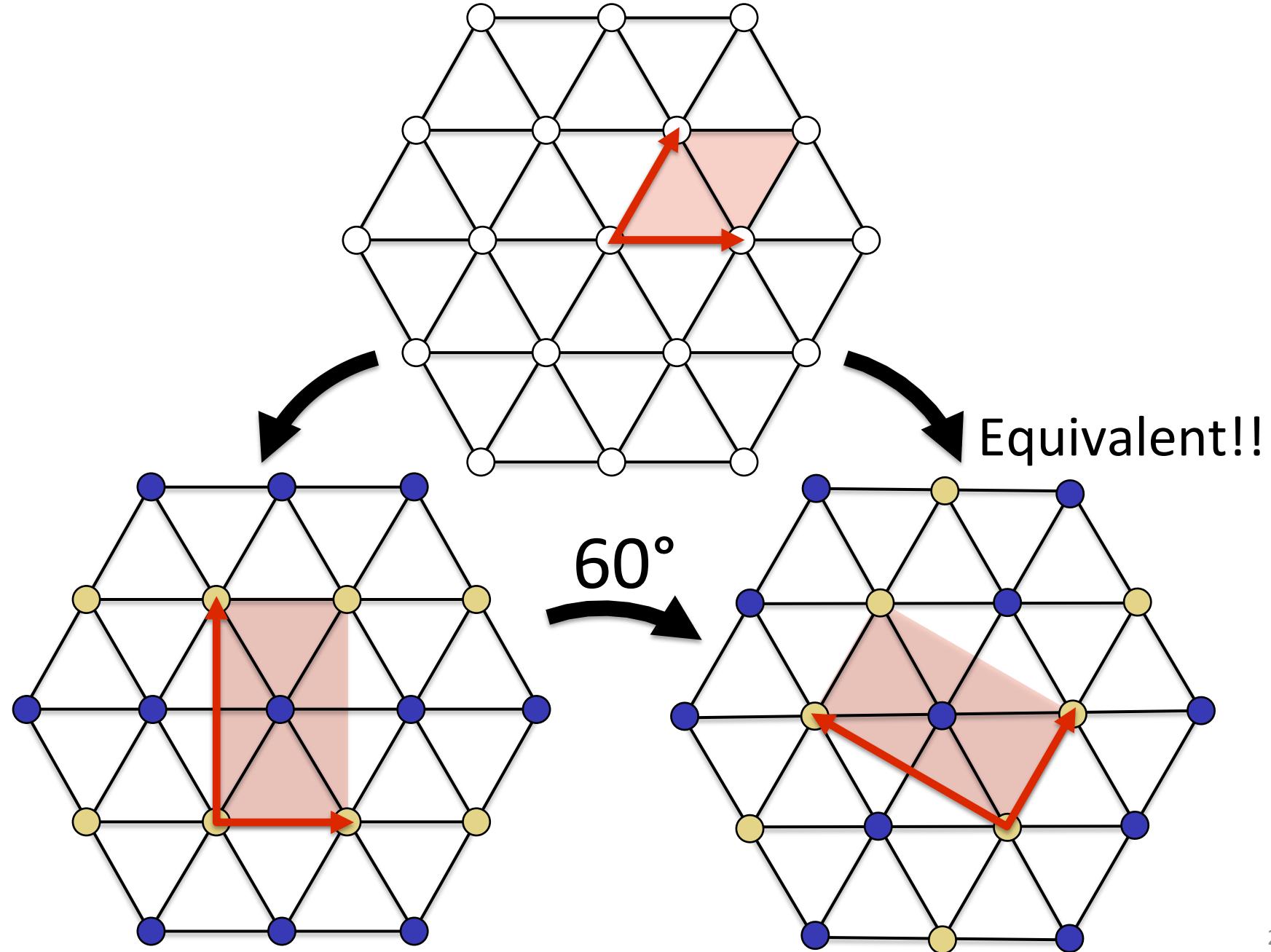
Crystal Configurations



Crystal Configurations

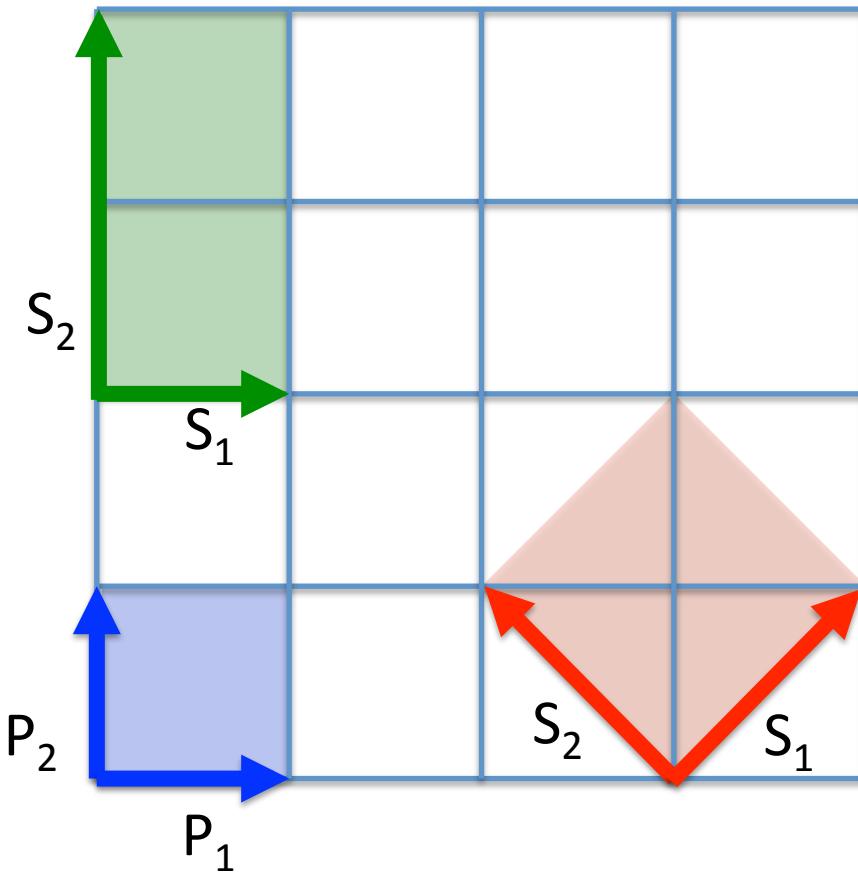


Crystal Configurations



Supercell Enumeration

$$\begin{aligned} S_1 &= P_1 \\ S_2 &= 2P_2 \end{aligned}$$



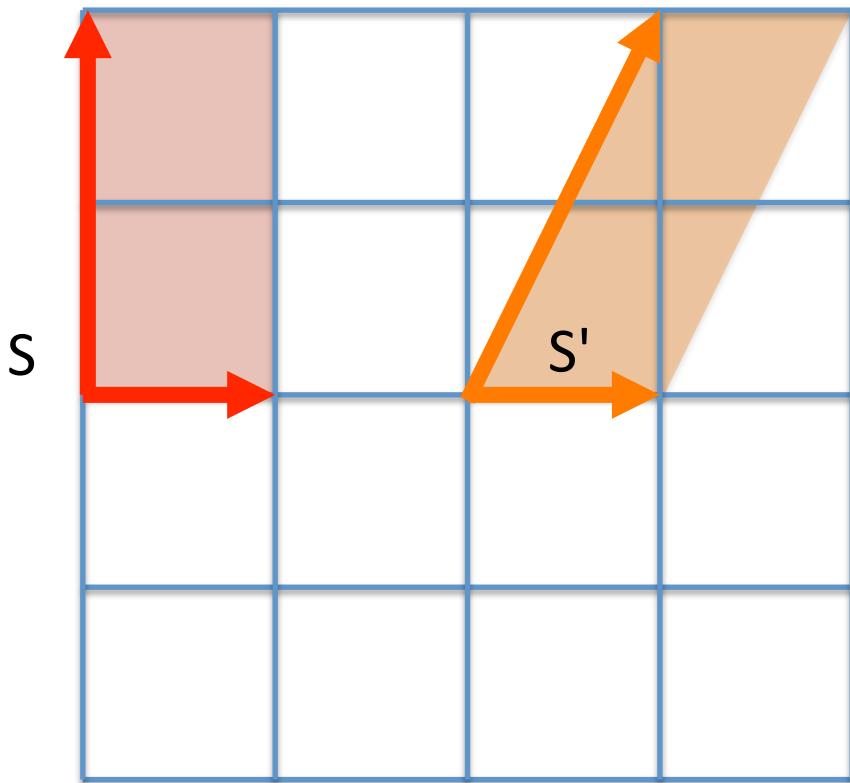
Represent lattice using column vector matrices:

$$S = PT$$

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

$$\begin{aligned} S_1 &= P_1 + P_2 \\ S_2 &= -P_1 + P_2 \end{aligned}$$

Supercell Enumeration

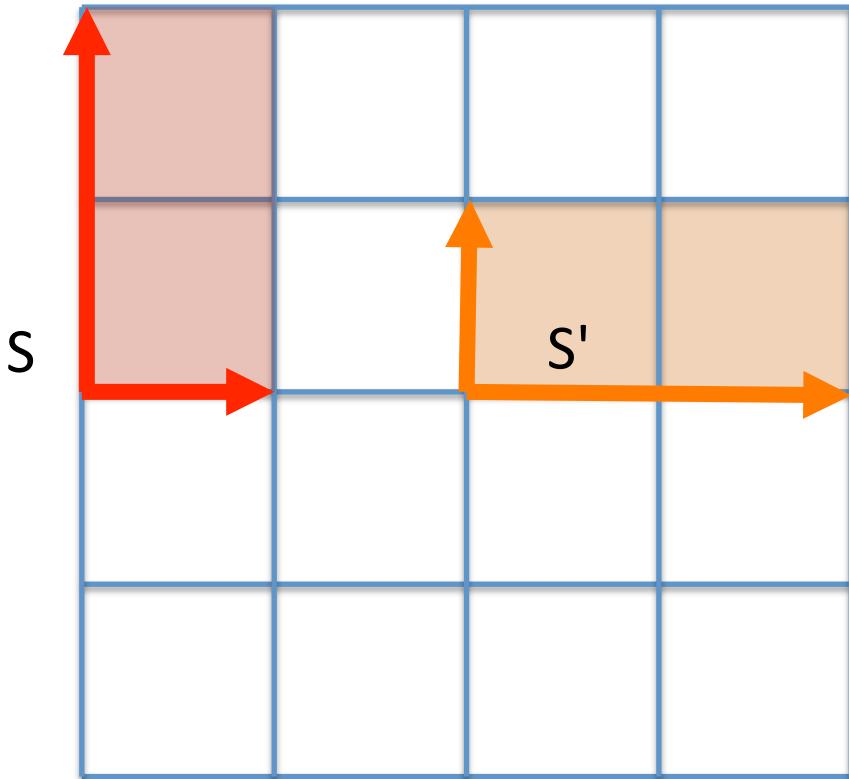


Equivalent supercells:

$$S' = ST$$

- 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.

```
:Zr0 bpuchala$ casm enum --method ScelEnum --max 4
-- Construct: CASM Project --
from: "/Users/bpuchala/casm/Zr0"

-- Load project data --
read: "/Users/bpuchala/casm/Zr0/.casm/composition_axes.json"
```

```
-- Begin: ScelEnum --
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.
```

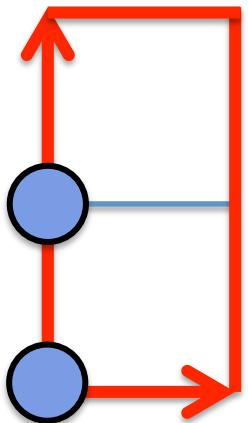
```
Write SCEL...
DONE
```

```
Writing config_list...
DONE
```

```
:Zr0 bpuchala$
```

SCEL4_1_4_1_0_0_0
Volume Shape Factors

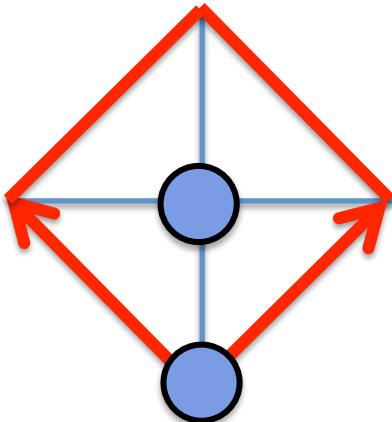
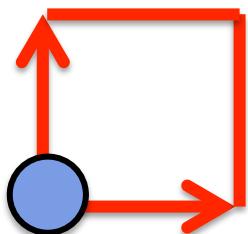
Configuration enumeration



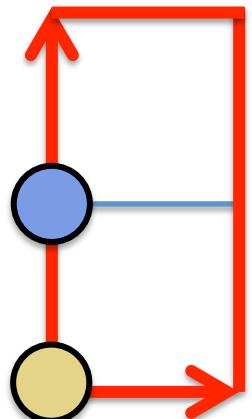
A:
B:

Equivalent configurations:

- Supercells of the volume 1 configuration



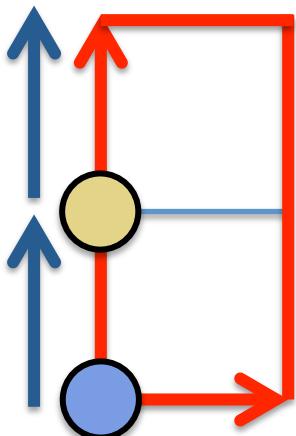
Configuration enumeration



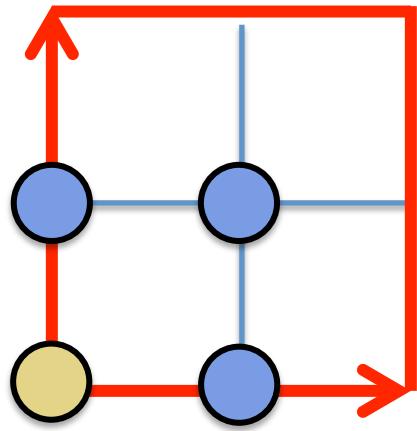
A:
B:

Equivalent configurations:

- Translation maps one configuration onto the other



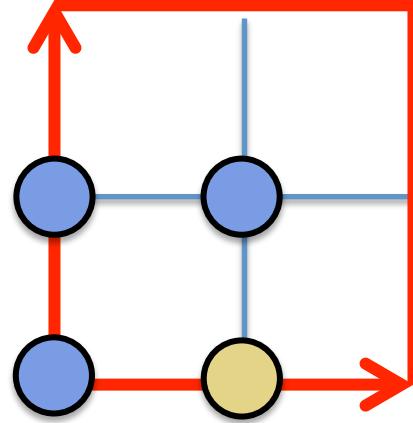
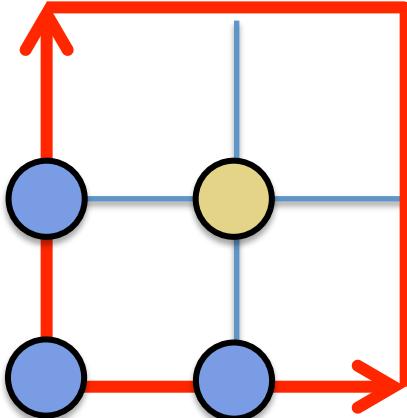
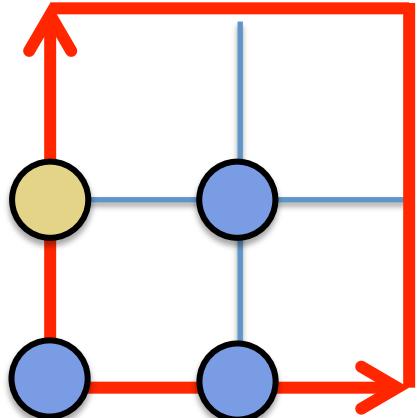
Configuration enumeration



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.

```
:Zr0 bpuchala$ casm enum --method ConfigEnumAllOccupations --all
-- Construct: CASM Project --
from: "/Users/bpuchala/casm/Zr0"

-- Load project data --
read: "/Users/bpuchala/casm/Zr0/.casm/composition_axes.json"
read: "/Users/bpuchala/casm/Zr0/training_data/SCEL"
read: "/Users/bpuchala/casm/Zr0/.casm/config_list.json"

# configurations in this project: 0

-- Begin: ConfigEnumAllOccupations --
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.

# new configurations: 336
# configurations in this project: 336

Write SCEL...
DONE

Writing config_list...
DONE

:Zr0 bpuchala$
```

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.

