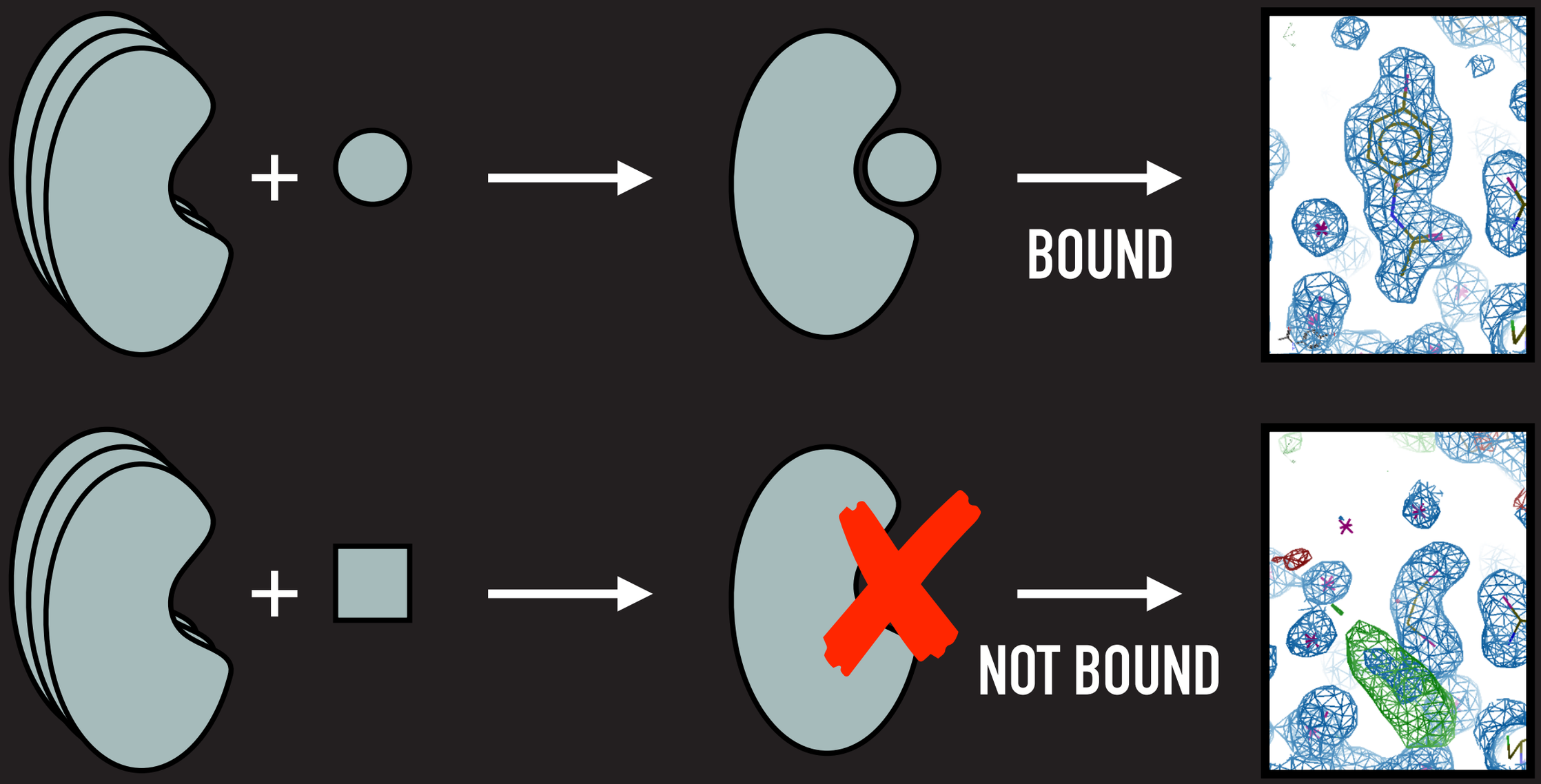
Generating and refining multi-state models

PARTIAL OCCUPANCY FEATURES

LIGAND BINDING IN CRYSTALLOGRAPHY

- Study ligand-binding in protein crystals — does it bind?
- If ligand binds, it appears in the electron density.



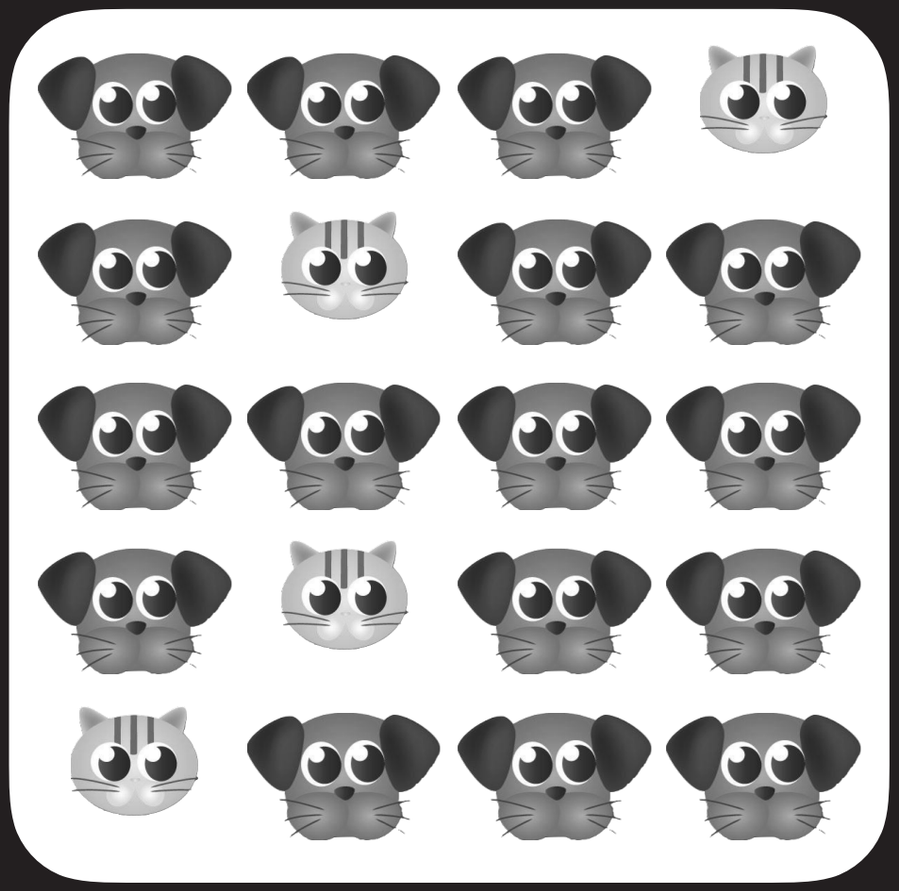
PROBLEMS: PARTIAL OCCUPANCY

What if the ligand only binds to a fraction of the crystal?

Crystal:
Contains bound (cat) and unbound (dog) states

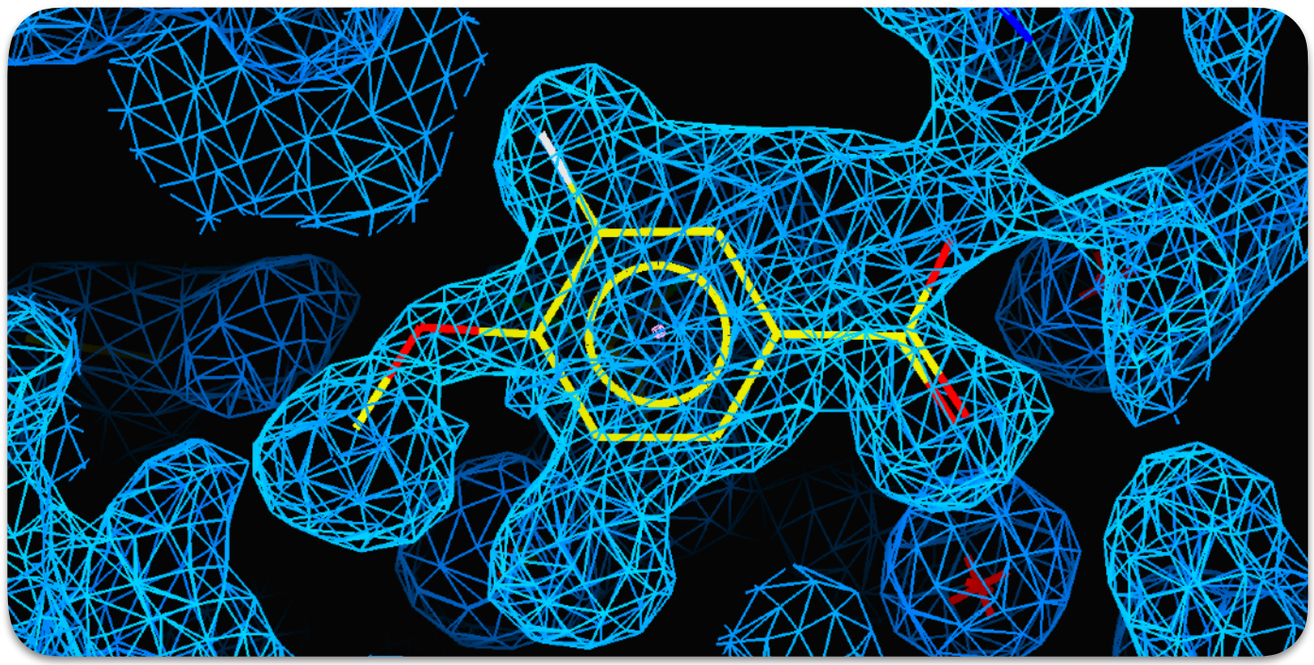
Diffraction:
Average over the crystal (80% dog + 20% cat)

Density:
Looks like the dominant state



EXAMPLE: OBSCURED LIGAND

Standard $2mF_o$-DF_c Map (1.39Å; 0.3rmsd)



Normal density maps show only the superposition of the full crystal.

They are not useful for identifying partial occupancy features.

REVISITING PARTIAL OCCUPANCY

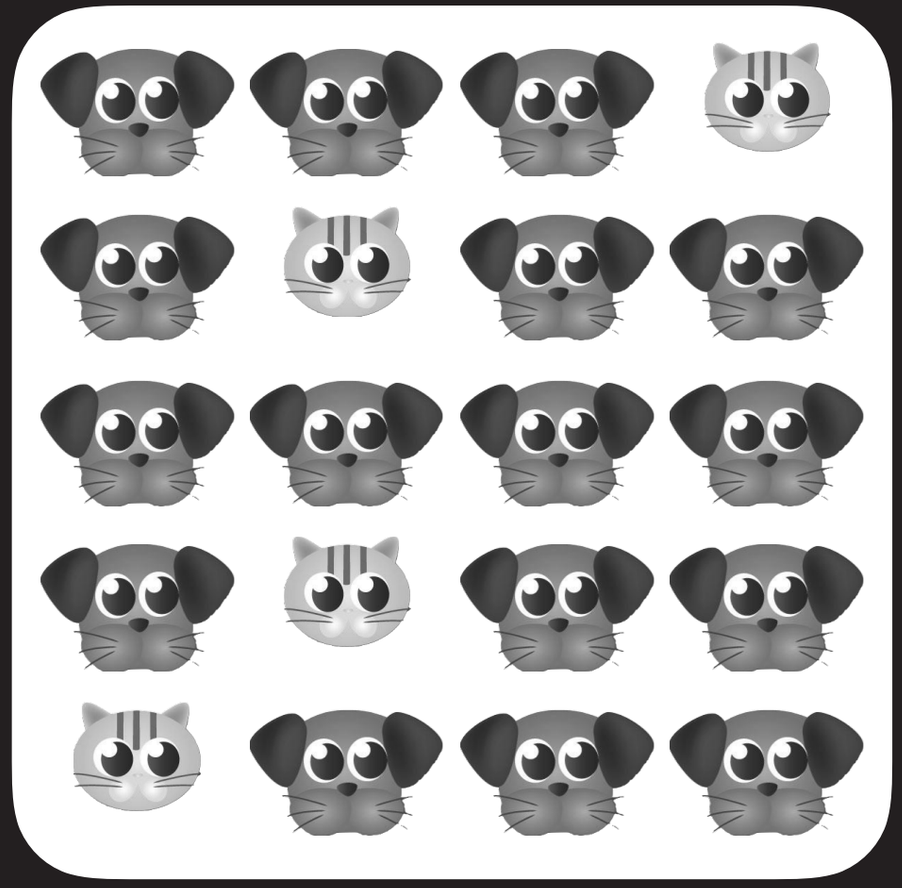
The ligand is bound to a fraction of the crystal:

80% dog + 20% cat

The model should reflect the crystal content:

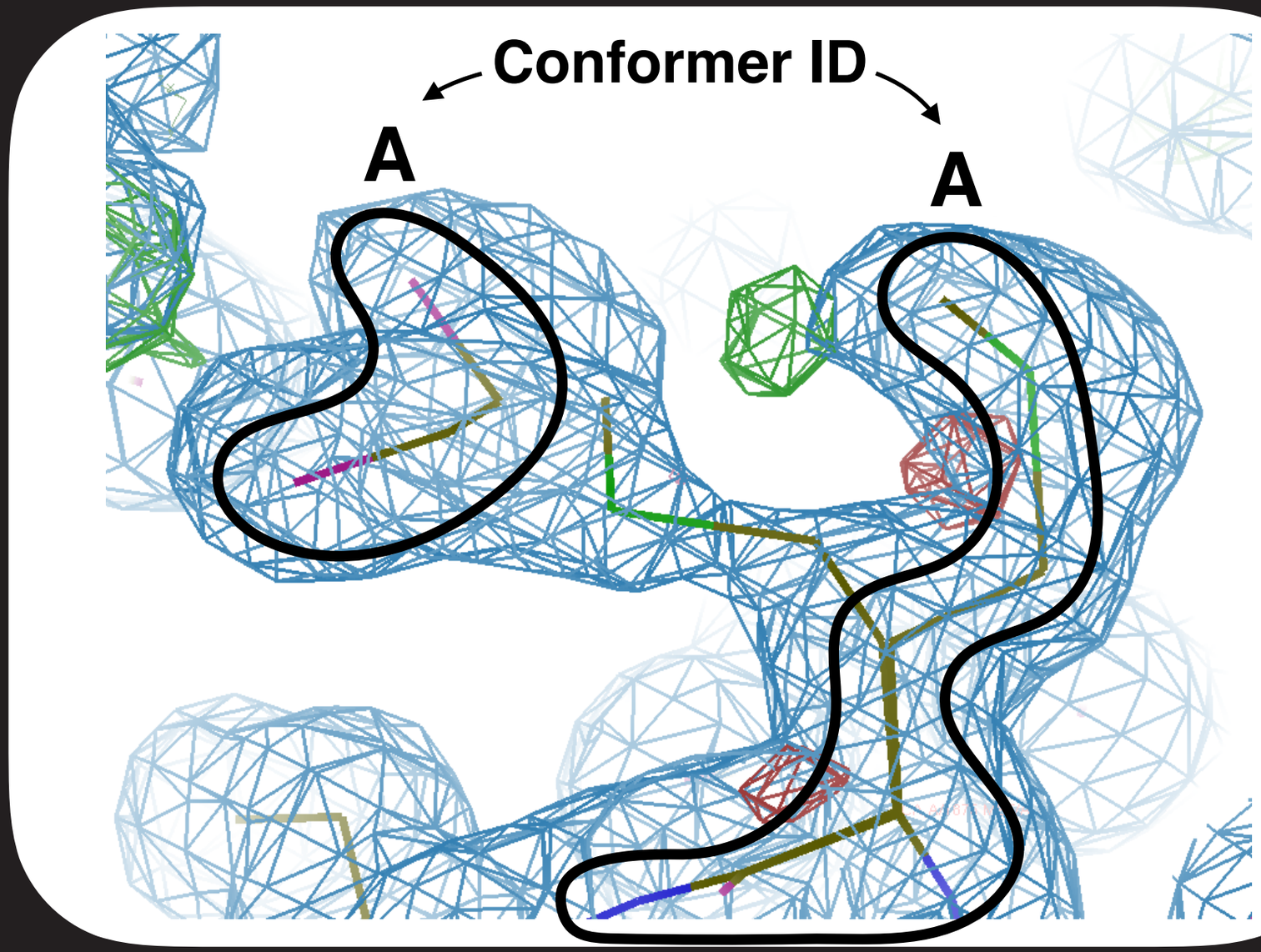
80% dog + 20% cat

or 20% dog + 80% cat !



NOTE ON MULTI-CONFORMER MODELS

- Crystallographic density is an average over many states
- “Alternate conformers” model these different crystal conformations