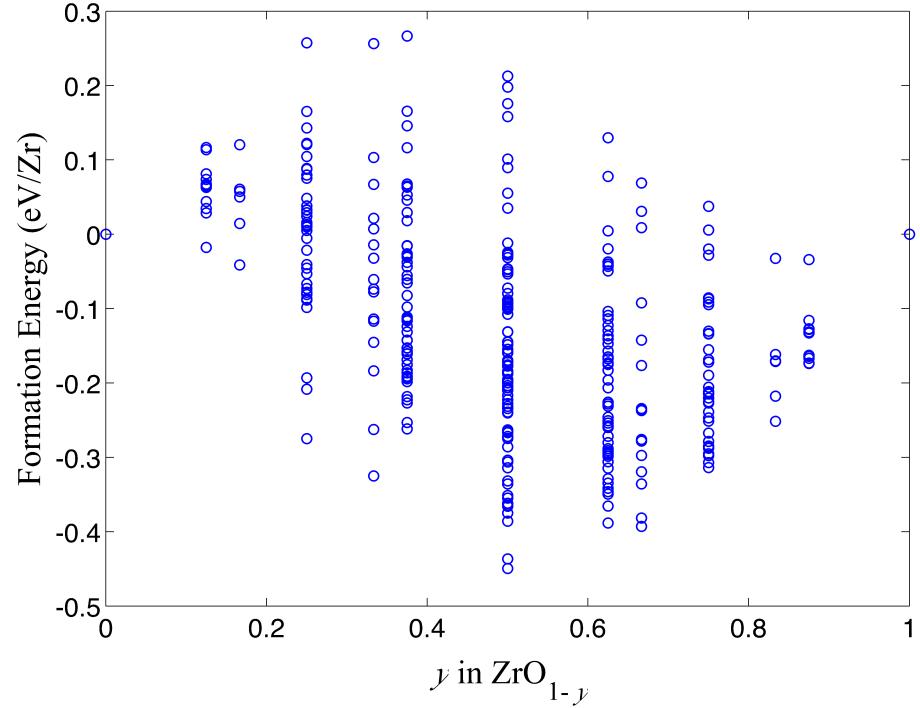
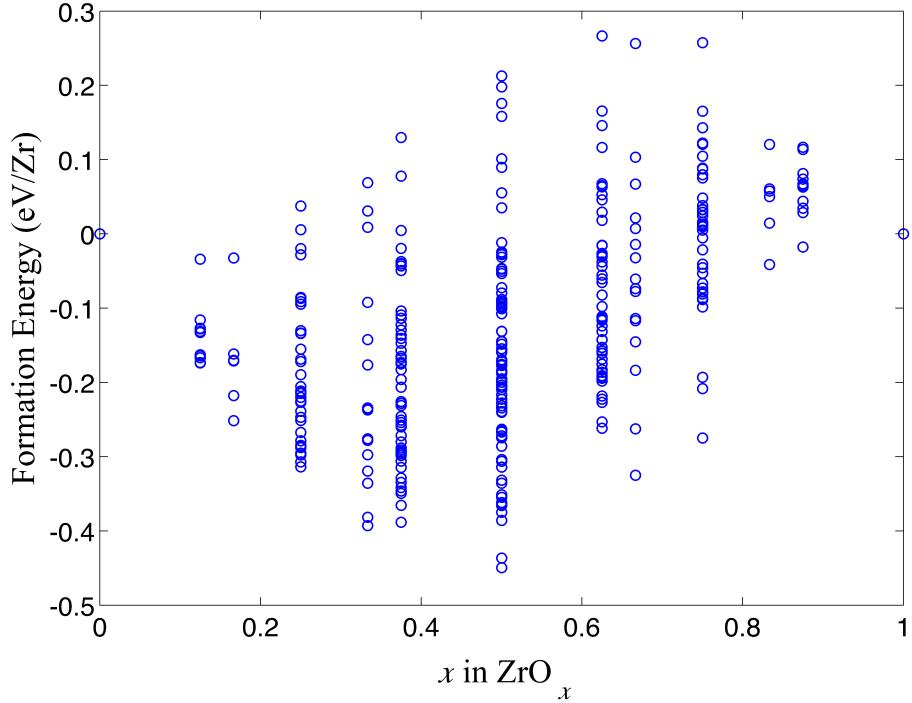
`comp`: `comp(a)`, `comp(b)`, etc.

- Primitive-cell-based formula parameters (e.g., $\text{Au}_a\text{Cu}_b\text{Pt}_{(1-a-b)}$), determined by user-specified composition axes. Accounts for vacancies.

When vacancies are present, ‘`atom_frac`’ is qualitatively different from ‘`comp_n`’ and ‘`comp`’

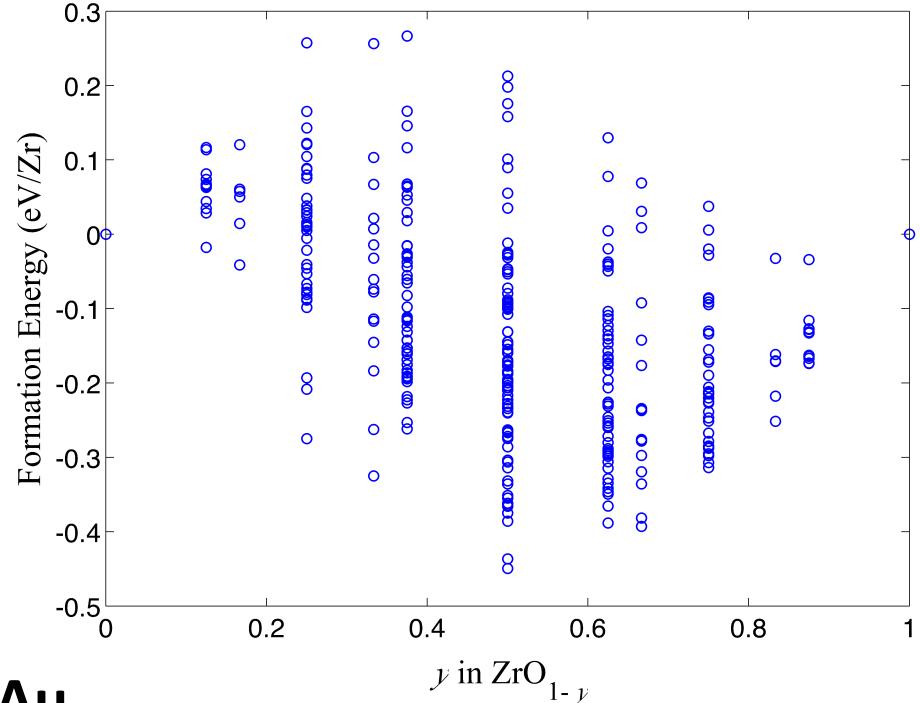
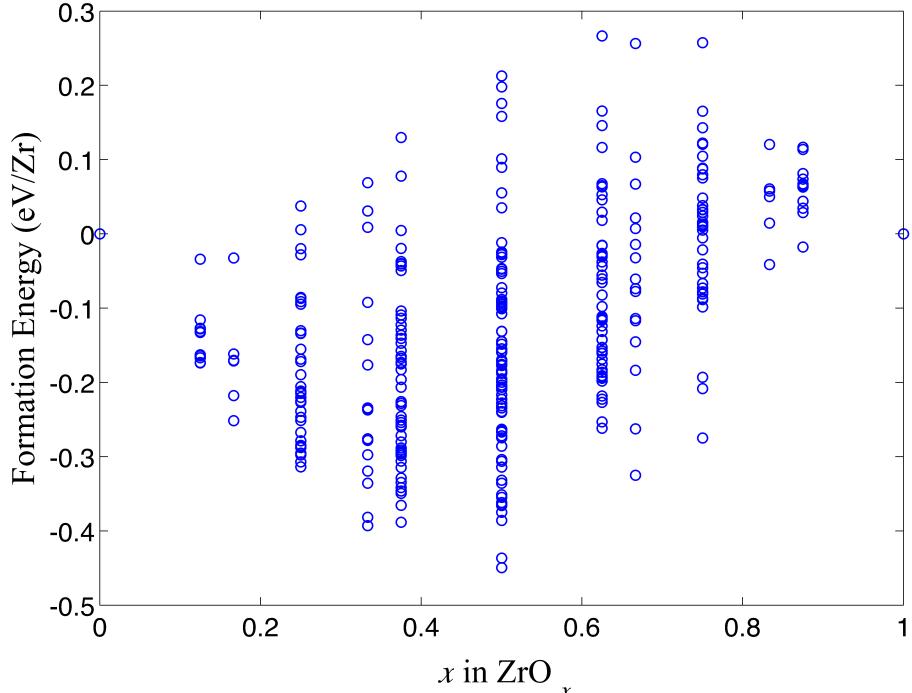
Specifying composition axes

Binary Compound:

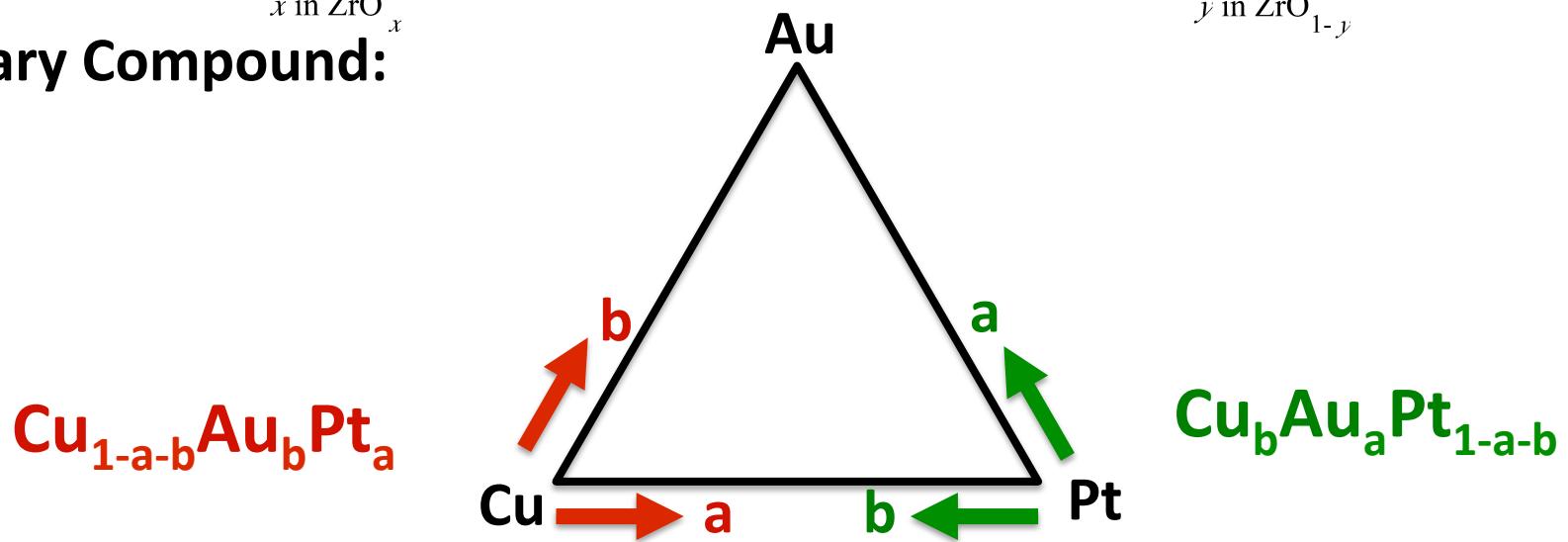


Specifying composition axes

Binary Compound:



Ternary Compound:

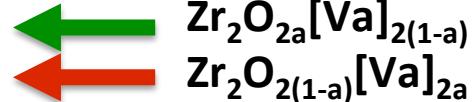


Specifying composition axes

```
[Adamant:~/Documents/Zr0$ casm composition -d  
-- Construct: CASM Project --  
from: "/Users/johnct/Documents/Zr0"  
  
-- Load project data --  
read: "/Users/johnct/Documents/Zr0/.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/Zr0$ █
```

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:

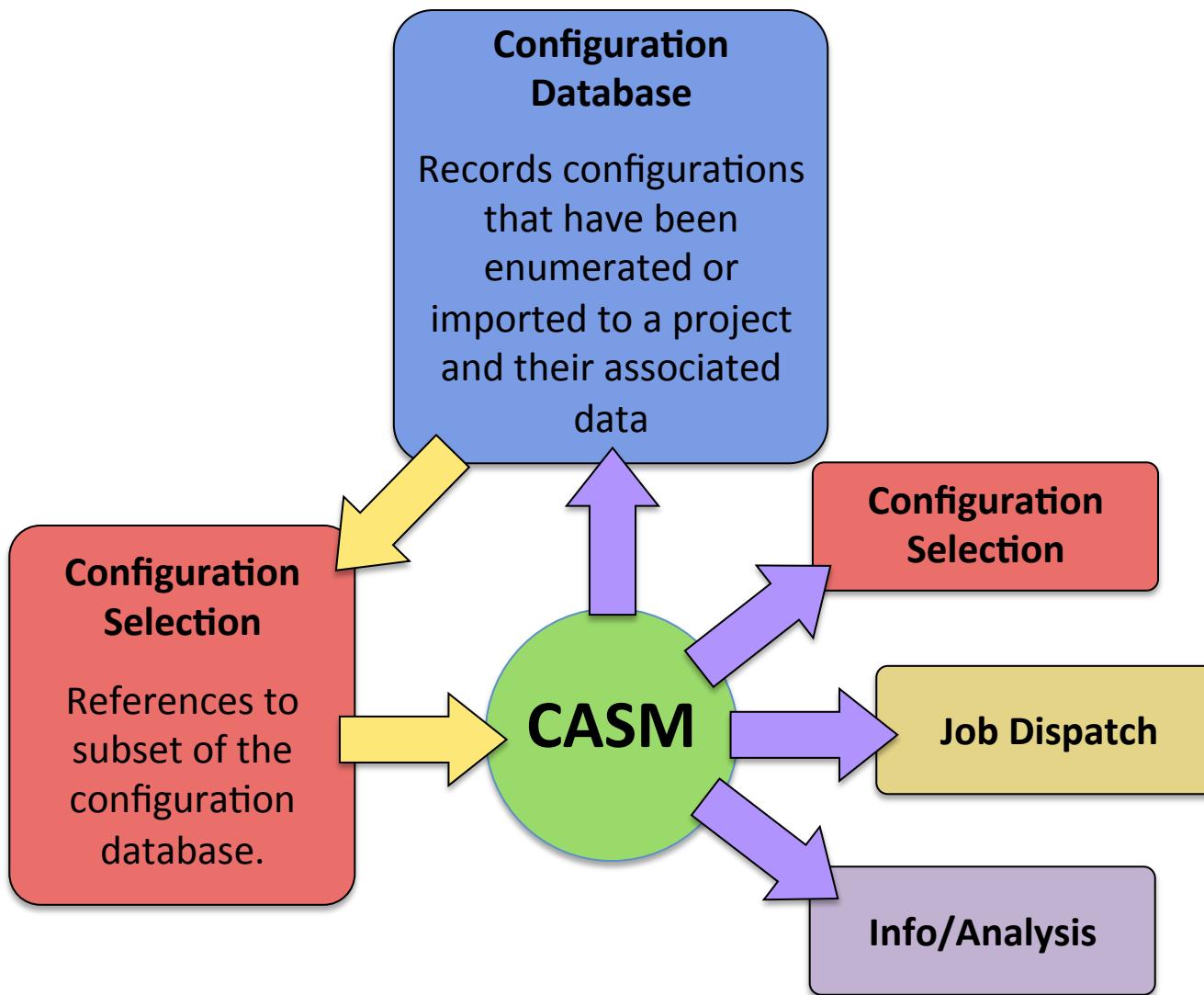
'calctype'	DFT parameters and setting (XC functional, pseudopotential, convergence parameters, etc.)
'property'	The particular quantity being measured (formation_energy, volume_relaxation, etc.)
'ref'	Baseline reference used for measuring quantity (zero-point of energy, reference volume, etc.)
'bset'	Basis functions used to “fingerprint” configurations and in terms of which the property model is expressed .
'eci'	The model parameters (effective cluster interactions) 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 that 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>

#	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>

The image shows two terminal windows side-by-side. The left window displays a list of configuration entries with columns: configname, selected, arbitrary_data1, arbitrary_data2, and arbitrary_data3. The right window shows the same data but with the 'selected' column highlighted in red, indicating the subset being manipulated.

#	configname	selected	arbitrary_data1	arbitrary_data2	arbitrary_data3
1	SCEL1_1_1_1_0_0_0/0	1	1.00000000	0.00000000	0.00000000
2	SCEL1_1_1_1_0_0_0/1	1	0.66666667	0.00000000	0.33333333
3	SCEL1_1_1_1_0_0_0/2	1	0.50000000	0.00000000	0.50000000
4	SCEL2_1_1_2_0_0_0/0	1	0.80000000	0.00000000	0.20000000
5	SCEL2_1_1_2_0_0_0/1	1	0.66666667	0.00000000	0.33333333
6	SCEL2_1_1_2_0_0_0/2	1	0.57142857	0.00000000	0.42857143
7	SCEL2_1_2_1_0_0_0/0	1	0.80000000	0.00000000	0.20000000
8	SCEL2_1_2_1_0_0_0/1	1	0.66666667	0.00000000	0.33333333
9	SCEL2_1_2_1_0_0_0/2	1	0.57142857	0.00000000	0.42857143
10	SCEL2_1_2_1_0_0_0/3	1	0.66666667	0.00000000	0.33333333
11	SCEL2_1_2_1_1_0_0/0	1	0.80000000	0.00000000	0.20000000
12	SCEL2_1_2_1_1_0_0/1	1	0.66666667	0.00000000	0.33333333
13	SCEL2_1_2_1_1_0_0/2	1	0.57142857	0.00000000	0.42857143
14	SCEL3_1_1_3_0_0_0/0	1	0.85714286	0.00000000	0.14285714
15	SCEL3_1_1_3_0_0_0/1	1	0.75000000	0.00000000	0.25000000
16	SCEL3_1_1_3_0_0_0/2	1	0.75000000	0.00000000	0.25000000
17	SCEL3_1_1_3_0_0_0/3	1	0.66666667	0.00000000	0.33333333
18	SCEL3_1_1_3_0_0_0/4	1	0.60000000	0.00000000	0.40000000
19	SCEL3_1_1_3_0_0_0/5	1	0.75000000	0.00000000	0.25000000
20	SCEL3_1_1_3_0_0_0/6	1	0.66666667	0.00000000	0.33333333
21	SCEL3_1_1_3_0_0_0/7	1	0.60000000	0.00000000	0.40000000
22	SCEL3_1_1_3_0_0_0/8	1	0.54545455	0.00000000	0.45454545
23	SCEL3_1_1_3_0_0_0/9	1	0.60000000	0.00000000	0.40000000
24	SCEL3_1_3_1_2_0_0/0	1	0.85714286	0.00000000	0.14285714
25	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-calc --submit --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>)

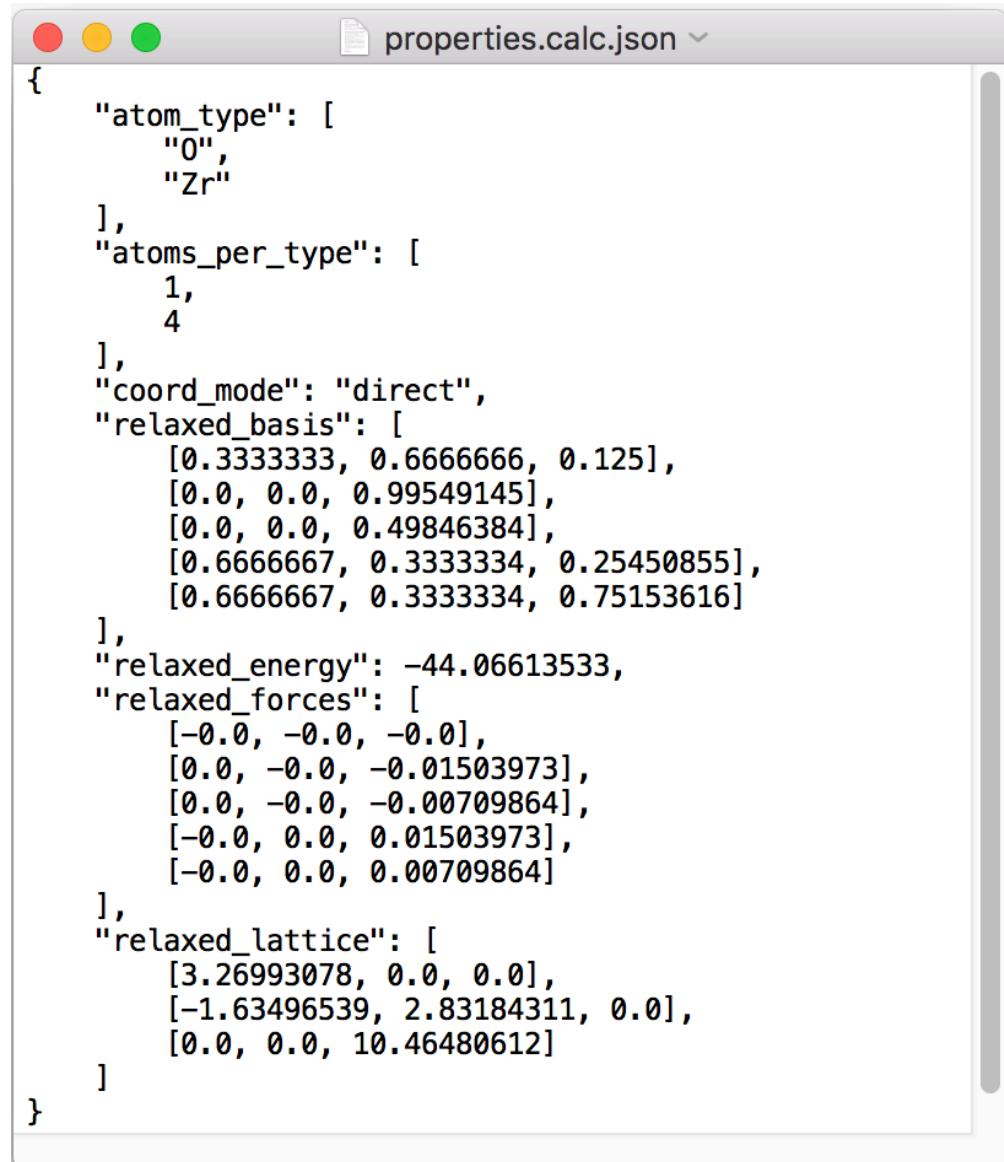
Current support for VASP and SeqQeust; QuantumEspresso partially completed, and support for other codes is planned. Contributions from the user community are encouraged!