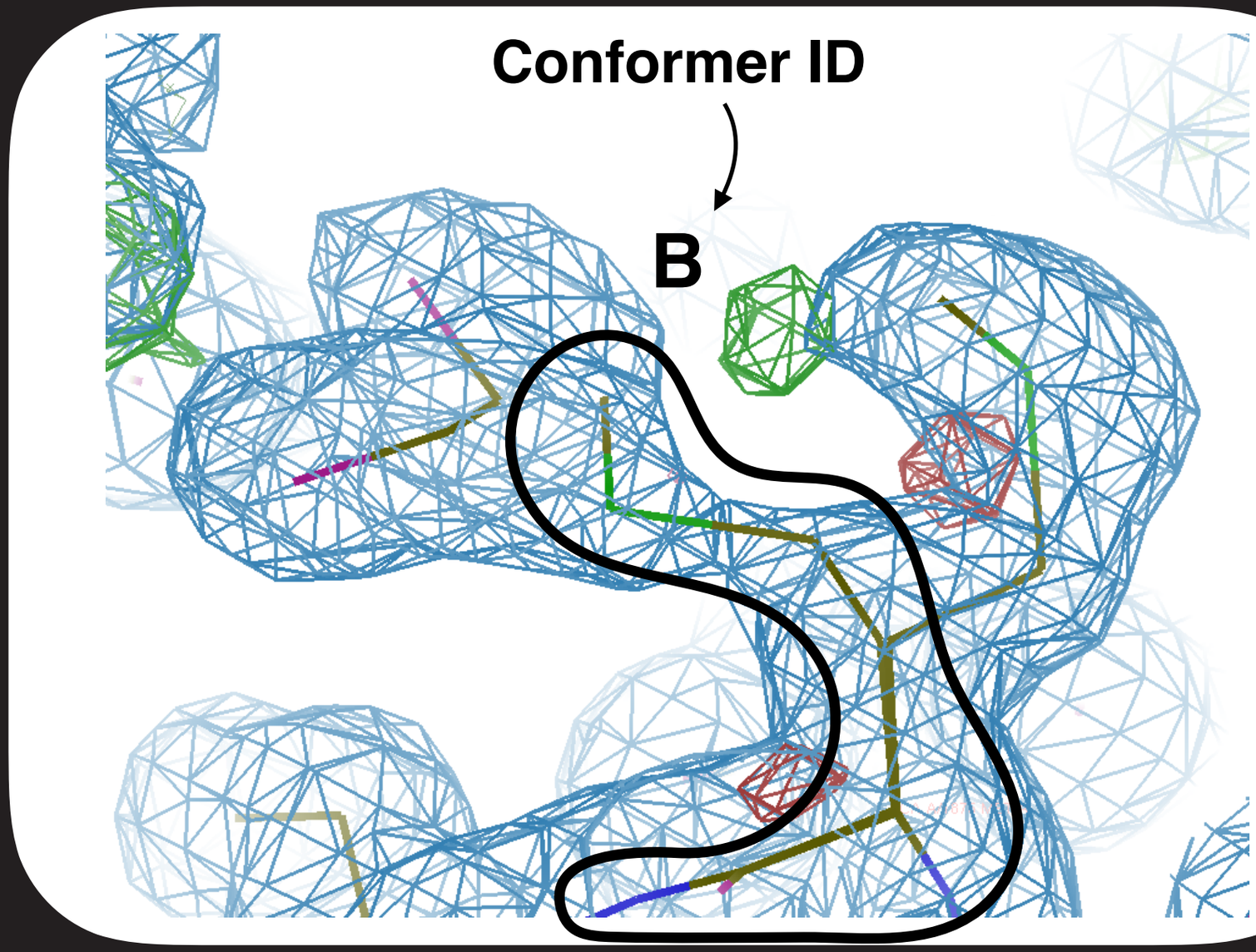


$2mF_o$-DF_c
blue
@ 1rmsd

mF_o-DF_c
green/red
@ ±3rmsd

NOTE ON MULTI-CONFORMER MODELS

- Crystallographic density is an average over many states
- “Alternate conformers” model these different crystal conformations



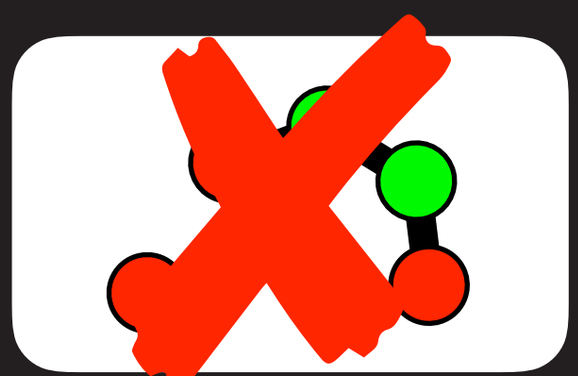
$2mF_o$-DF_c
blue
@ 1rmsd

mF_o-DF_c
green/red
@ ±3rmsd

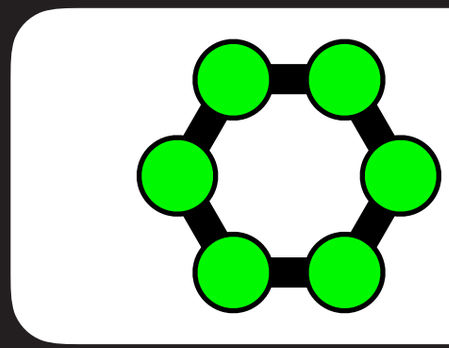
PARTIAL-OCCUPANCY MODELLING & REFINEMENT

NORMAL APPROACH

DELETE INPUT MODEL

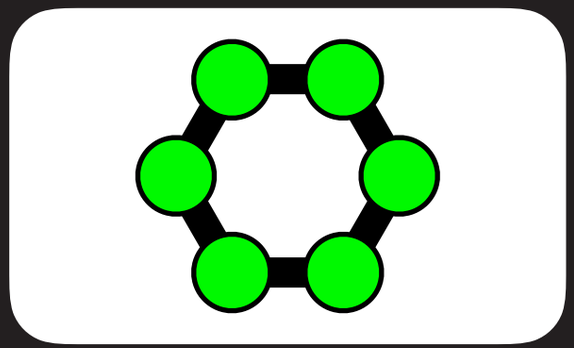


MODEL LIGAND



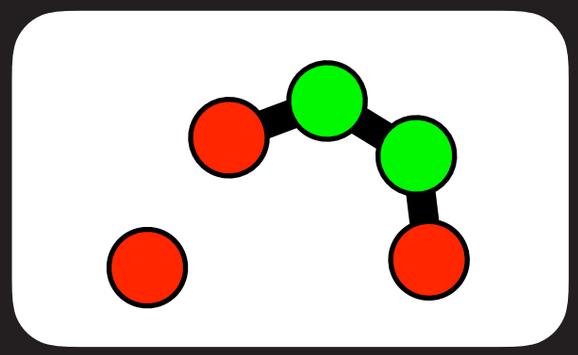
=

FINAL MODEL

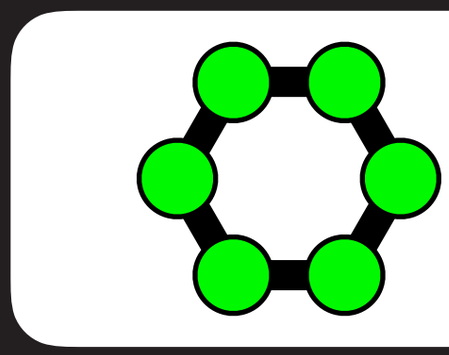


CORRECT APPROACH

COMBINE INPUT MODEL AND LIGAND MODEL

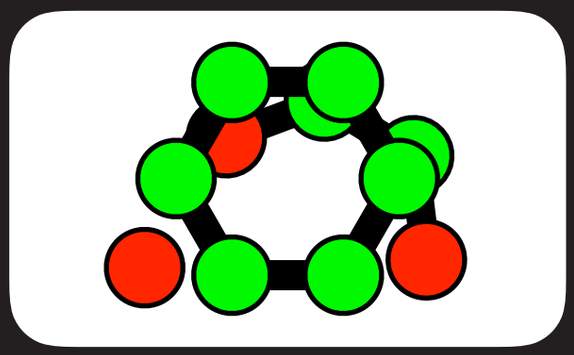


+



=

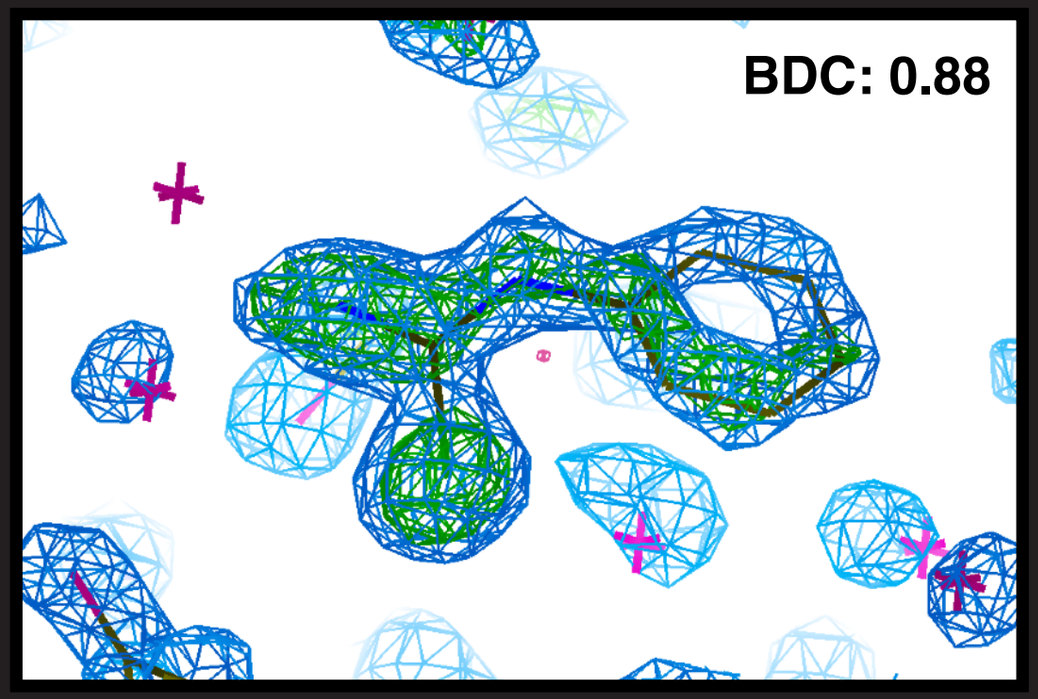
ENSEMBLE MODEL



PARTIAL-OCCUPANCY MODELLING & REFINEMENT

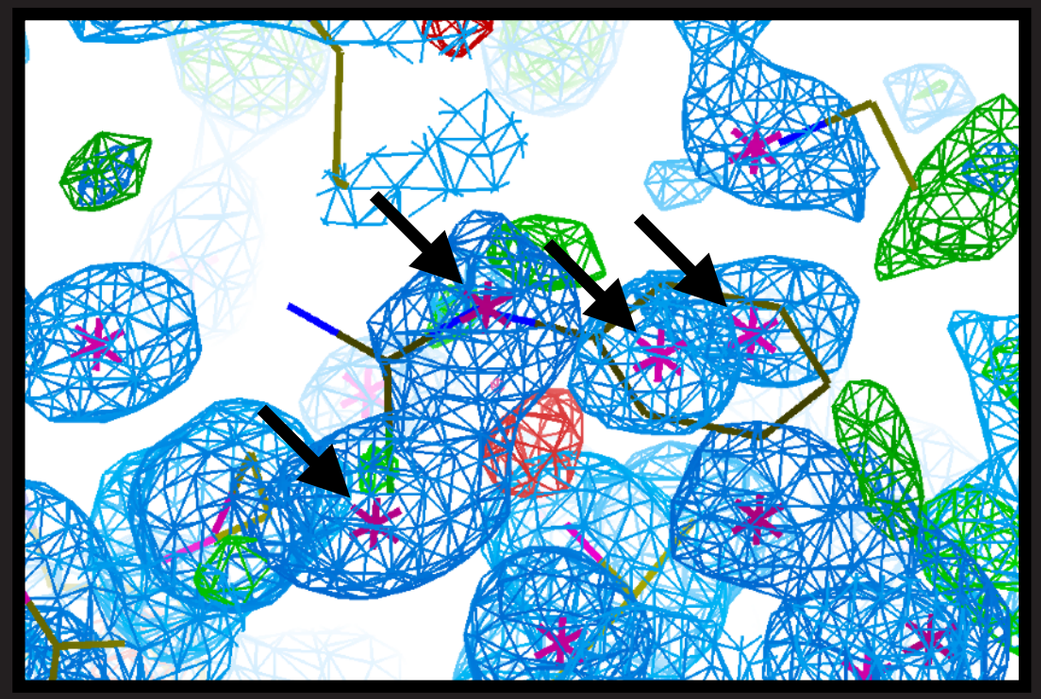
Resolution: 1.39Å; Occupancy: 0.23

PANDDA



EVENT (blue, 2rmsd)
ZMAP (green/red, ±4)

REFINED



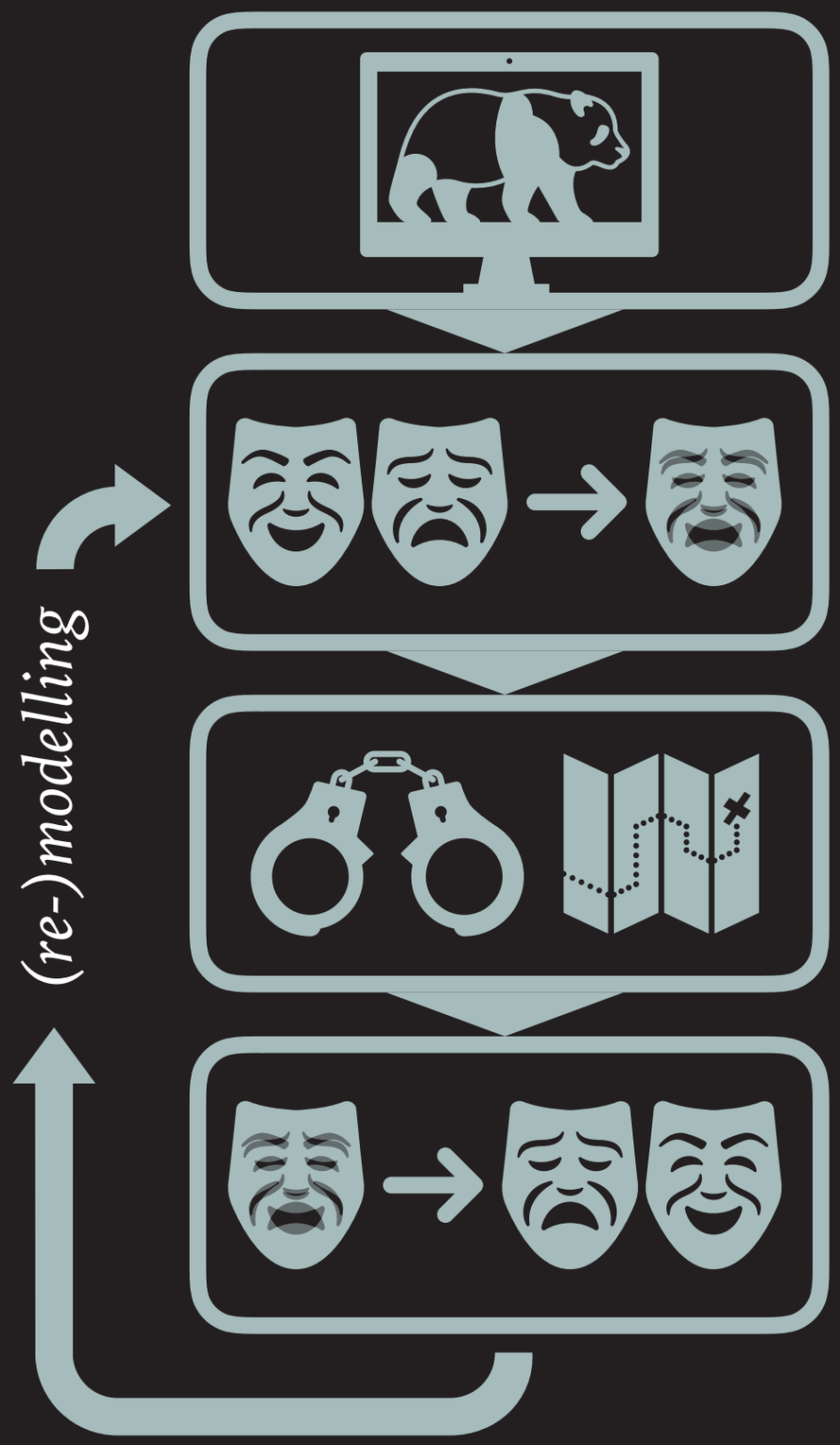
2FOFC (blue, 0.5rmsd)
FOFC (green/red, ±3rmsd)

- Merging the solvent model from another dataset creates a good model
- Ligands do not “appear” in the maps after refinement (nor should they!)

MULTI-STATE REFINEMENT PROTOCOL

From PanDDA ⇝ Multi-state models

OVERALL PROTOCOL



Analysis & Modelling

pandda.analyse & pandda.inspect

Generation of Ensemble Model

giant.merge_conformations (pandda.export)

Restraints & Refinement

giant.make_restraints (& giant.quick_refine)

Separation of Ensemble States

giant.split_conformations

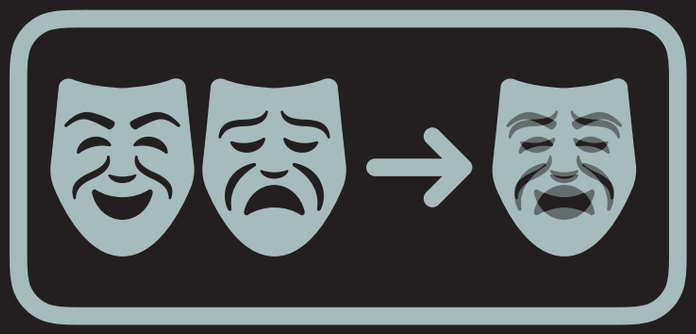
Analysis & Modelling

pandda.analyse & pandda.inspect

- Documentation on https://pandda.bitbucket.io
  - Soon to be updated to match new version (v0.2.X)

- Output for each dataset:
  - 1 model for the ground-state of the crystal
  - 1 model for the bound state of the crystal (or changed-state)

- Need to merge models to allow refinement



Generation of Ensemble Model

giant.merge_conformations (pandda.export)

- Need to create a **crystallographically correct** model for refinement
- Define: “Crystallographically correct”
  - Conformer “A” only “sees” atoms of conformer “A” or “ ” (no conf.).
  - Cannot have residue of “C” bound to/interacting with residue of “A”
  - Only can have “A”-“A” and “C”-“C”
- Takeaway message: expand conformers explicitly during merging.
- (Performed automatically in pandda.export).

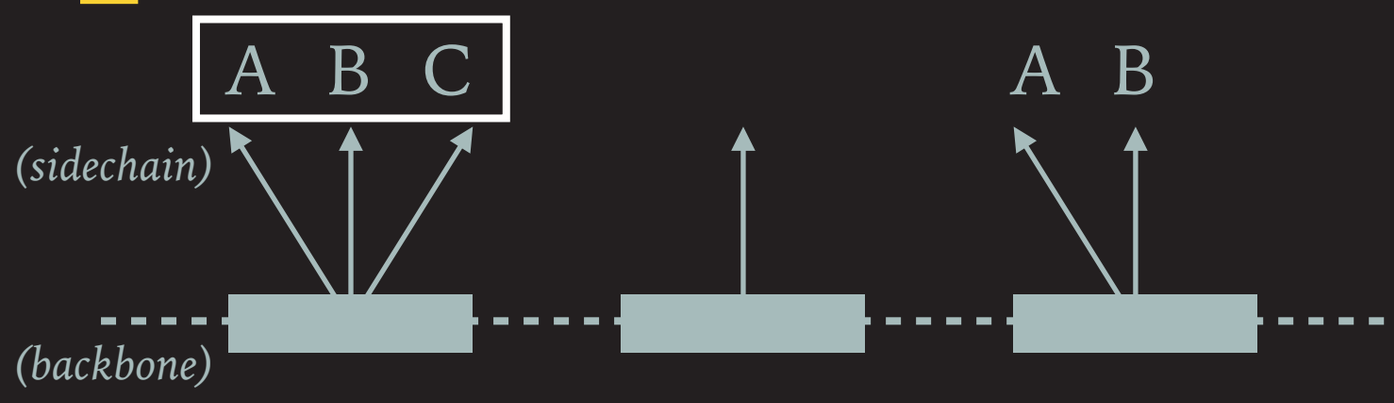
Generation of Ensemble Model

giant.merge_conformations (pandda.export)