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'



```
{  
    "atom_type": [  
        "0",  
        "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):

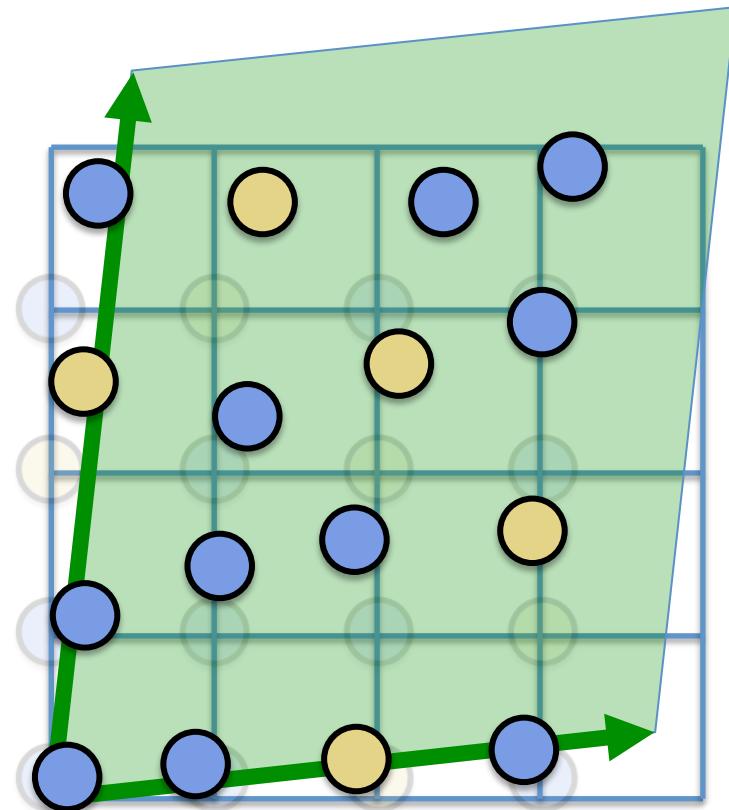
Mean square displacement

total deformation cost:

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

casm import:

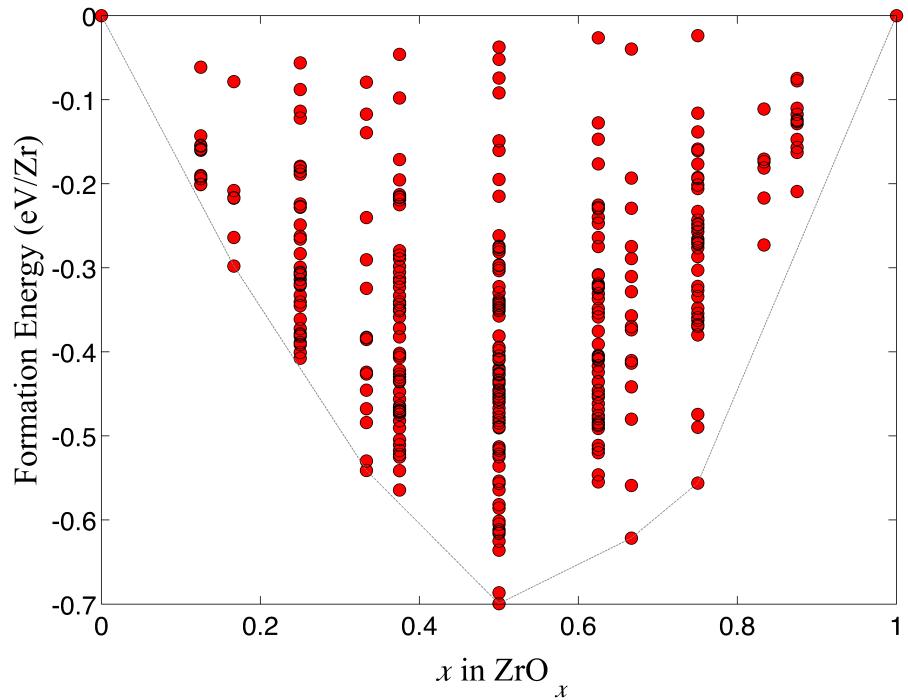
Use same algorithms as casm update to add ideal or relaxed configurations to project (even if not enumerated)



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

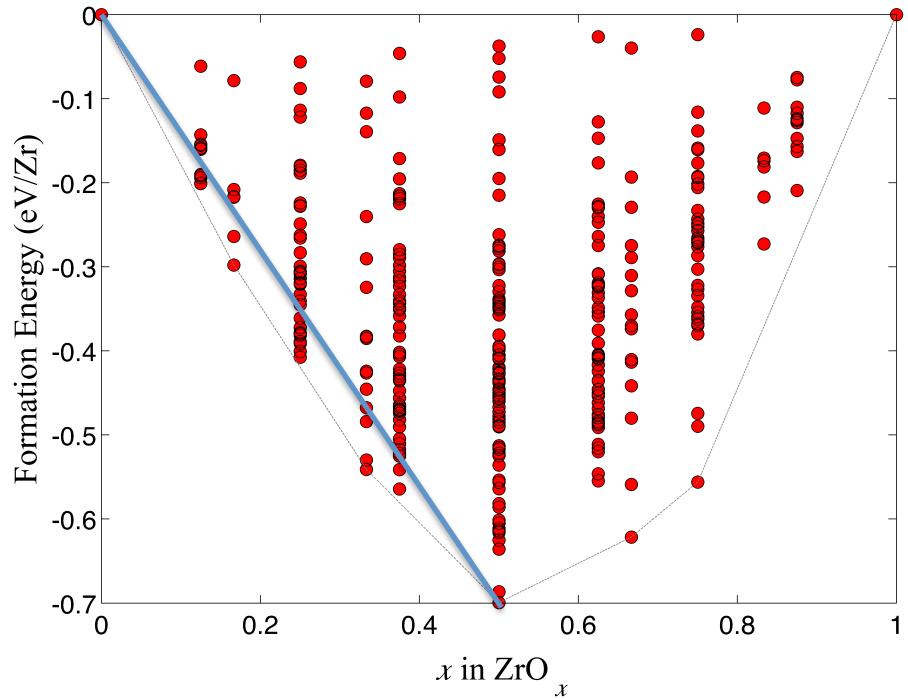
Setting energy reference

Chemical energy can only be measured relative to reference compounds



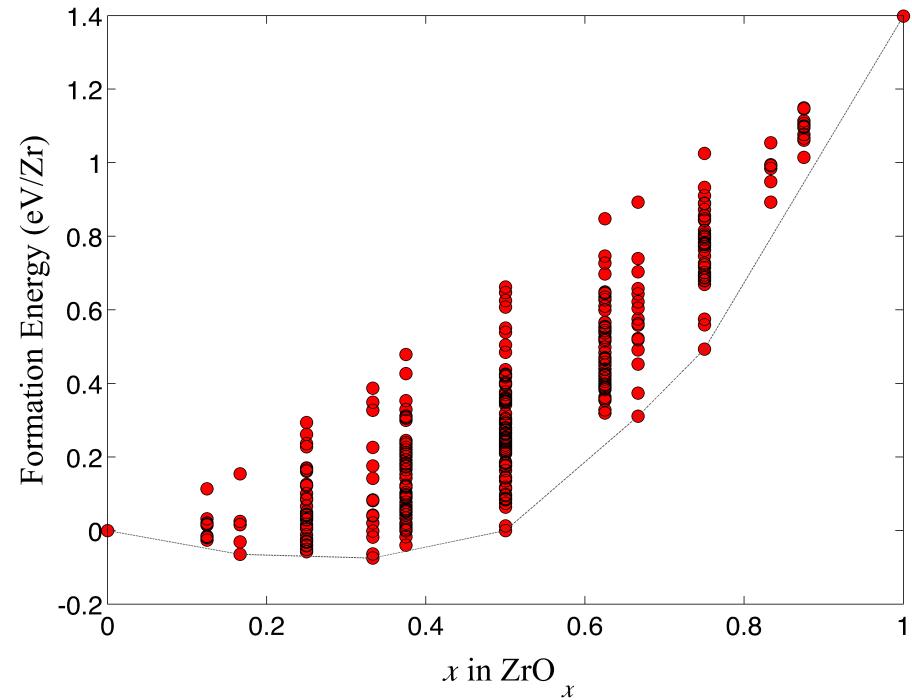
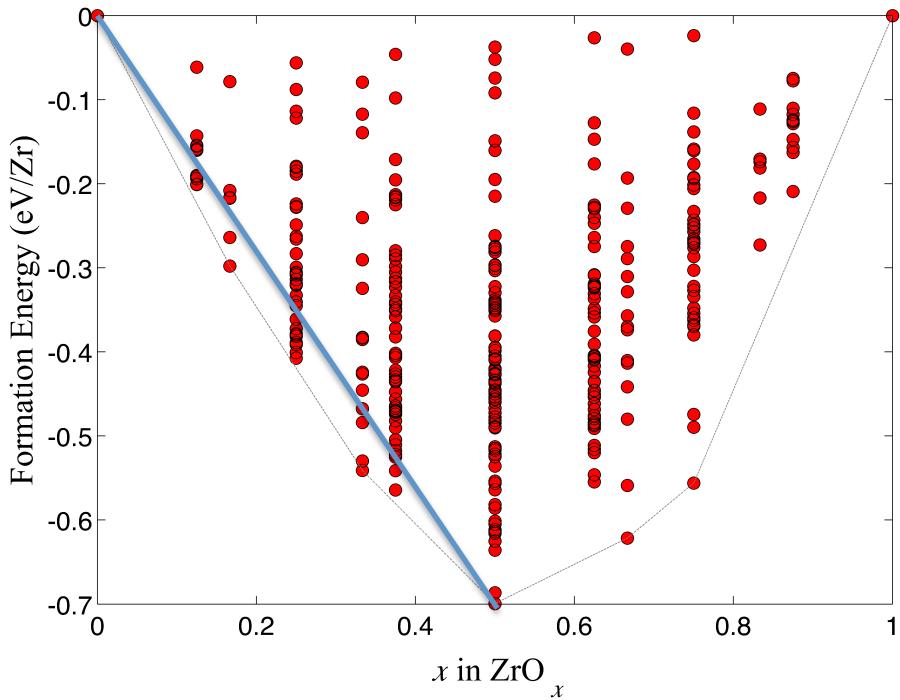
Setting energy reference

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

First-principles energies of a “few” excitations

$$H = \sum_{i=1}^{N_e} \left(-\nabla_i^2 + V_{nuc}(r_i) \right) + \frac{1}{2} \sum_{i \neq j} \frac{1}{|r_i - r_j|} + E_{nuc}(\{R\})$$



Lattice Model Hamiltonian

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



Monte Carlo

$$Z = \sum_{\vec{\sigma}} \exp\left(-\frac{E(\vec{\sigma})}{k_B T}\right)$$



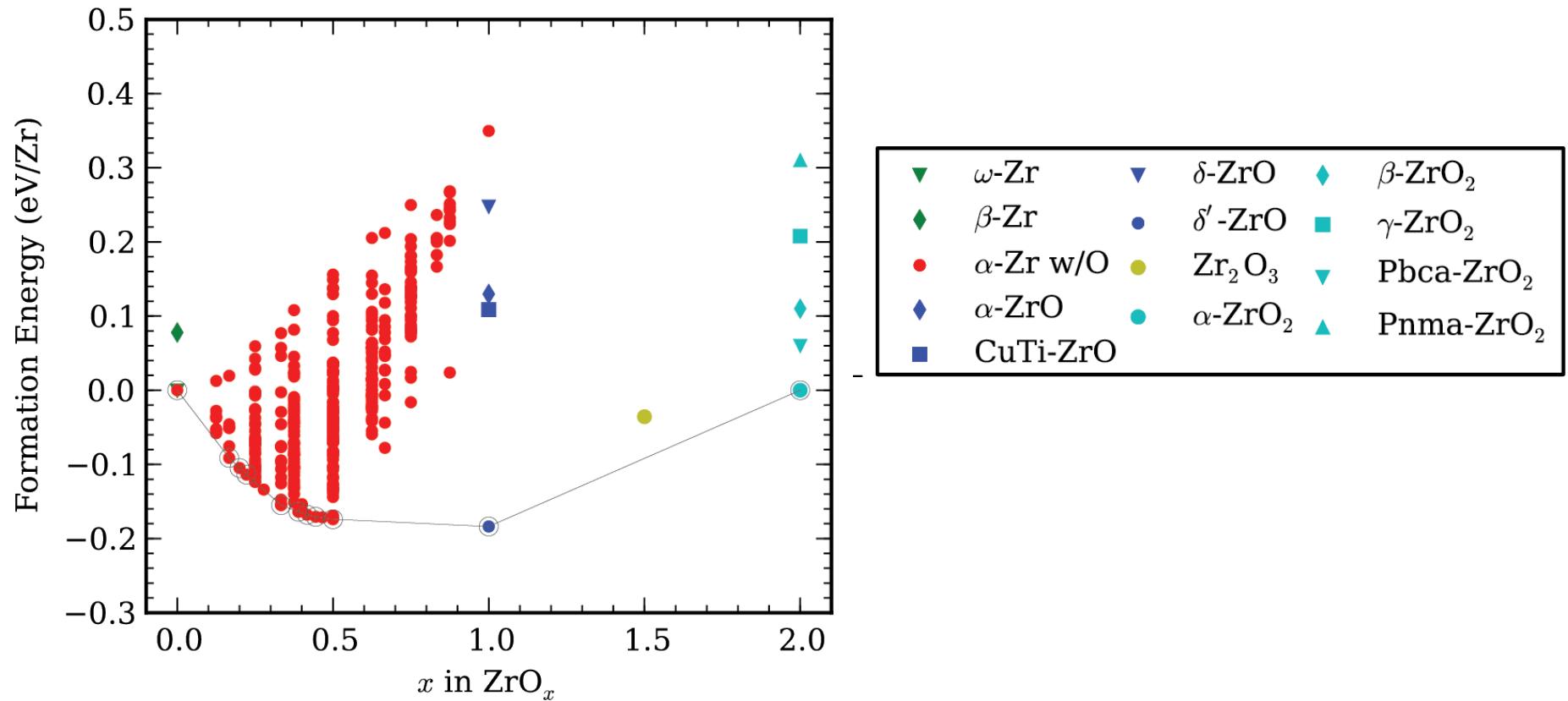
$$F = -k_B T \ln(Z)$$

Thermodynamics

Extrapolate to
all other excitations

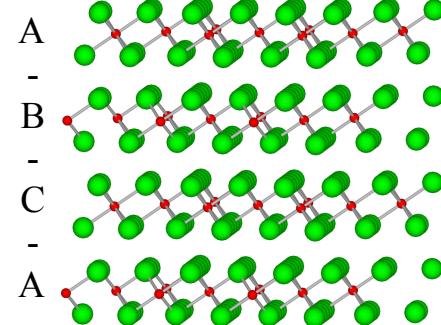
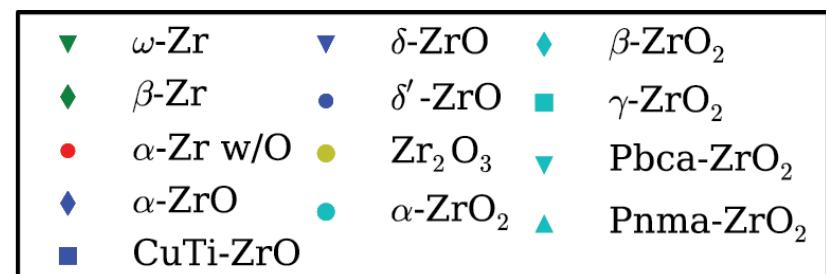
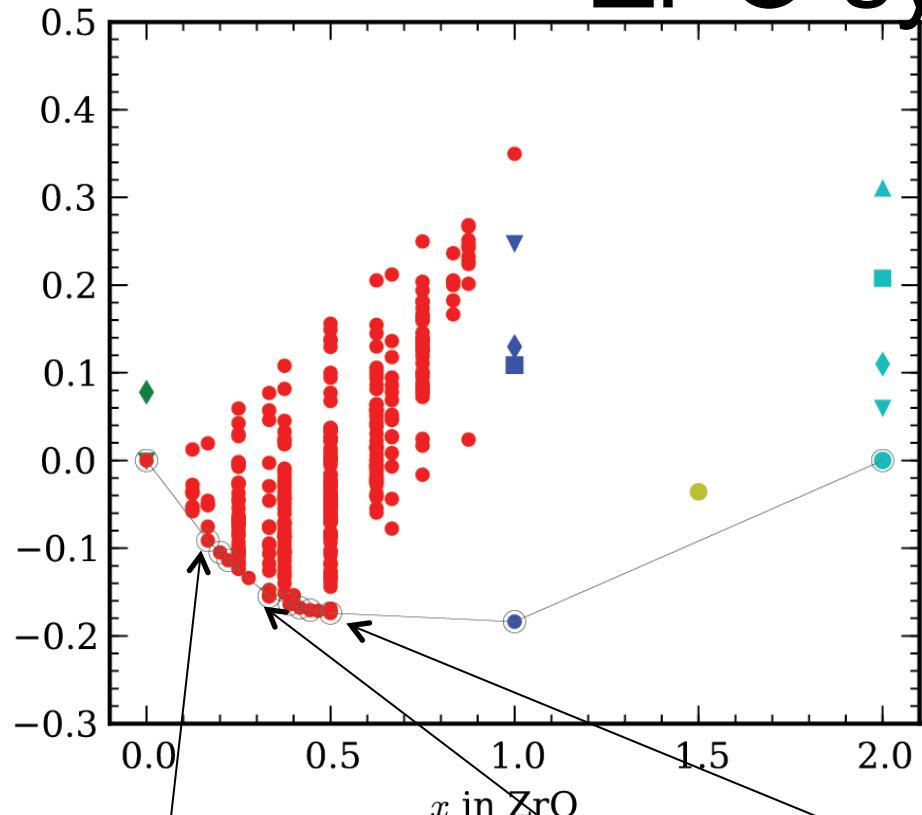
Zr-O system

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

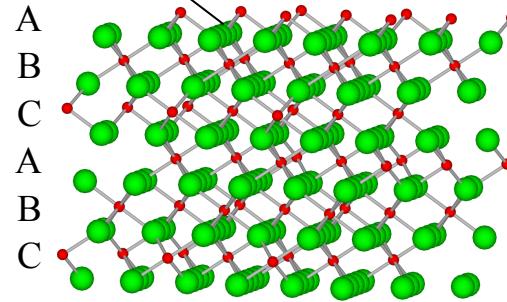


Zr-O system

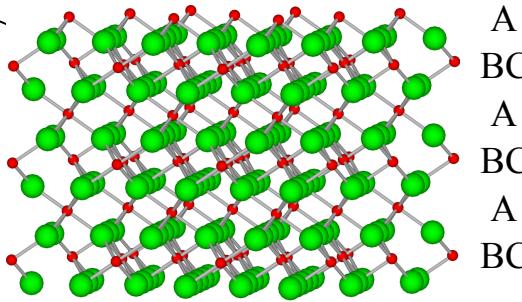
Formation Energy (eV/Zr)



$\text{ZrO}_{1/6}$



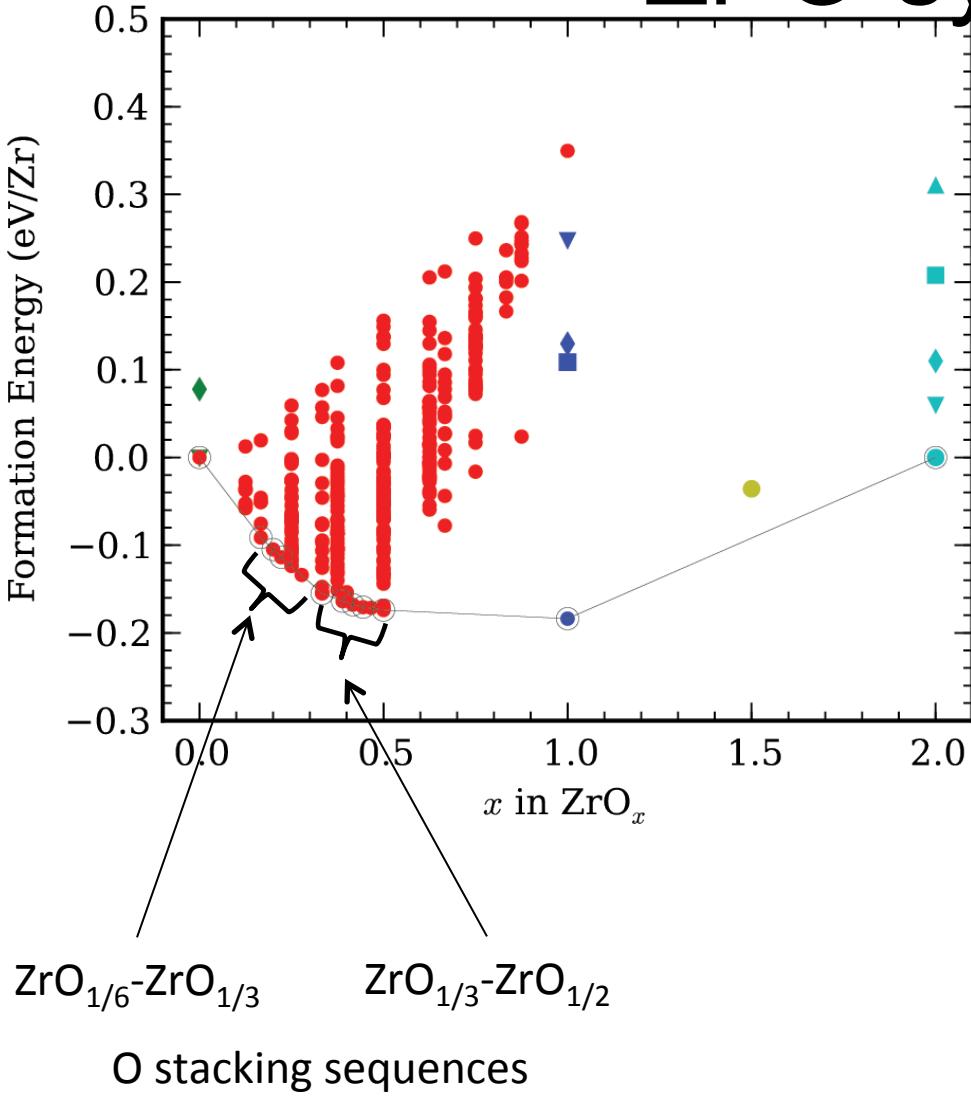
$\text{ZrO}_{1/3}$



$\text{ZrO}_{1/2}$

Zr-O system

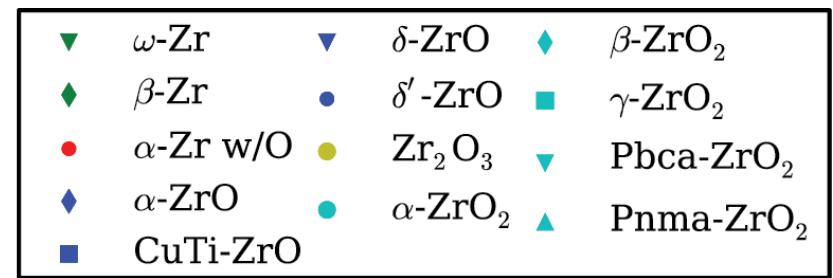
Formation Energy (eV/Zr)



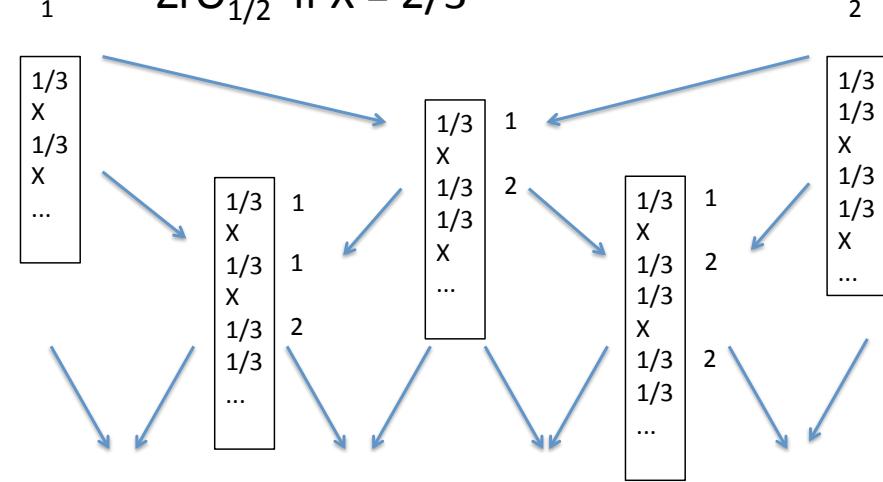
x in ZrO_x

$\text{ZrO}_{1/6}\text{-}\text{ZrO}_{1/3}$ $\text{ZrO}_{1/3}\text{-}\text{ZrO}_{1/2}$

O stacking sequences



$\text{ZrO}_{1/6}$ if $X = 0$
 $\text{ZrO}_{1/2}$ if $X = 2/3$



... can repeat for more "generations"

CASM: Clusters Approach to Statistical Mechanics

First-principles energies of a “few” excitations

$$H = \sum_{i=1}^{N_e} \left(-\nabla_i^2 + V_{nuc}(r_i) \right) + \frac{1}{2} \sum_{i \neq j} \frac{1}{|r_i - r_j|} + E_{nuc}(\{R\})$$



Lattice Model Hamiltonian

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



Monte Carlo

$$Z = \sum_{\vec{\sigma}} \exp\left(-\frac{E(\vec{\sigma})}{k_B T}\right)$$



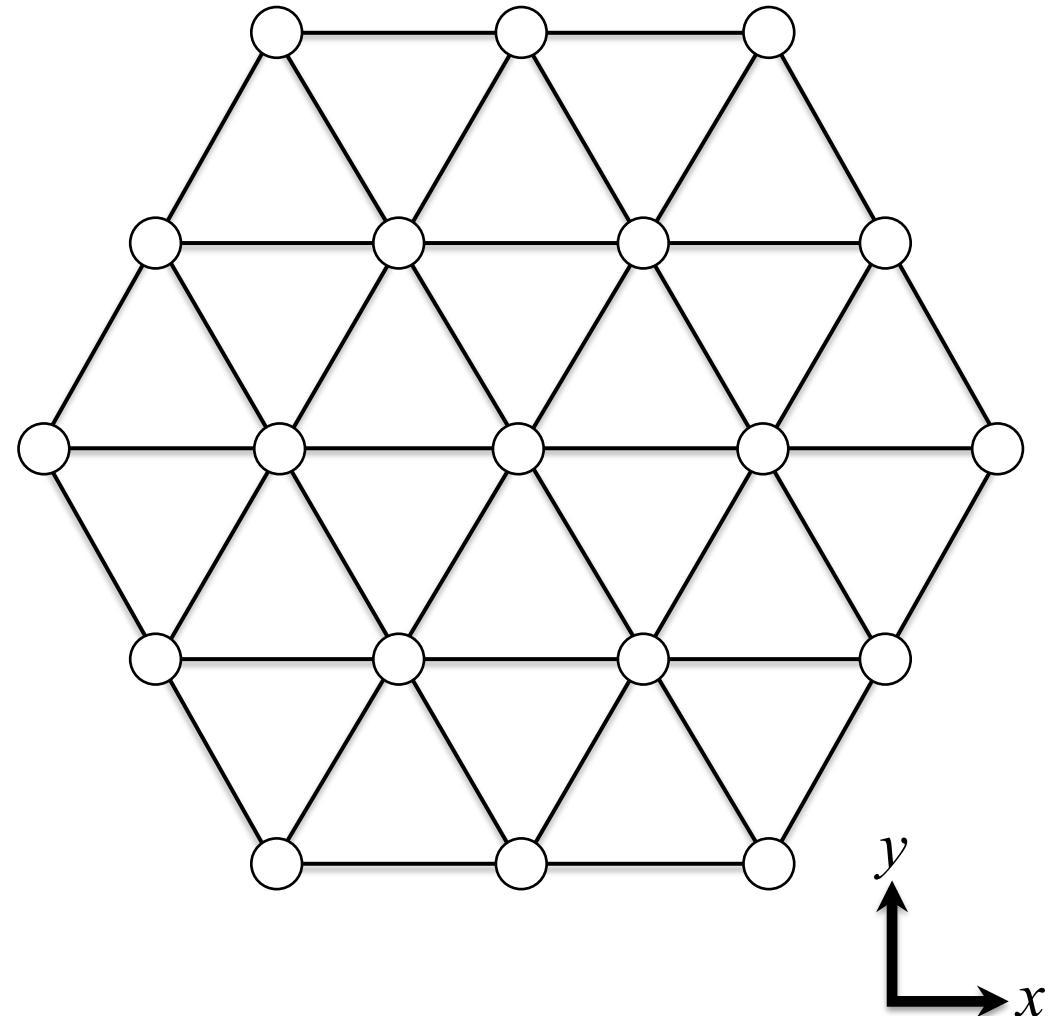
$$F = -k_B T \ln(Z)$$

Thermodynamics

Extrapolate to
all other excitations

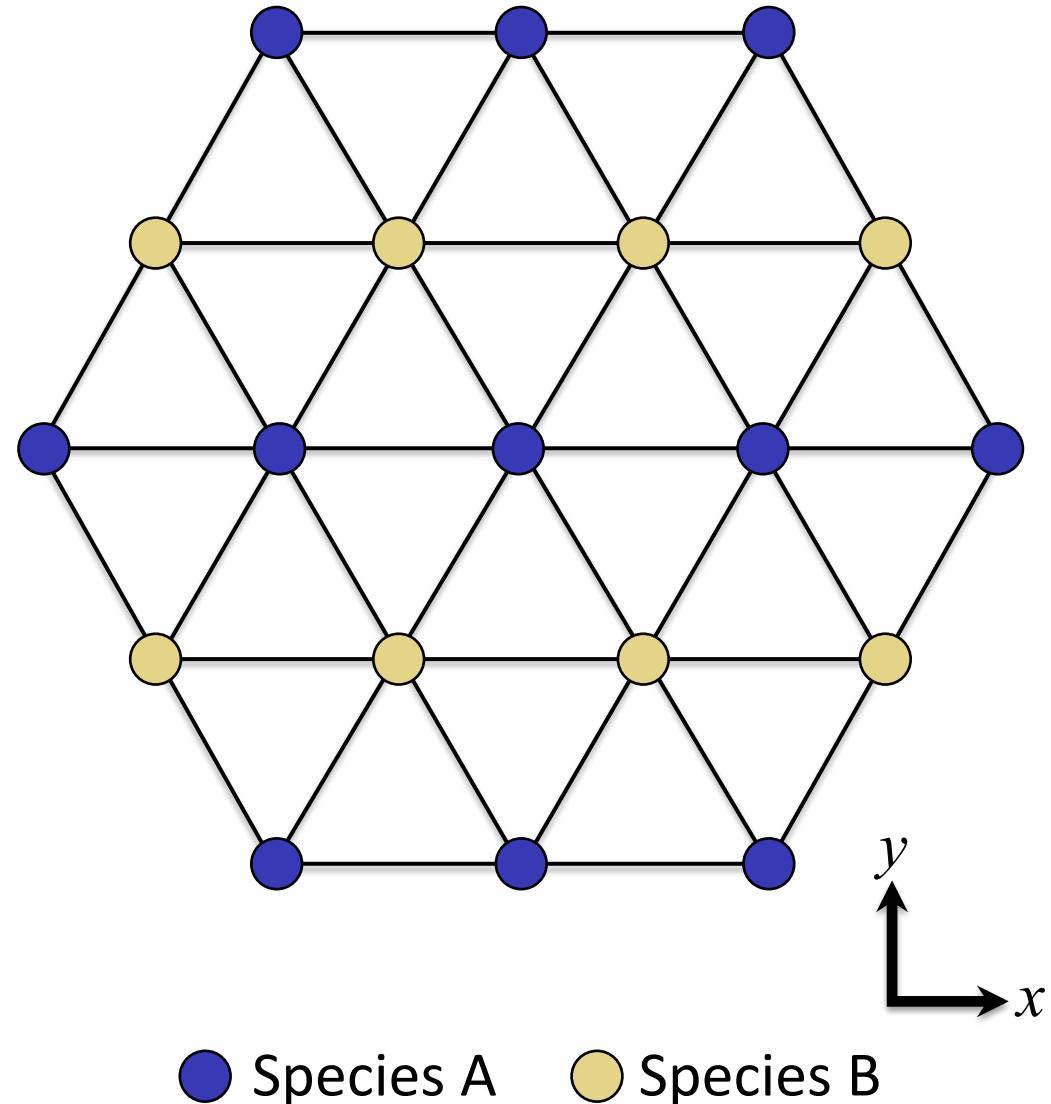