(“ ” indicates atom with no conformers: “main conf.”)	Model 1	Model 2
1) Identify conformers in each model	“ ”, A, B	“ ”, A, B, C
2) Create missing confs. from existing confs. (make any protein residue with alternate confs. have all confs.)	- (n/a for two confs)	e.g. A,B → A,B,C (for each residue)
3) Copy atoms with no conformer to all conformers	“ ” → A,B	“ ” → A,B,C
4) Increment conformers so not overlapping	- (n/a for model 1)	A→C, B→D, C→E
5) Copy all atoms from model 2 into model 1	A, B, C, D, E	
6) Remove confs. for atoms where all confs. are the same	“ ”, A, B, C, D, E	

Generation of Ensemble Model

1) Identify conformers in each model

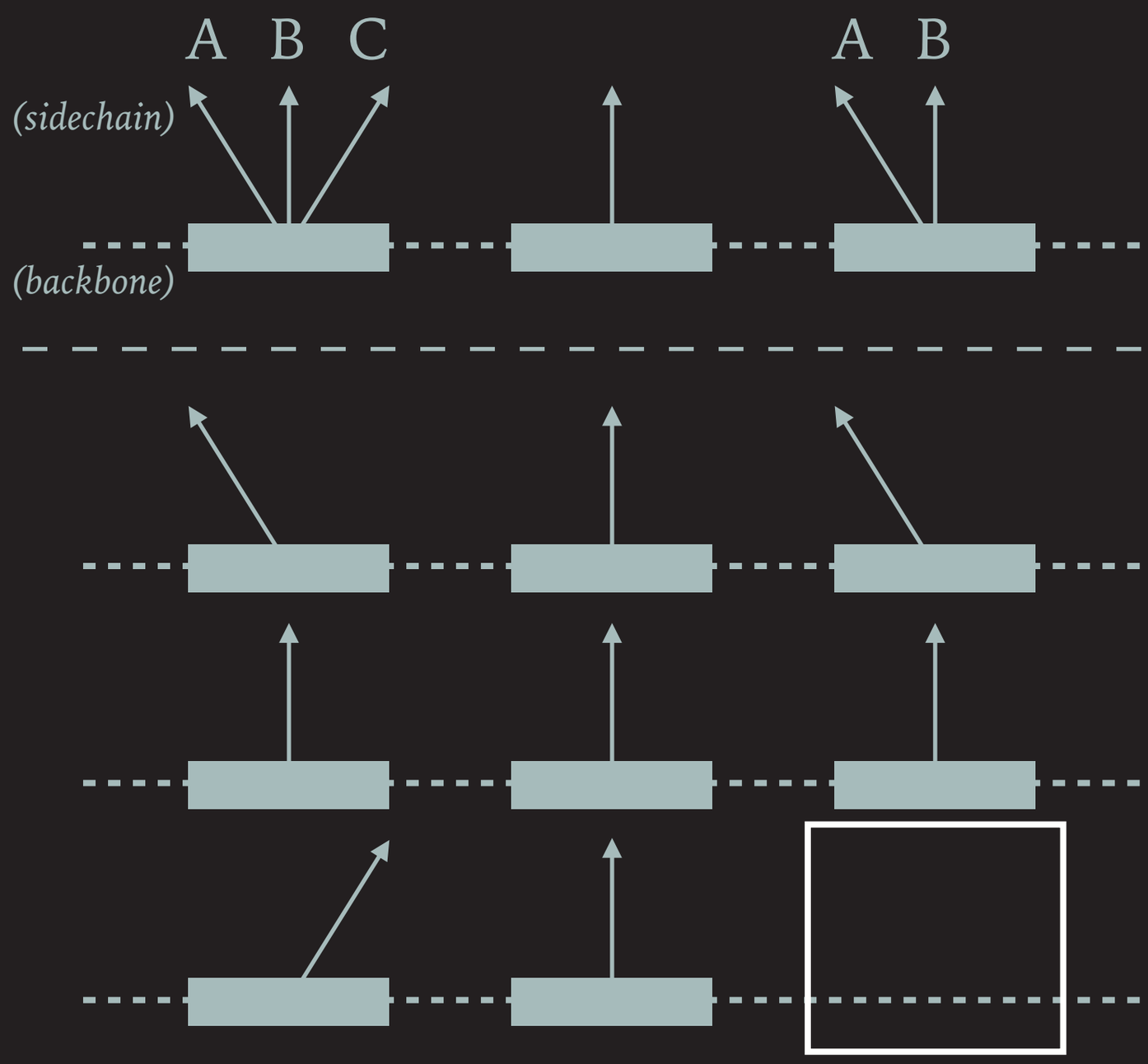


Model has 3 conformers!

This is not the most complicated slide

Generation of Ensemble Model

2) Create missing confs. from existing confs.

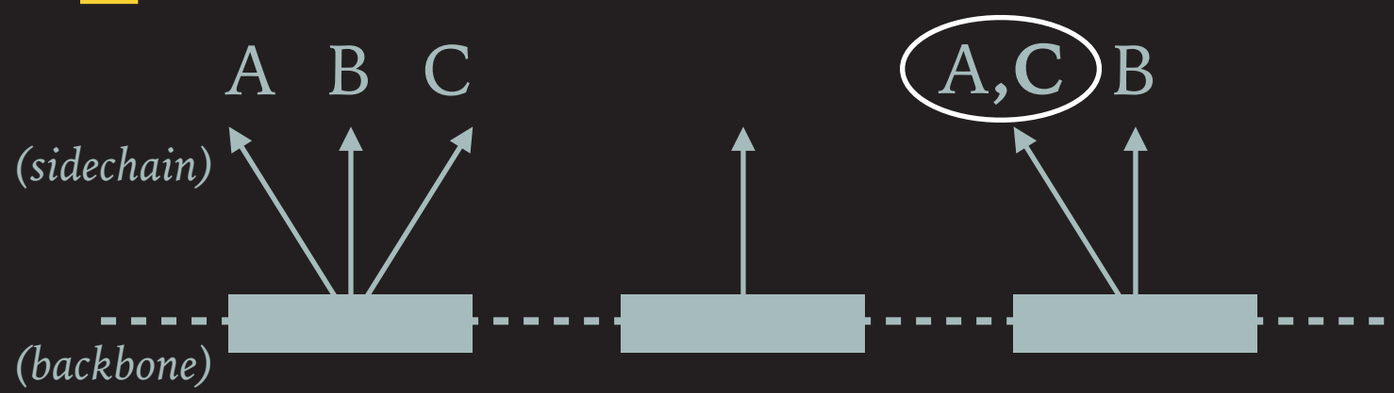


***NOT** a crystallographically correct model (some conformers not present for some residues)*

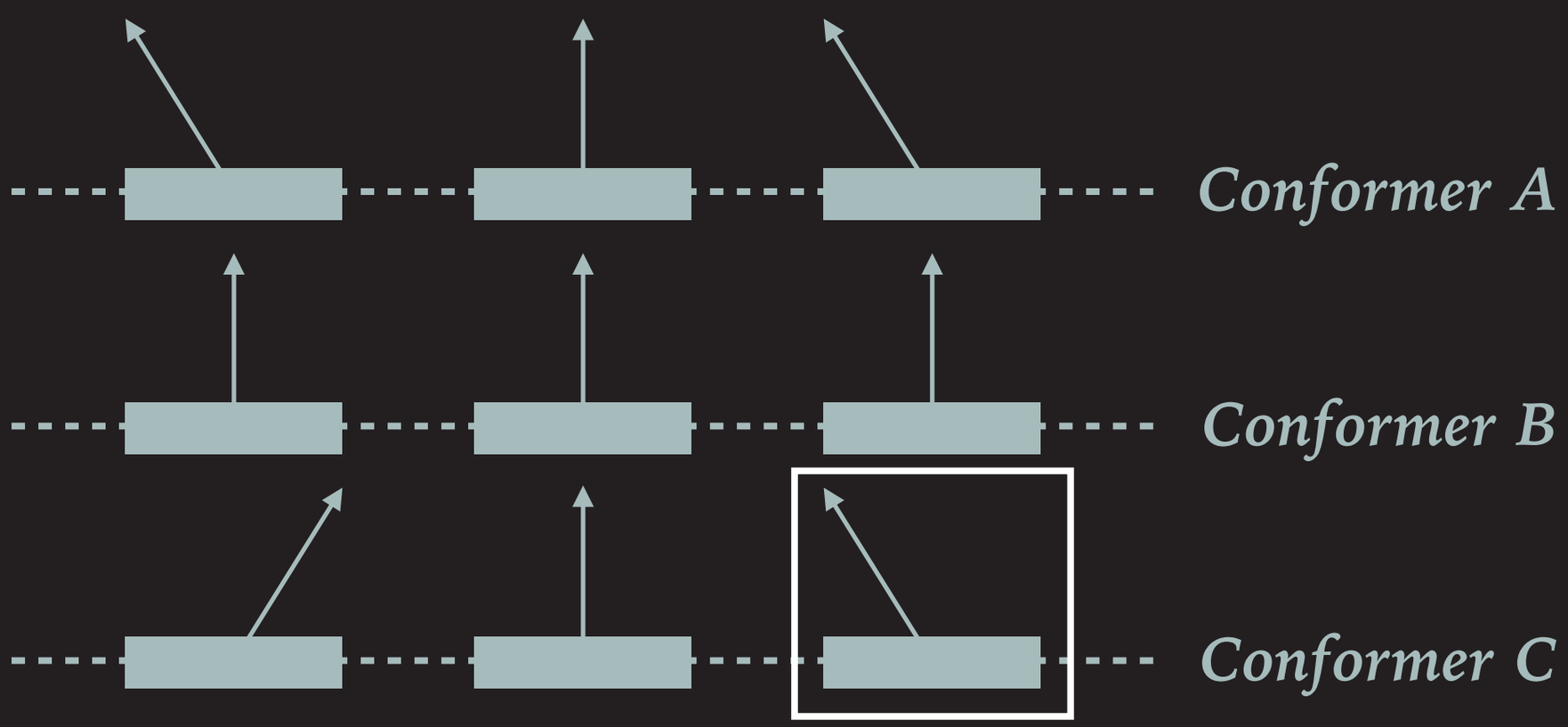
Conformer C is missing residues!