The Clusters Approach to Statistical Mechanics

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



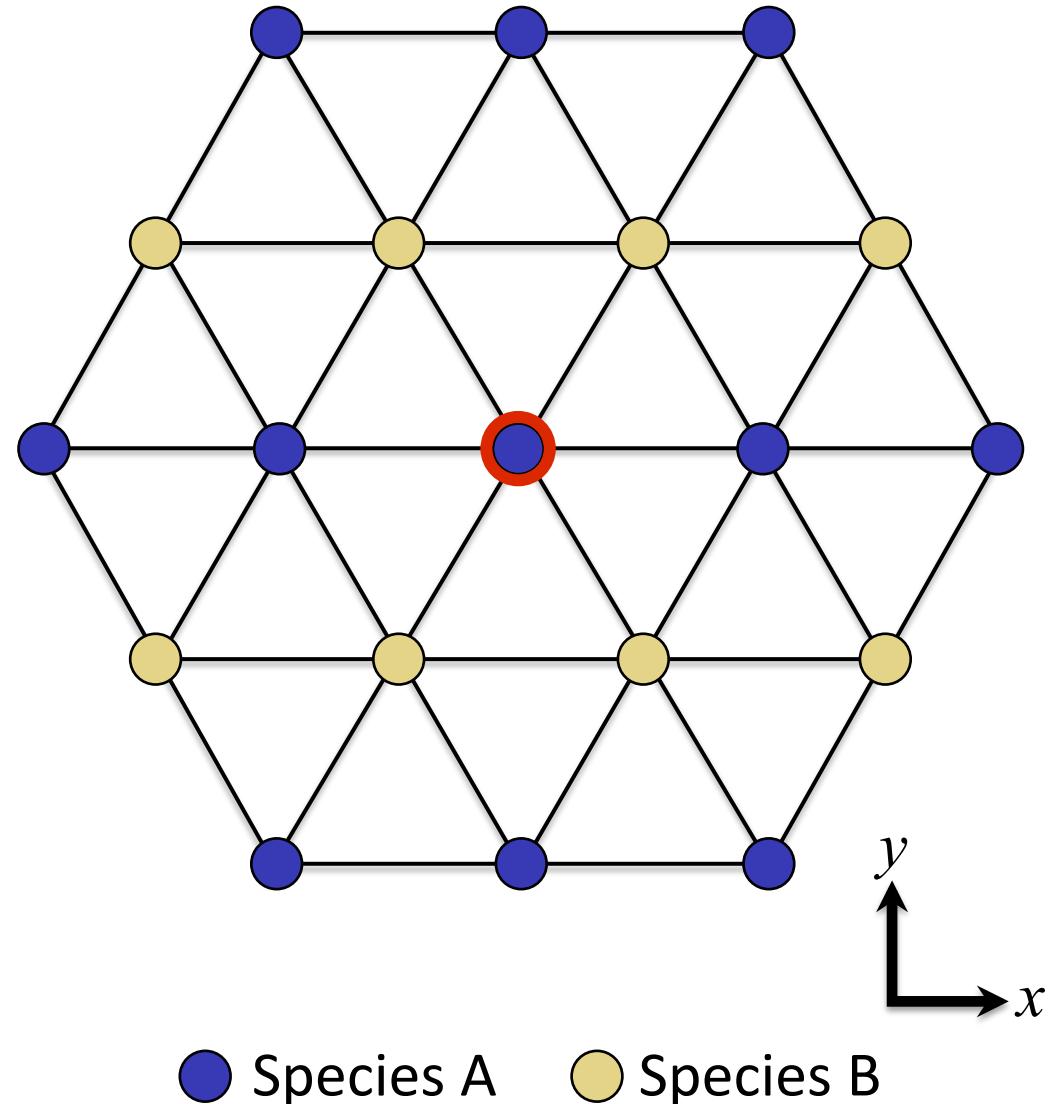
The Clusters Approach to Statistical Mechanics

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



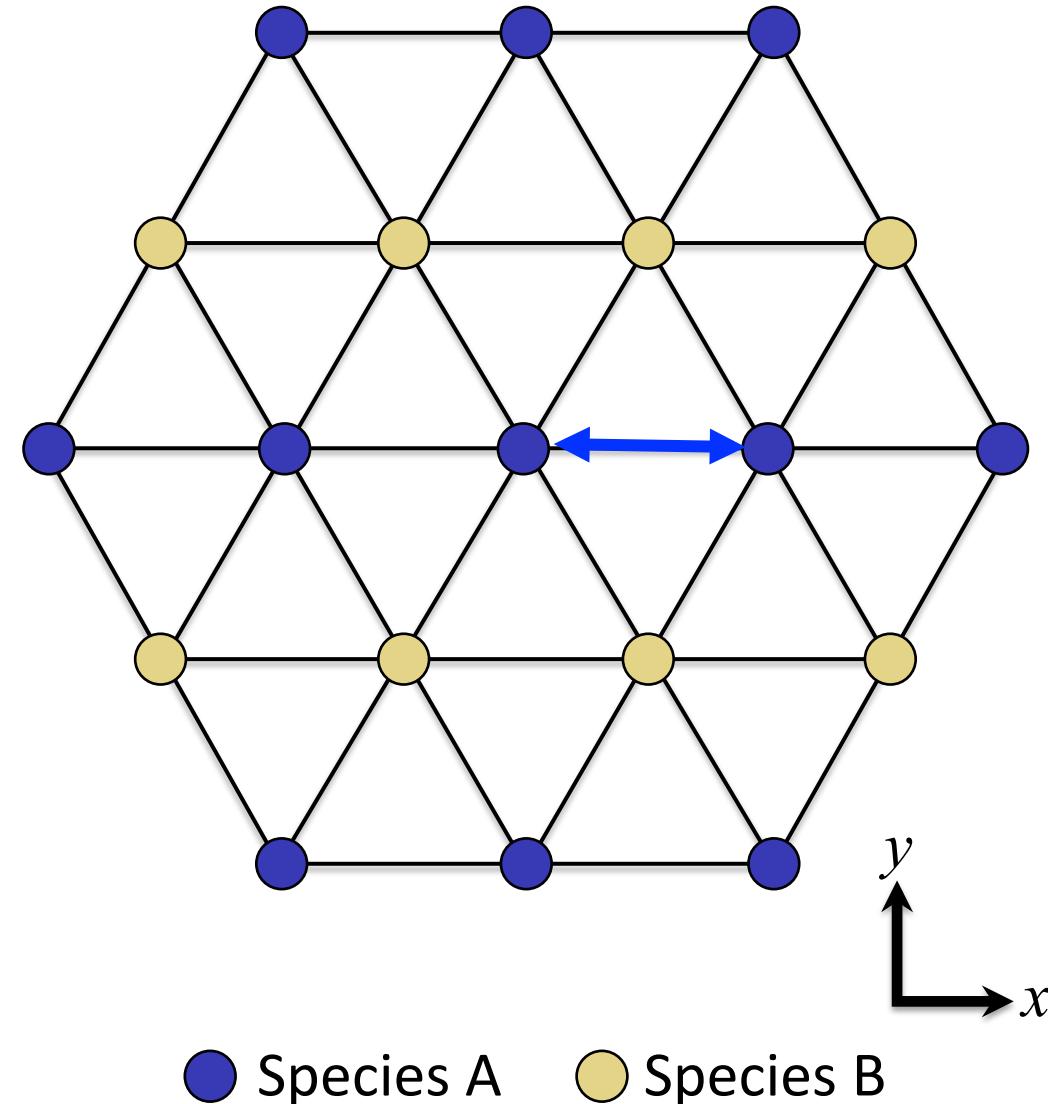
The Clusters Approach to Statistical Mechanics

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



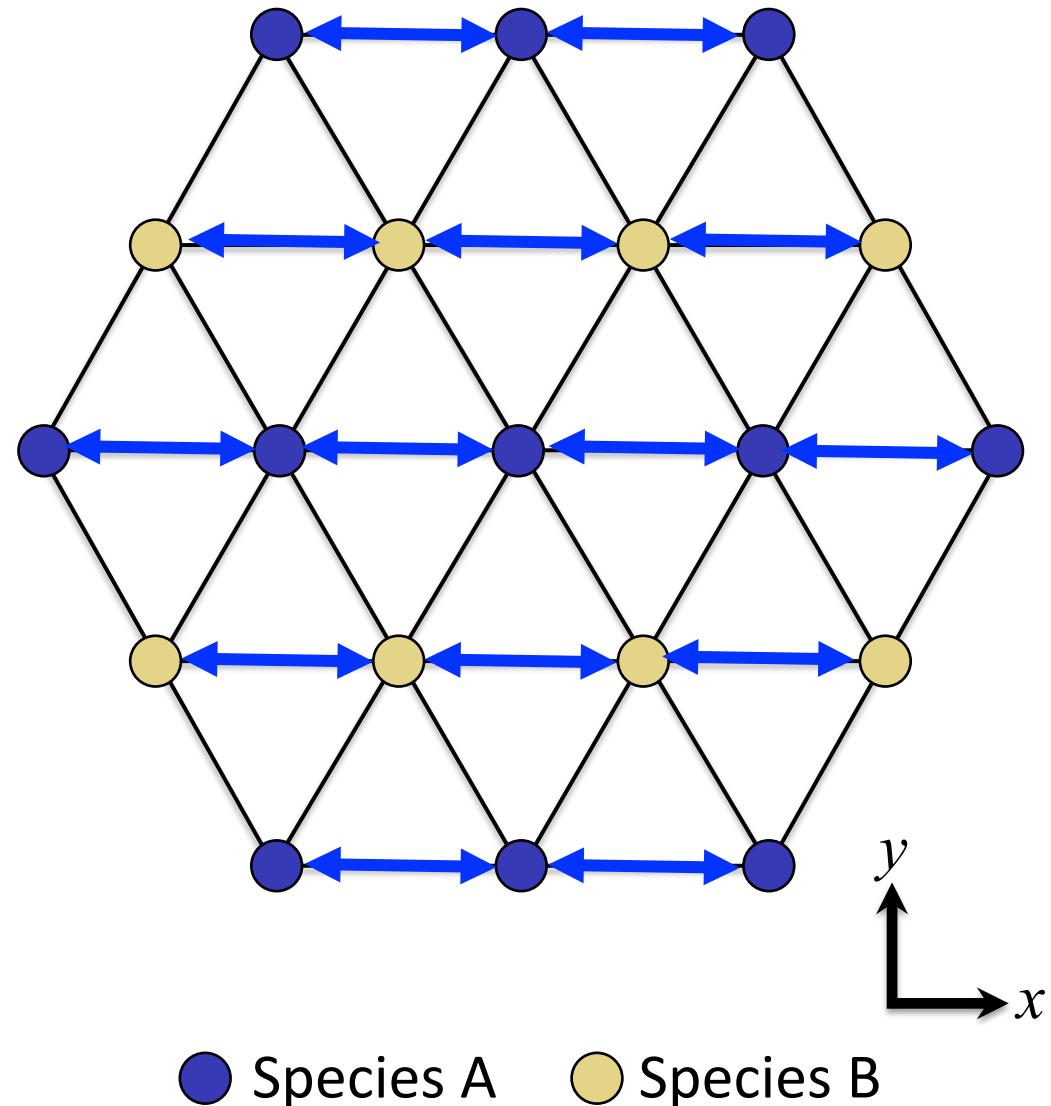
The Clusters Approach to Statistical Mechanics

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



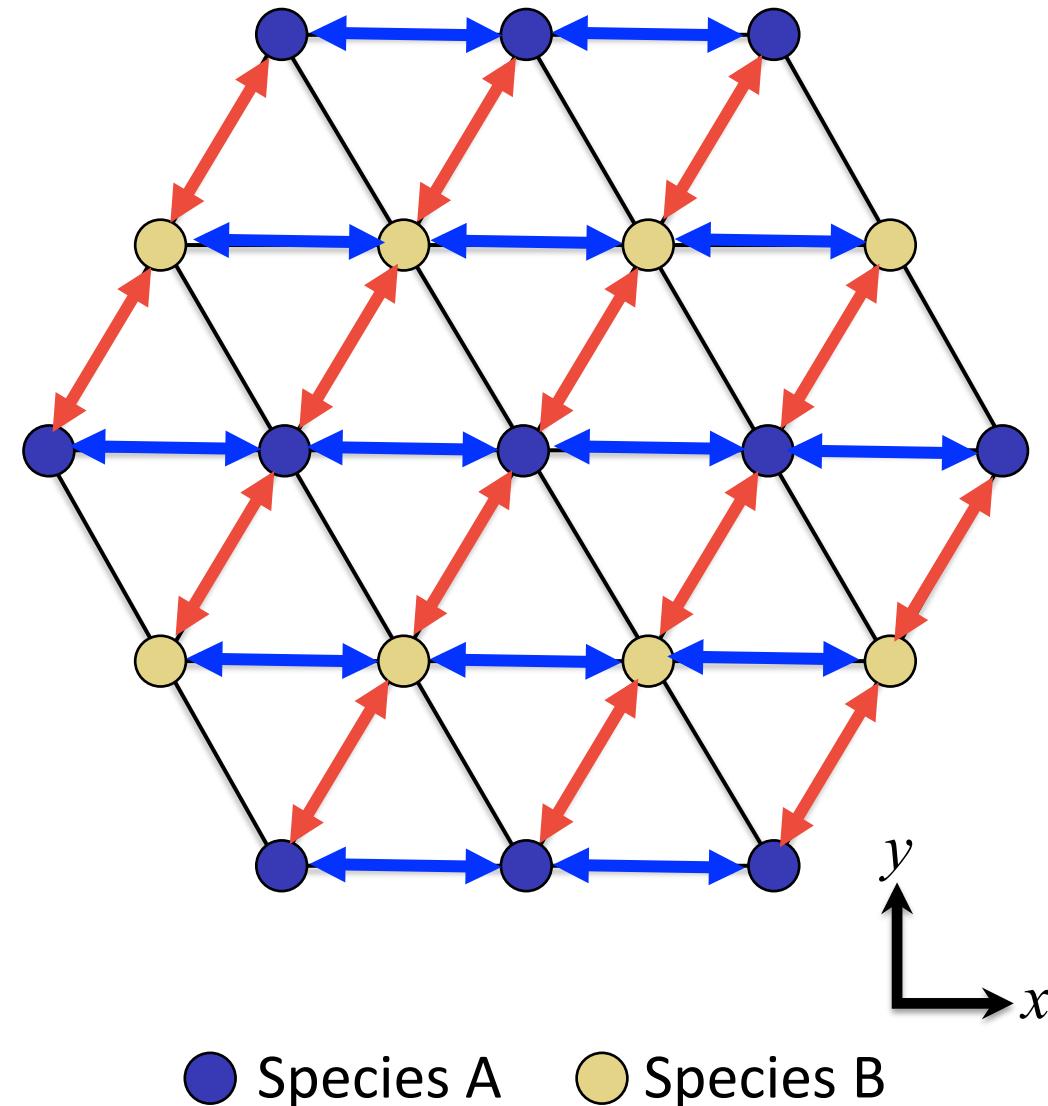
The Clusters Approach to Statistical Mechanics

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



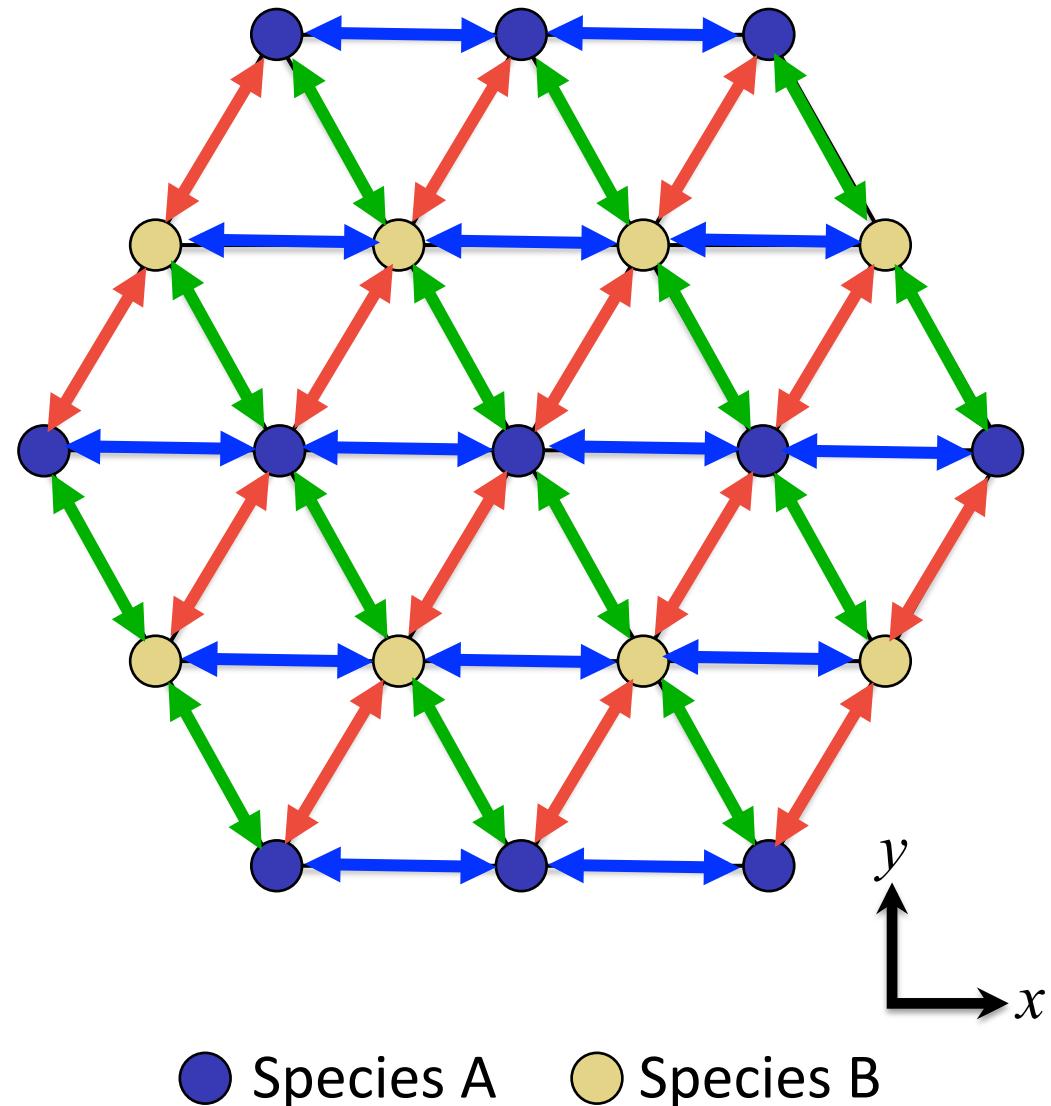
The Clusters Approach to Statistical Mechanics

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?

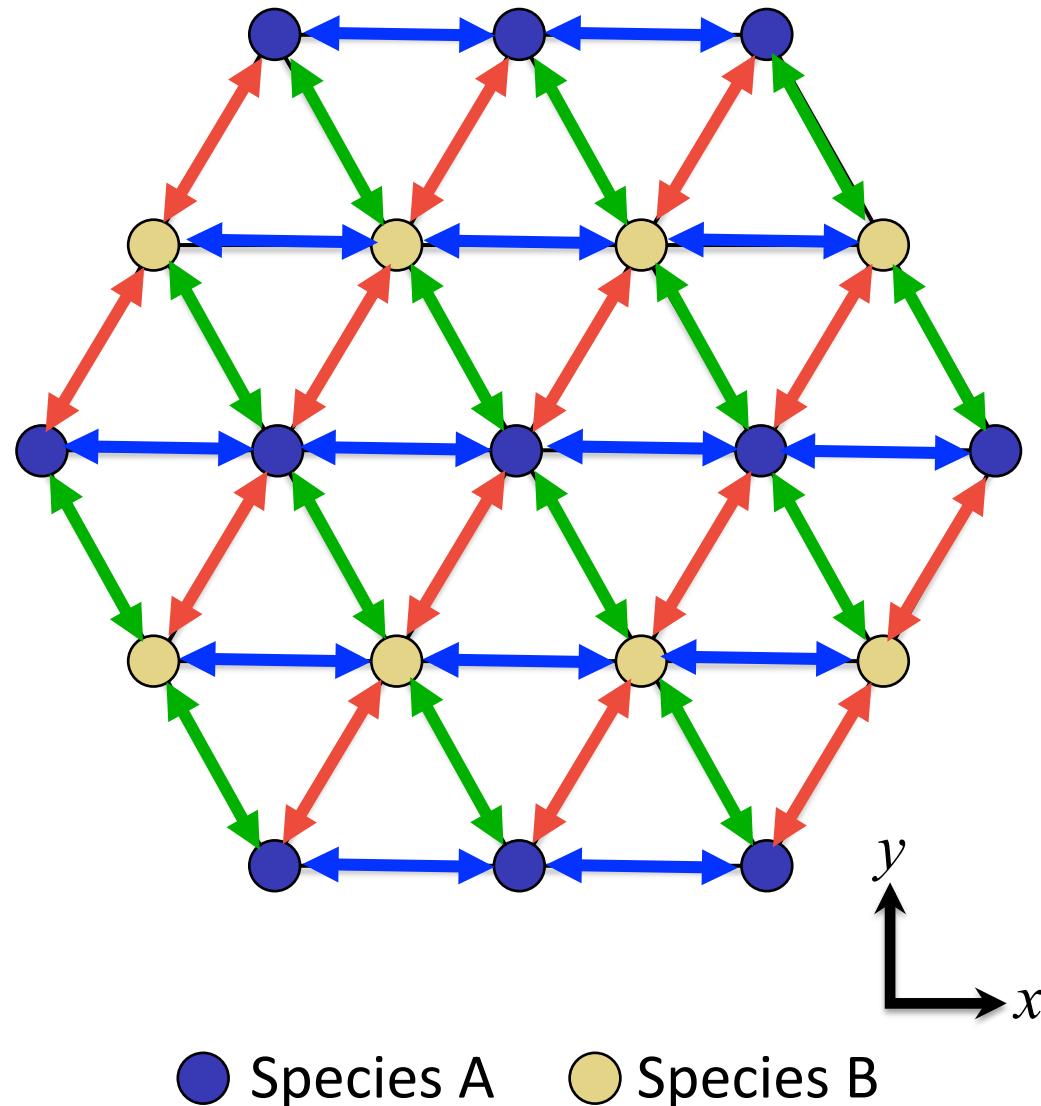


The Clusters Approach to Statistical Mechanics

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

Symmetric averages of measurements of cluster occupation states.

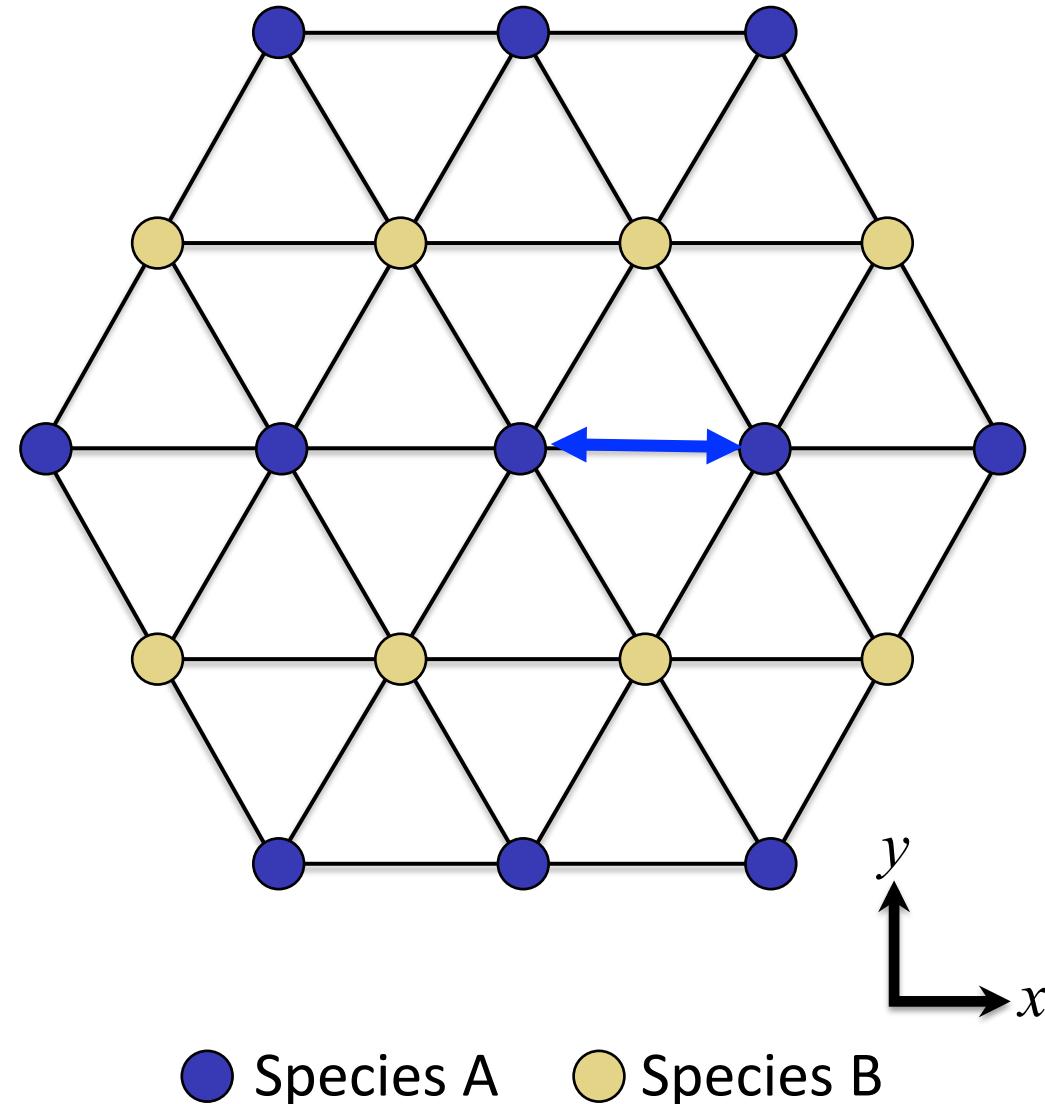
The **orbit** of the NN pair



The Clusters Approach to Statistical Mechanics

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

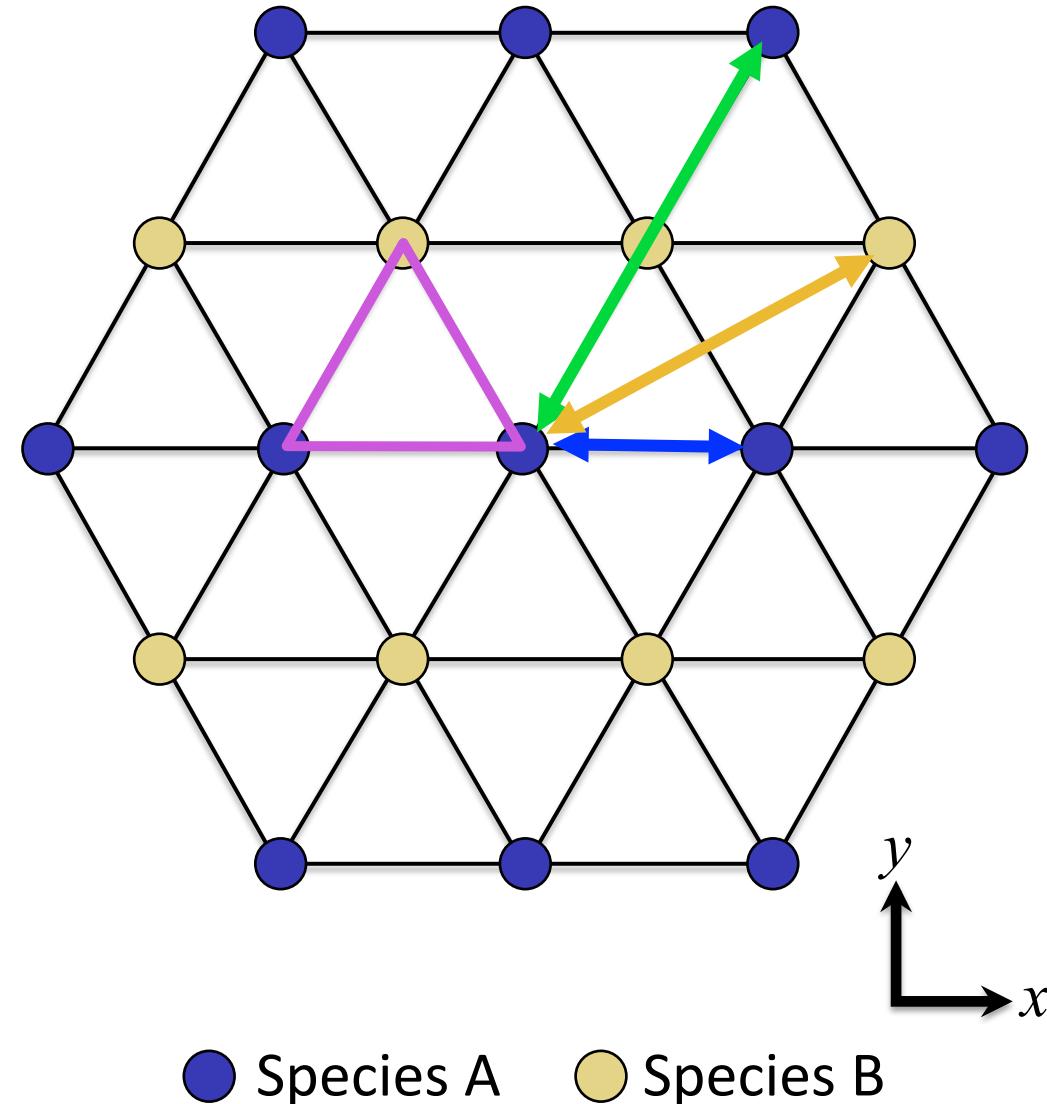
Symmetric averages of measurements of cluster occupation states.



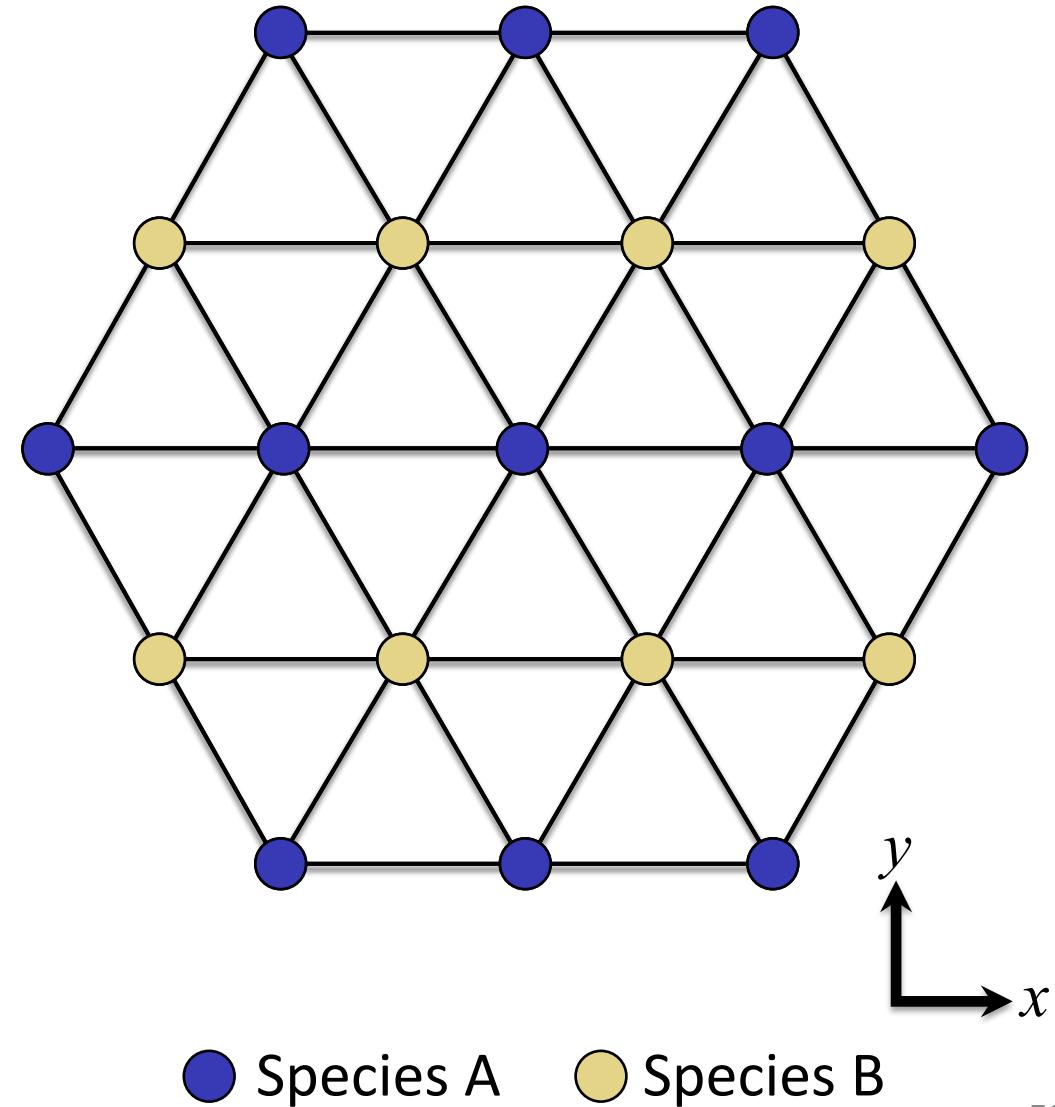