Generation of Ensemble Model

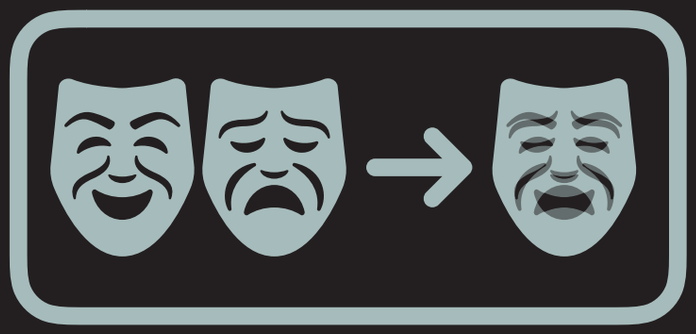
2) Create missing confs. from existing confs.



Copy existing conformers to fill in the gaps (best guess of appropriate states for "C")

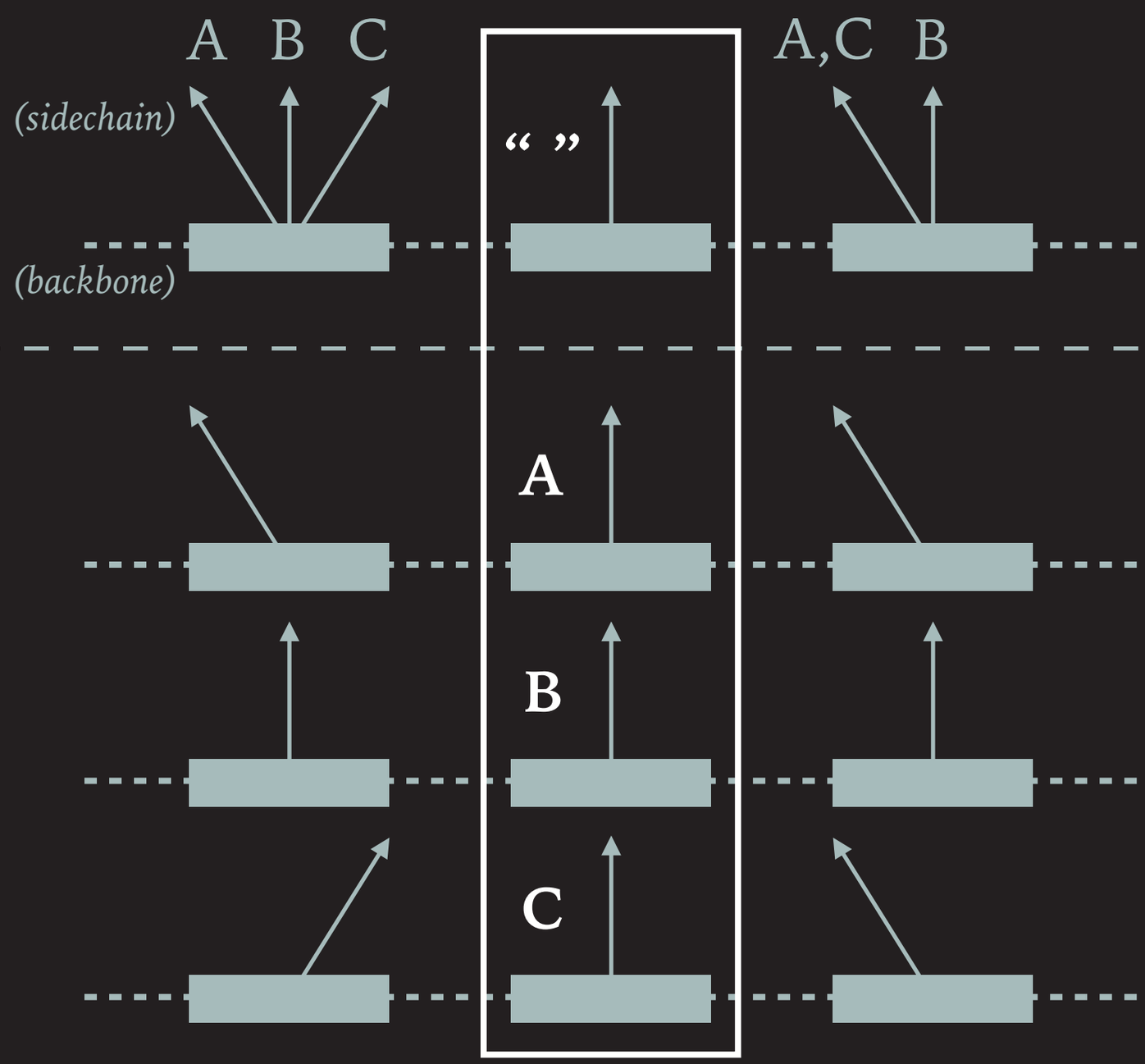


Conformer C is now complete!

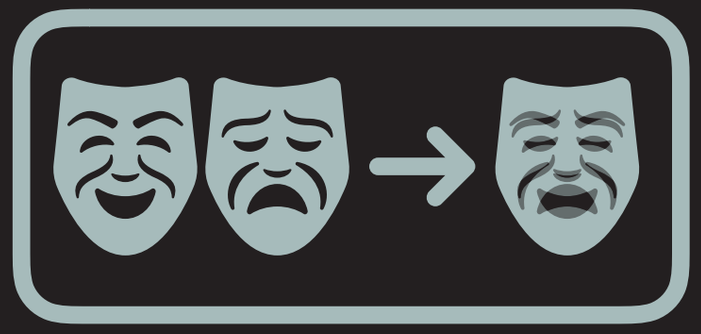


Generation of Ensemble Model

3) Copy atoms with no conformer

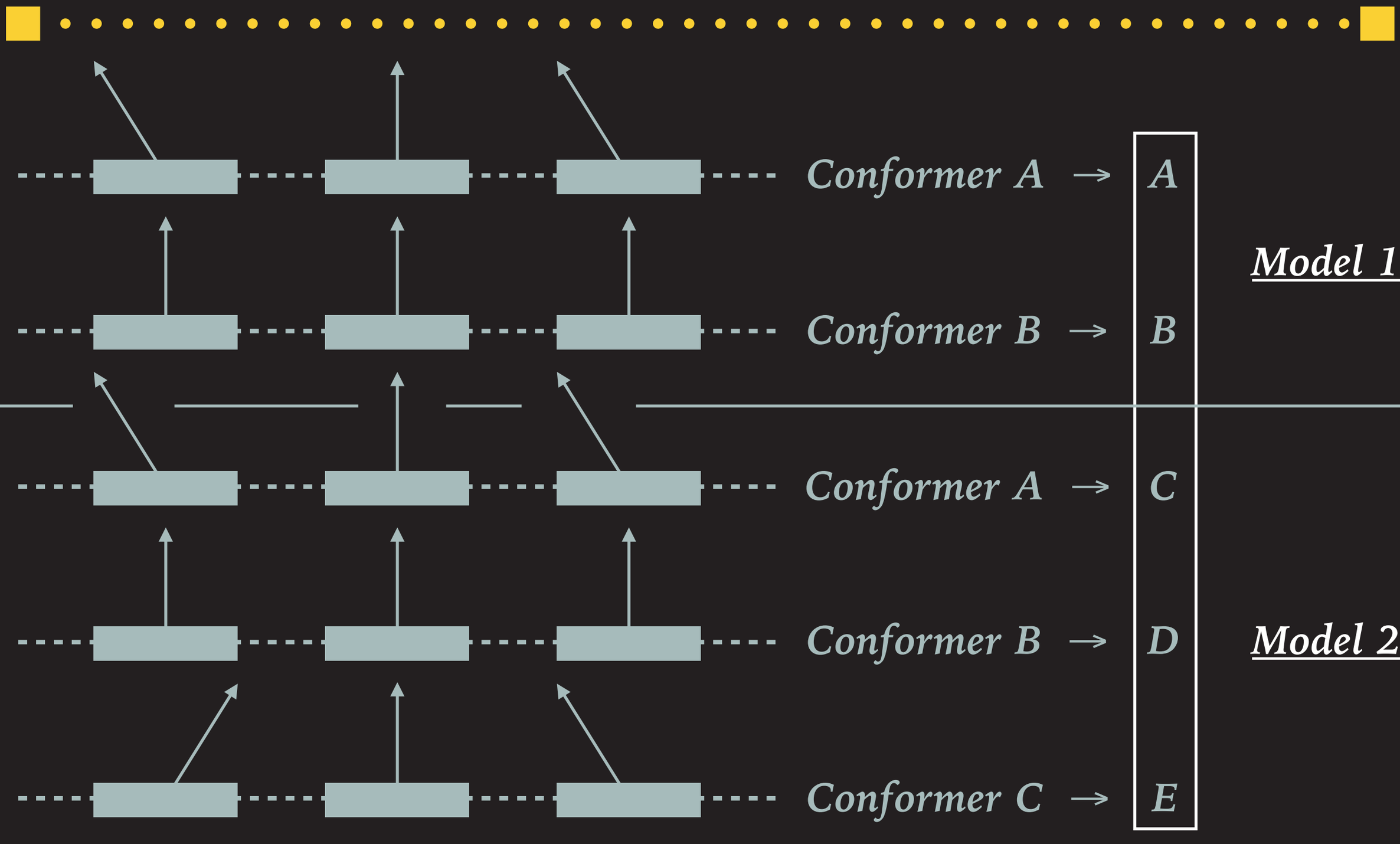


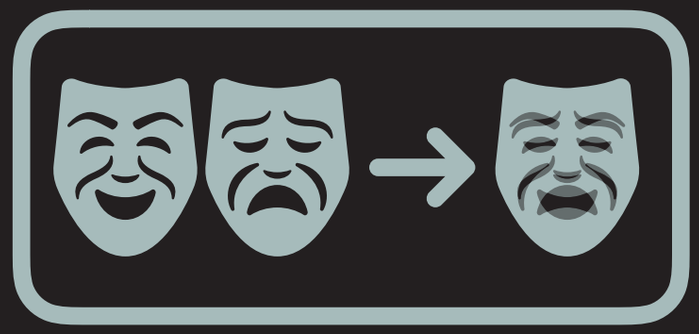
Explicitly expand (copy) single-conformation atoms to all conformers



Generation of Ensemble Model

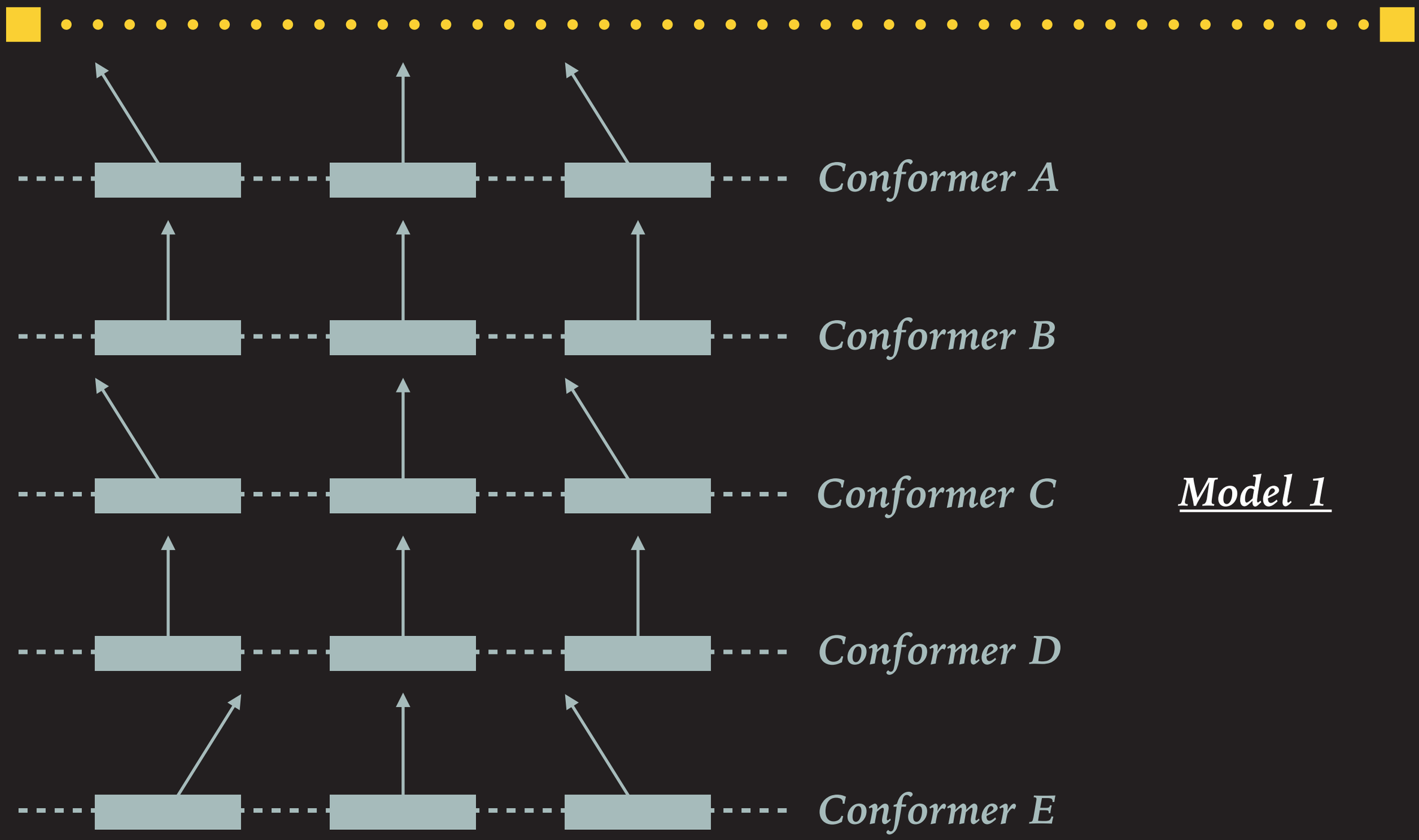
4) Increment conformers so not overlapping





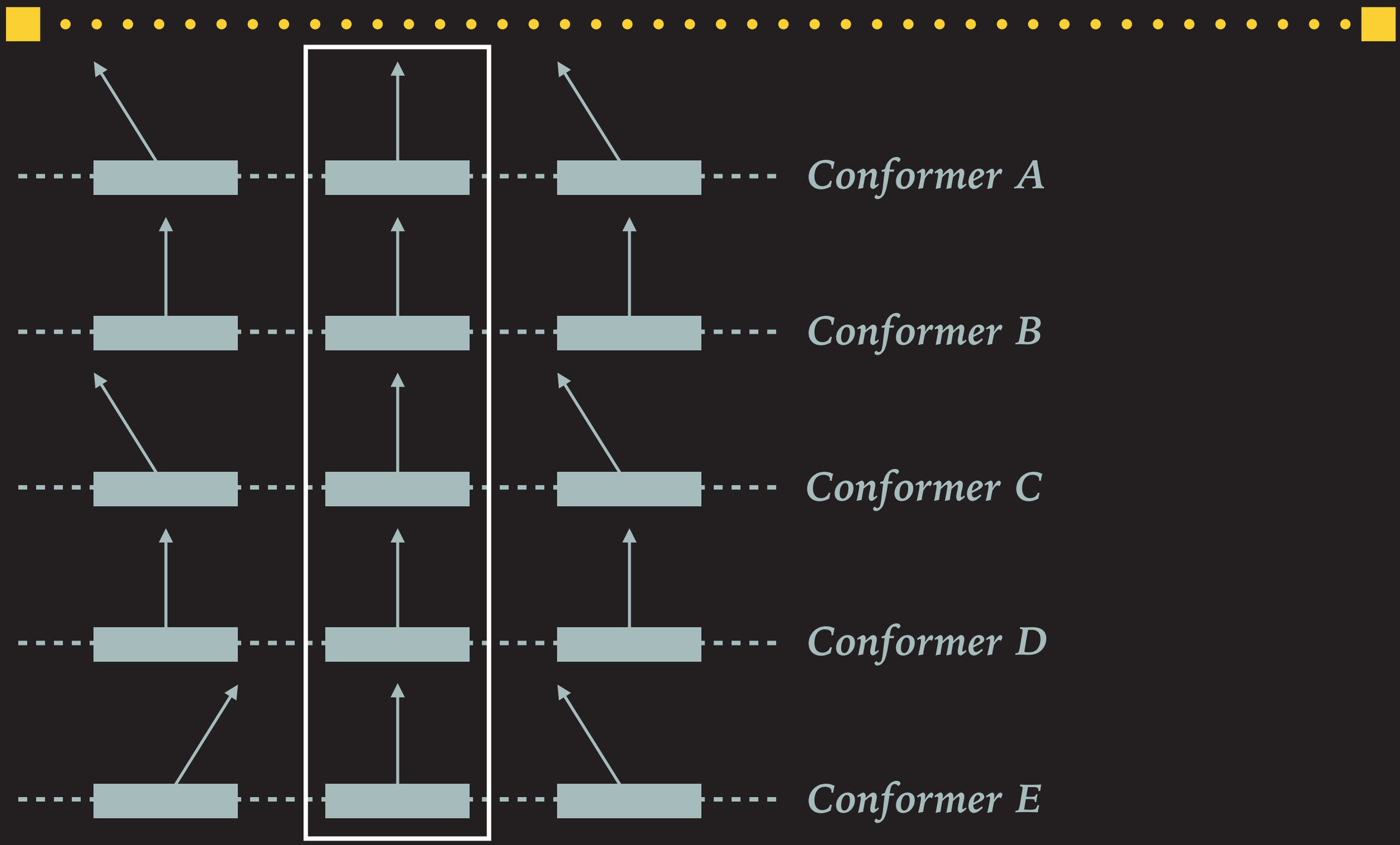
Generation of Ensemble Model

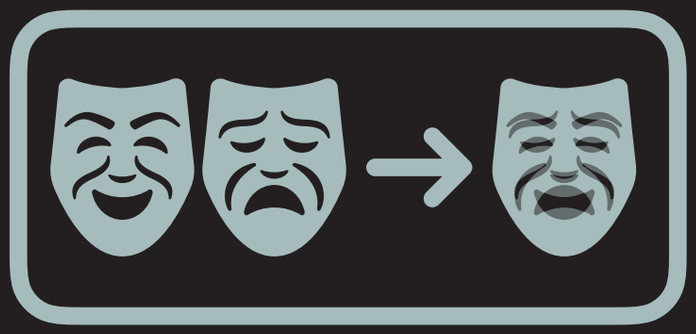
5) Copy all atoms from model 2 into model 1