The Clusters Approach to Statistical Mechanics

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

Symmetric averages of measurements of cluster occupation states.



The Cluster Expansion

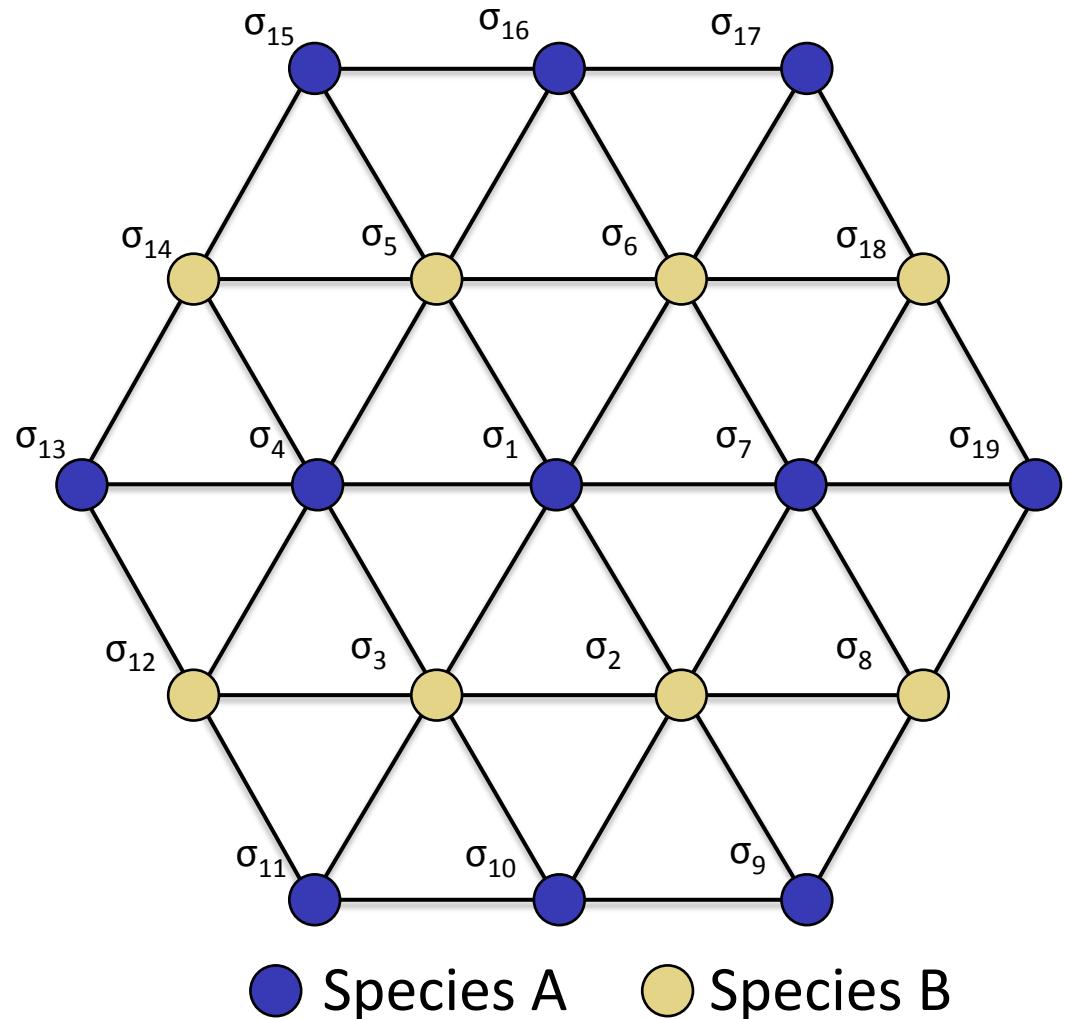


The Cluster Expansion

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

$\sigma_i = +1$ if species A

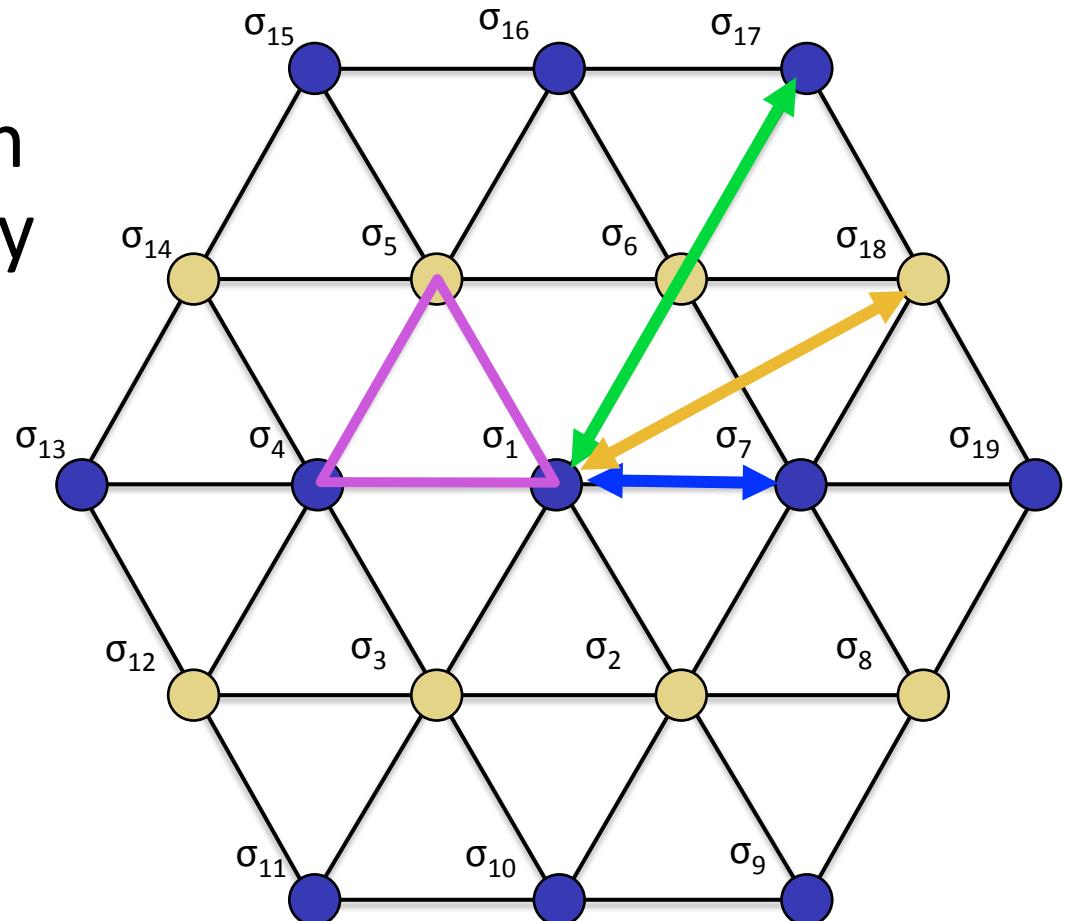
$\sigma_i = -1$ if species B



The Cluster Expansion

Interactions between sites are described by products of σ_i .

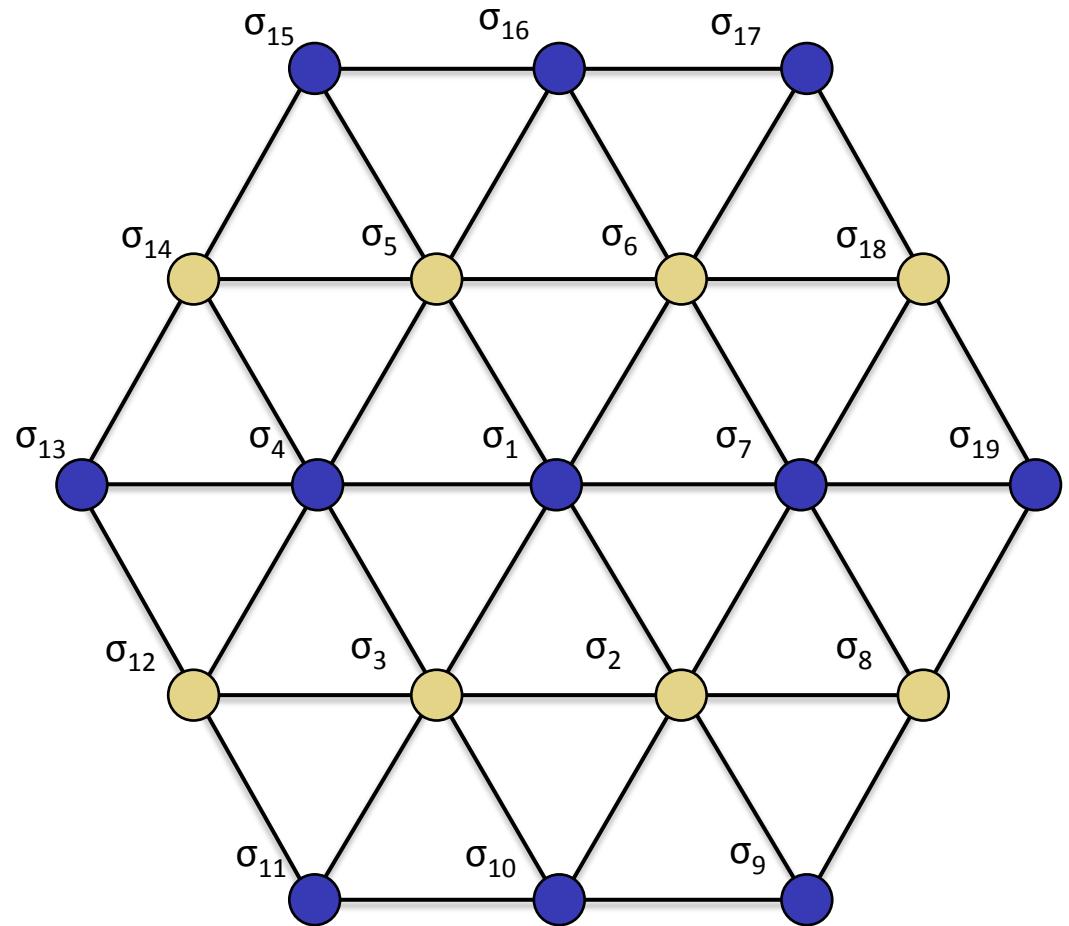
Fit interaction parameters to a database of DFT energies.



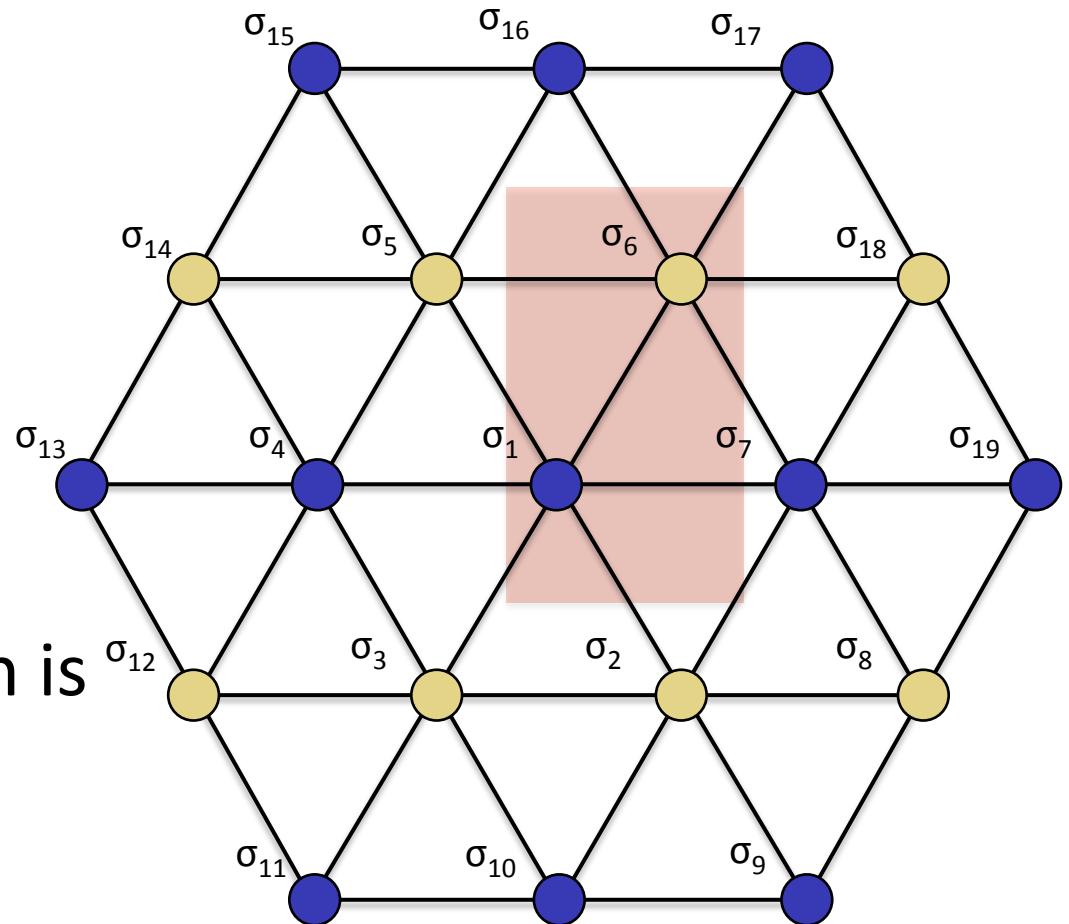
$$E = J_1 \sigma_1 + J_{1,7} \sigma_1 \sigma_7 + J_{1,18} \sigma_1 \sigma_{18} + J_{1,17} \sigma_1 \sigma_{17} + J_{1,4,5} \sigma_1 \sigma_4 \sigma_5 + \dots$$

Then use symmetry to average over the entire crystal

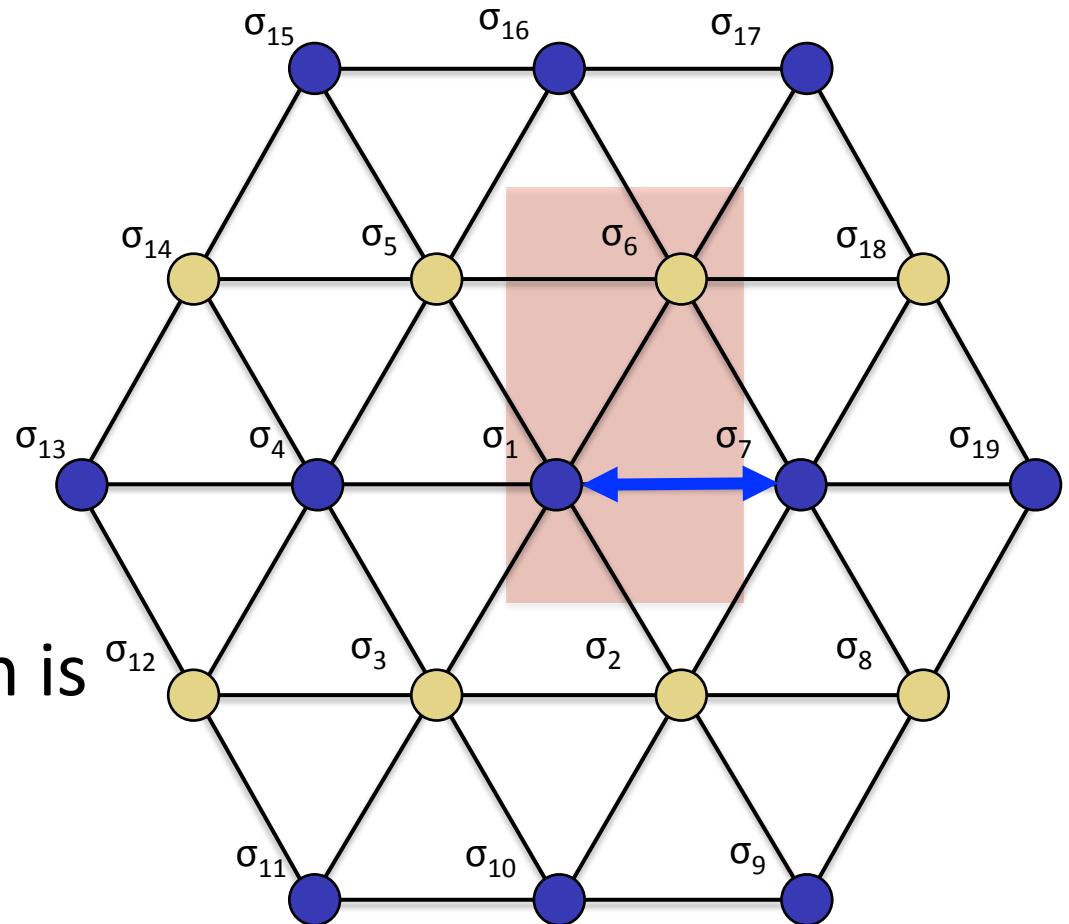
Cluster Expansion Example



Cluster Expansion Example

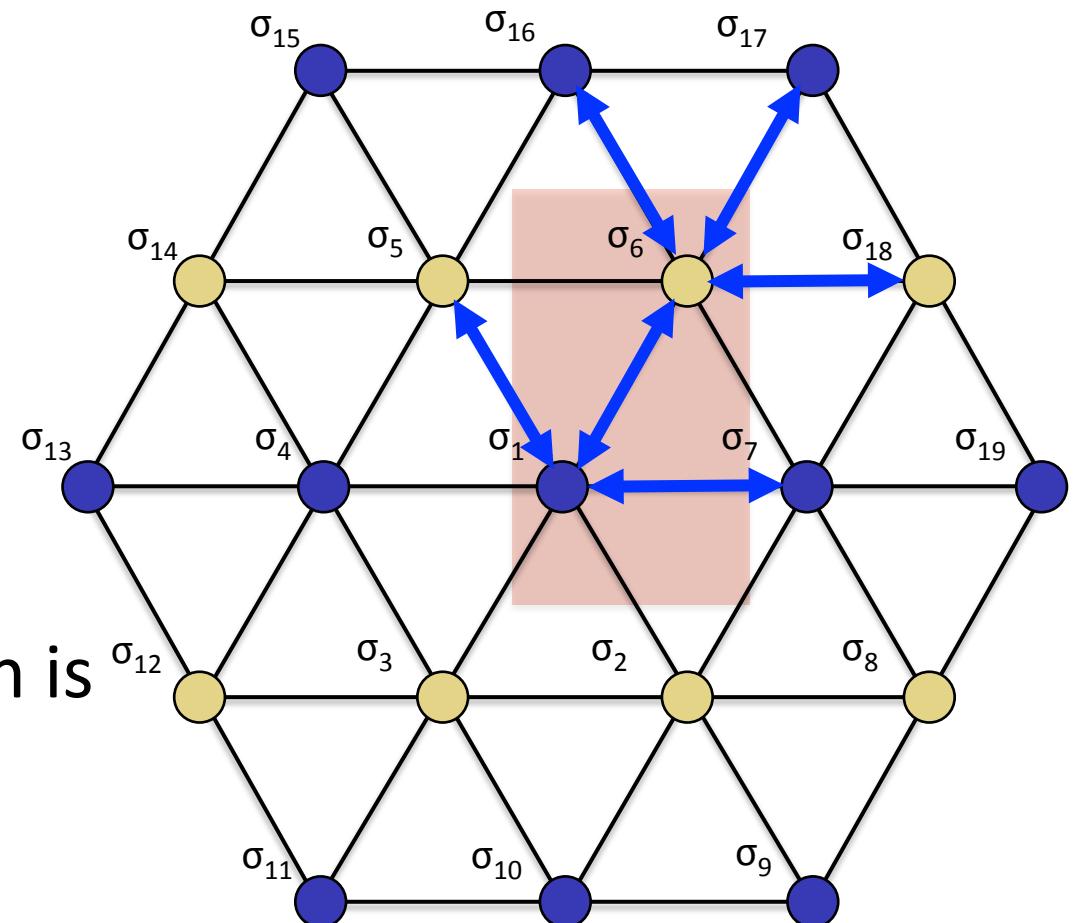


Cluster Expansion Example



Average site occupation is
 $\Gamma_1 = (\sigma_1 + \sigma_6)/2 = 0$

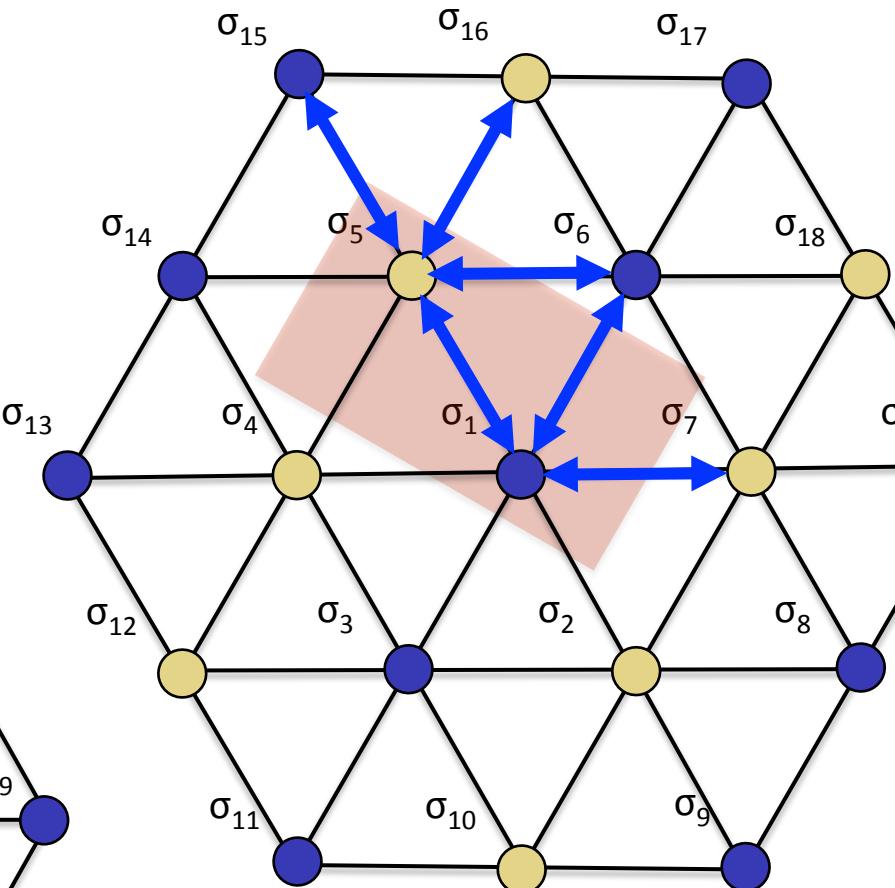
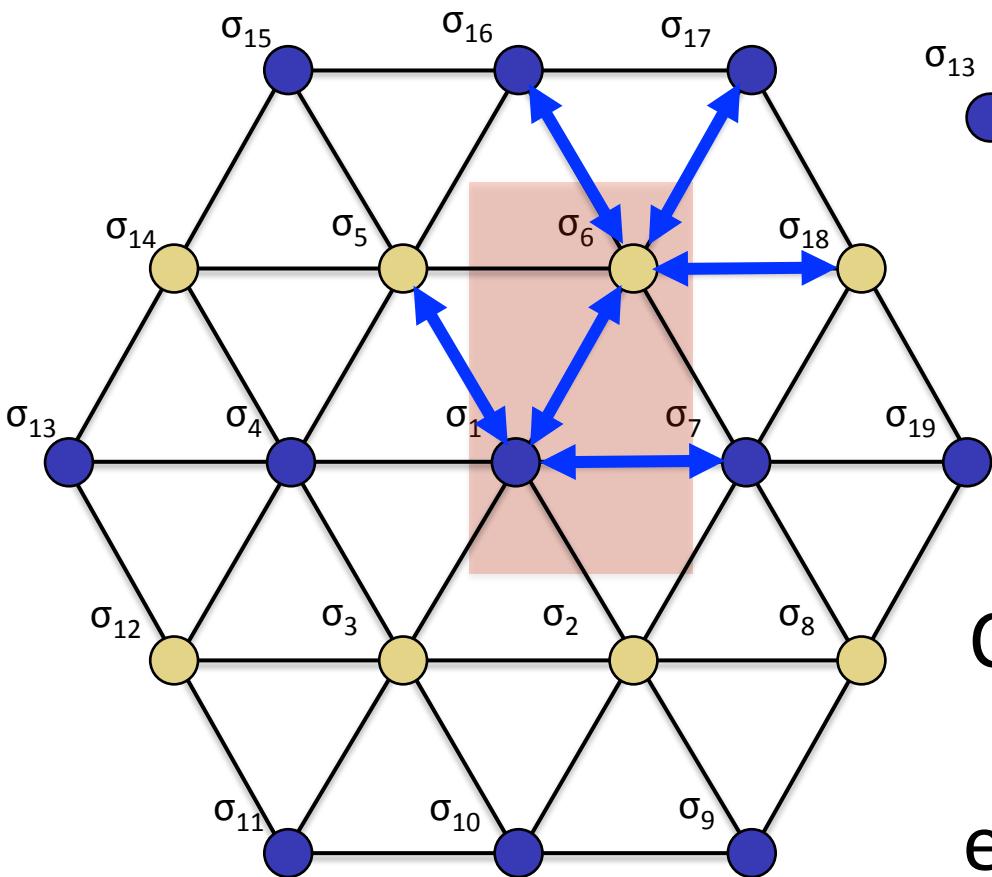
Cluster Expansion Example



Average site occupation is
 $\Gamma_1 = (\sigma_1 + \sigma_6)/2 = 0$

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



```
{  
    "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

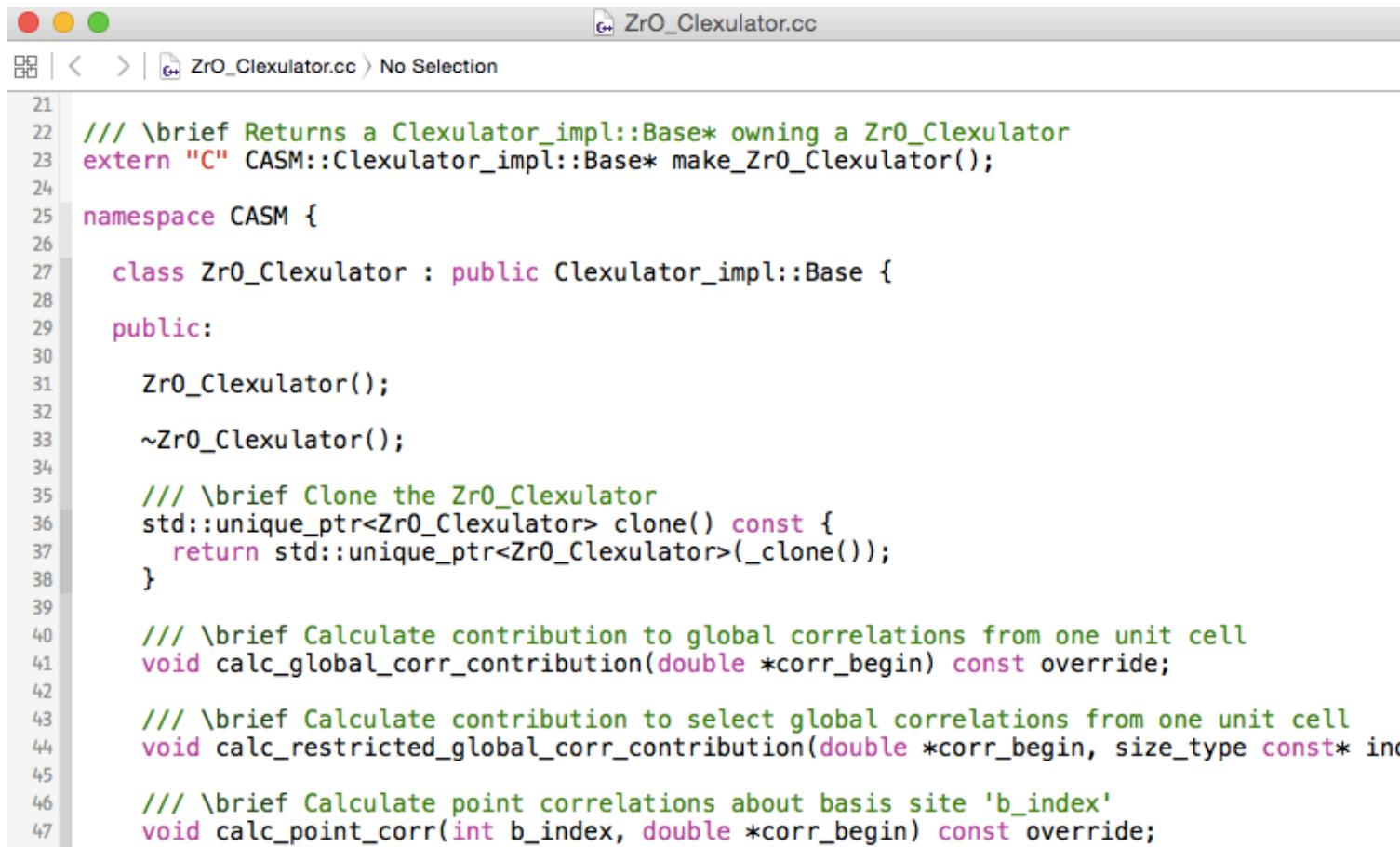
Lists one example
("prototype") of
each cluster type



```
[{"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

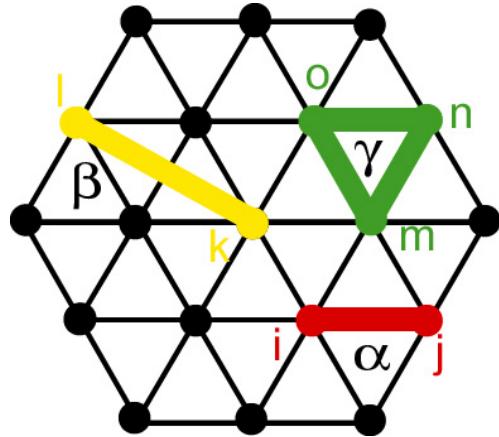


A screenshot of a code editor window titled "ZrO_Clexulator.cc". The file contains C++ code for a class named "ZrO_Clexulator". The code includes several methods with detailed documentation comments (/// \brief) and some inline comments. The code is color-coded for readability.

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

Generated code for evaluating basis functions

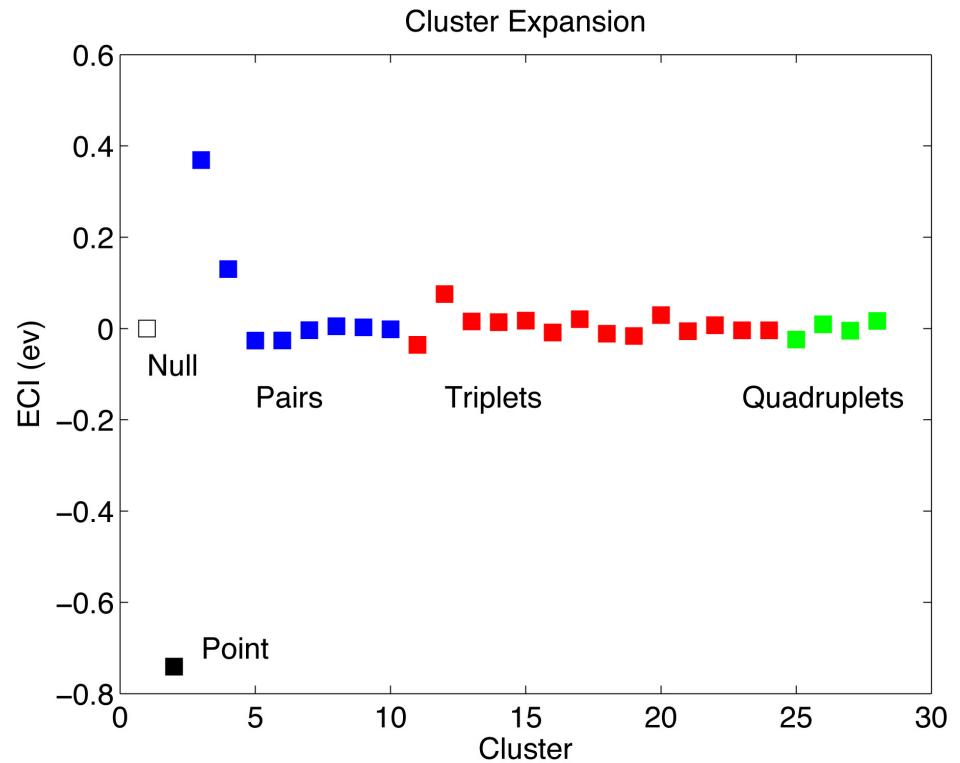
Cluster expansion for hcp Zr-O



$$\Phi_{\alpha} = \sigma_i \sigma_j$$

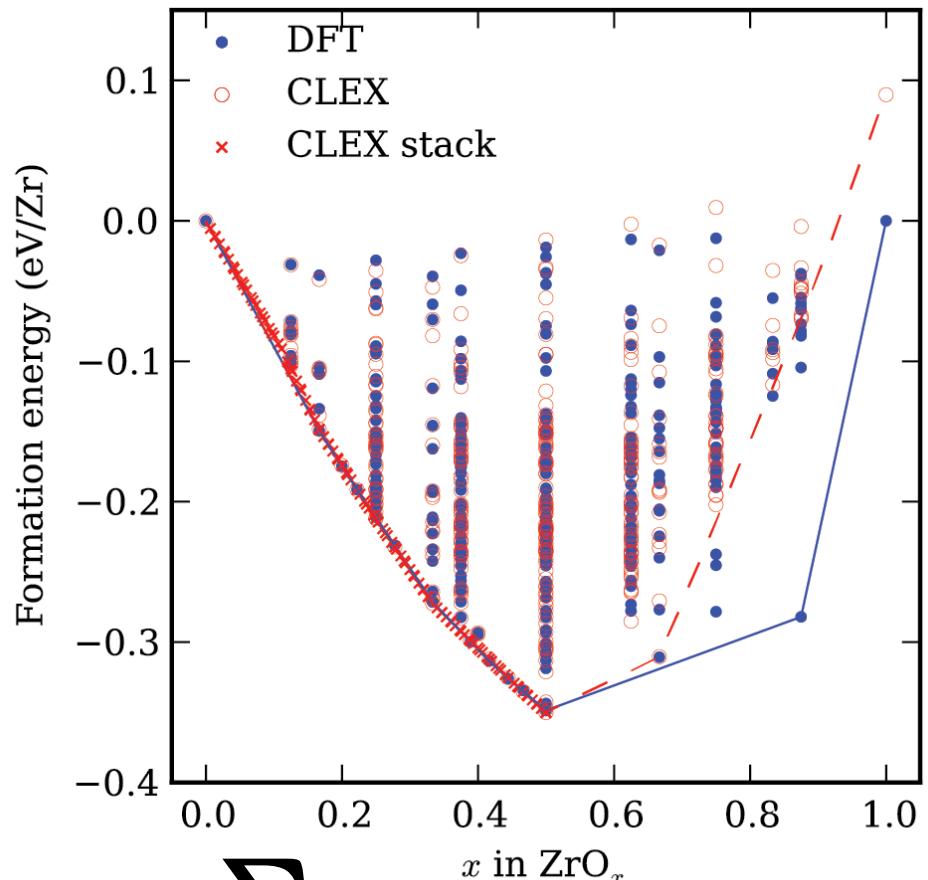
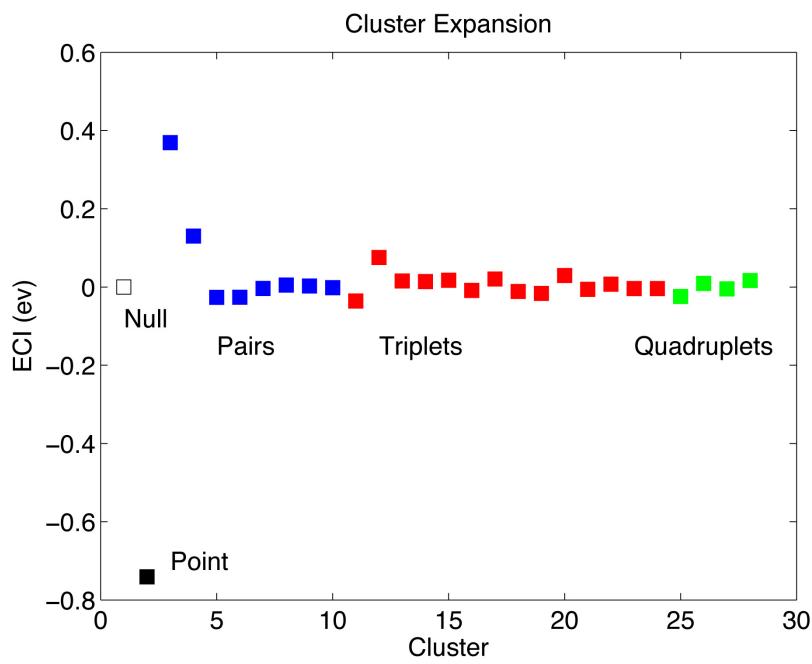
$$\Phi_{\beta} = \sigma_l \sigma_k$$

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



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

Cluster expansion for hcp Zr-O



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

CASM: Clusters Approach to Statistical Mechanics

First-principles energies of a “few” excitations

$$H = \sum_{i=1}^{N_e} \left(-\nabla_i^2 + V_{nuc}(r_i) \right) + \frac{1}{2} \sum_{i \neq j} \frac{1}{|r_i - r_j|} + E_{nuc}(\{R\})$$



Lattice Model Hamiltonian

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



Monte Carlo

$$Z = \sum_{\vec{\sigma}} \exp\left(-\frac{E(\vec{\sigma})}{k_B T}\right)$$



$$F = -k_B T \ln(Z)$$

Thermodynamics

Extrapolate to
all other excitations

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

After break:

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