Generation of Ensemble Model

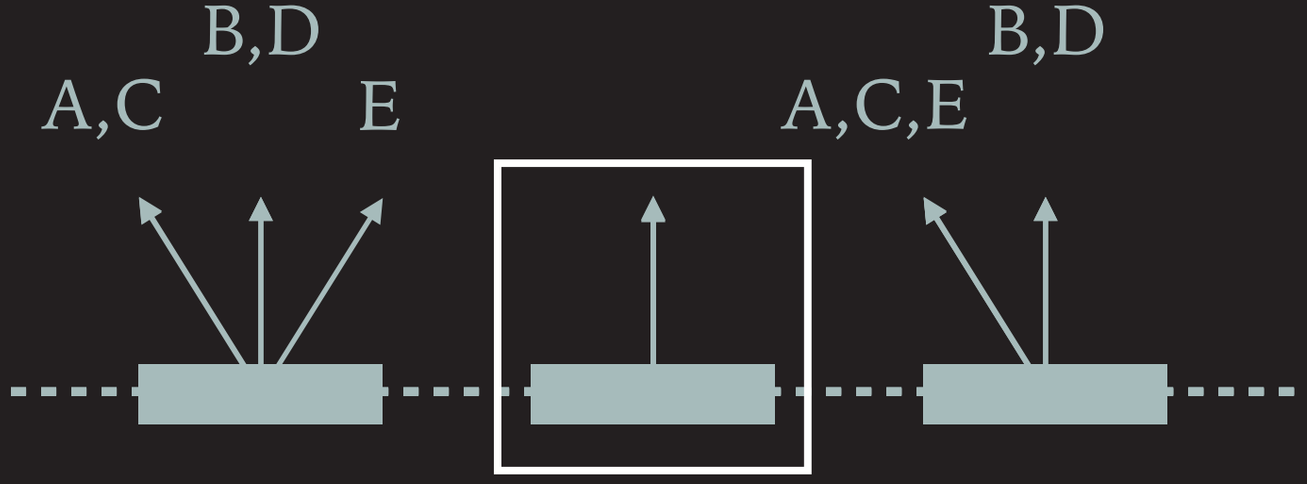
6) Remove confs. where all confs. are the same

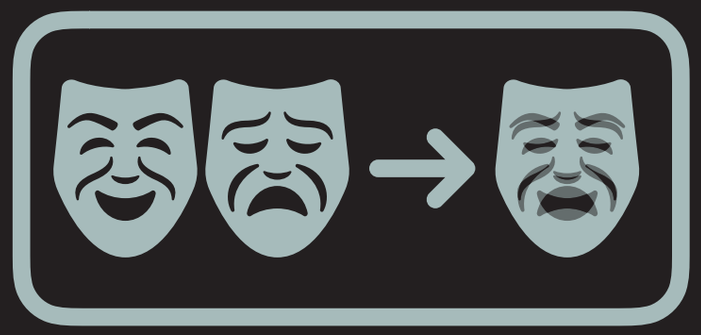




Generation of Ensemble Model

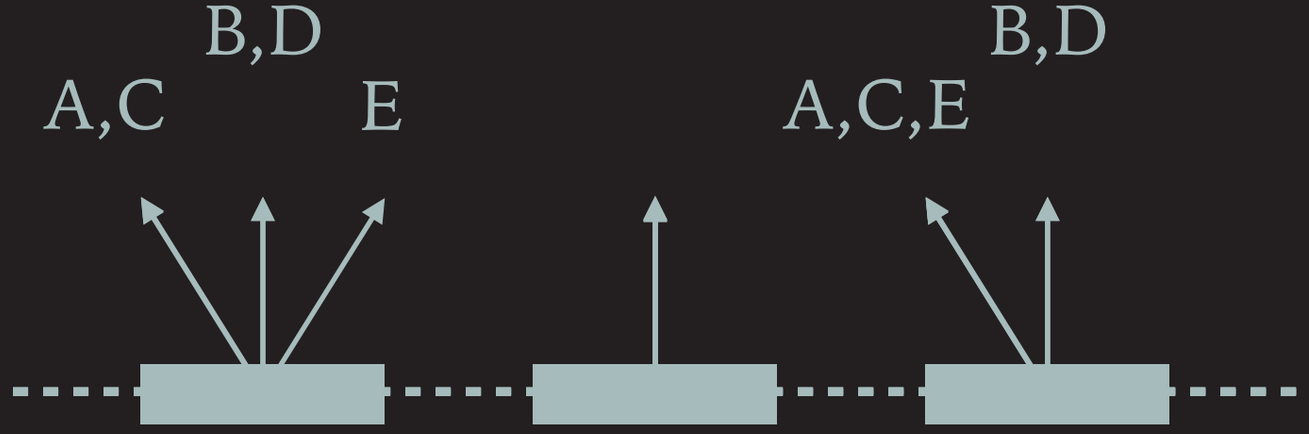
6) Remove duplicated confs. for atoms





Generation of Ensemble Model

6) Remove duplicated confs. for atoms



Final merged model

This is the simplest model that is "crystallographically correct"

(i.e. CAN BE REFINED PROPERLY).

Restraints & Refinement

giant.make_restraints & giant.quick_refine

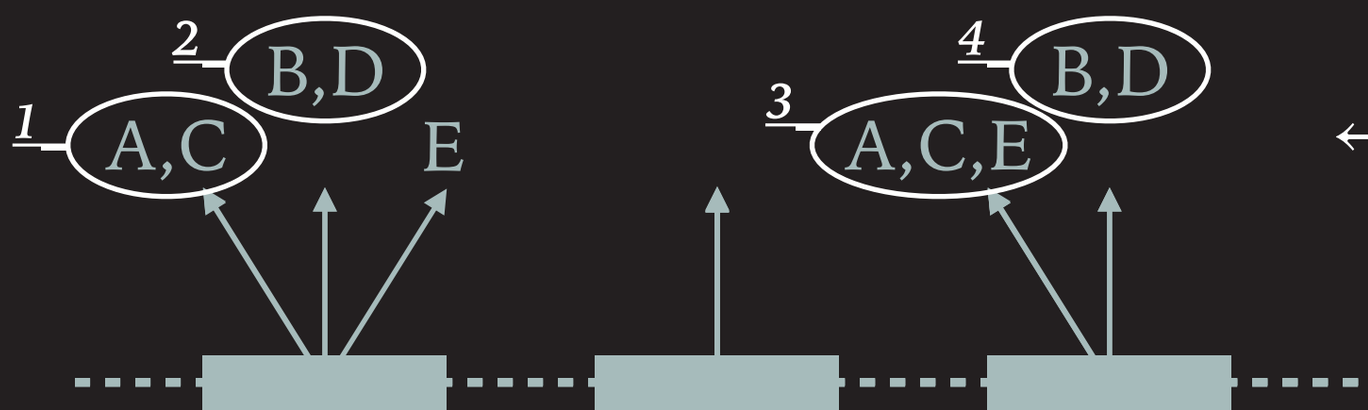
Need to apply external restraints to stabilise refinement	Situation
1) Generate tight restraints for atoms that are duplicated	e.g. duplicated A,B confs.
2) Occupancy groups for all spatially-proximate conformers	nearby conformer “X” occupancies should be equal
3) Generate PROSMART-like distance restraints for selected conformers (i.e. restraints to maintain local structure)	i.e. restrain refinement to input model
Also does checks to ensure continuity of the protein backbone across all conformers (each peptide bond is “crystallographically correct”).	

Restraints & Refinement

giant.make_restraints & giant.quick_refine

1) Duplicated Atoms

- The merged process explicitly duplicates residues, and thus increases the apparent number of model parameters; without additional restraints, this will lead to increased overfitting in refinement.
- To remove the “freedom” of these parameters in refinement, we generate atomic restraints to keep the conformers “the same”.
  - Result: tight zero-distance restraints for duplicated atoms



⟵ *each circled group of conformers moves together in refinement*

Restraints & Refinement

giant.make_restraints & giant.quick_refine

2) Occupancies

- Generate occupancy restraints to stabilise occupancy refinement
  - Using the merged models, can simply look for groups of atoms with the same conformer (atom within x Å of each other).
  - Create groups of residues, and split into groups by conformer
    - local conf. A occ. + local conf. B occ. + … = 1.0

3) Local structure restraints

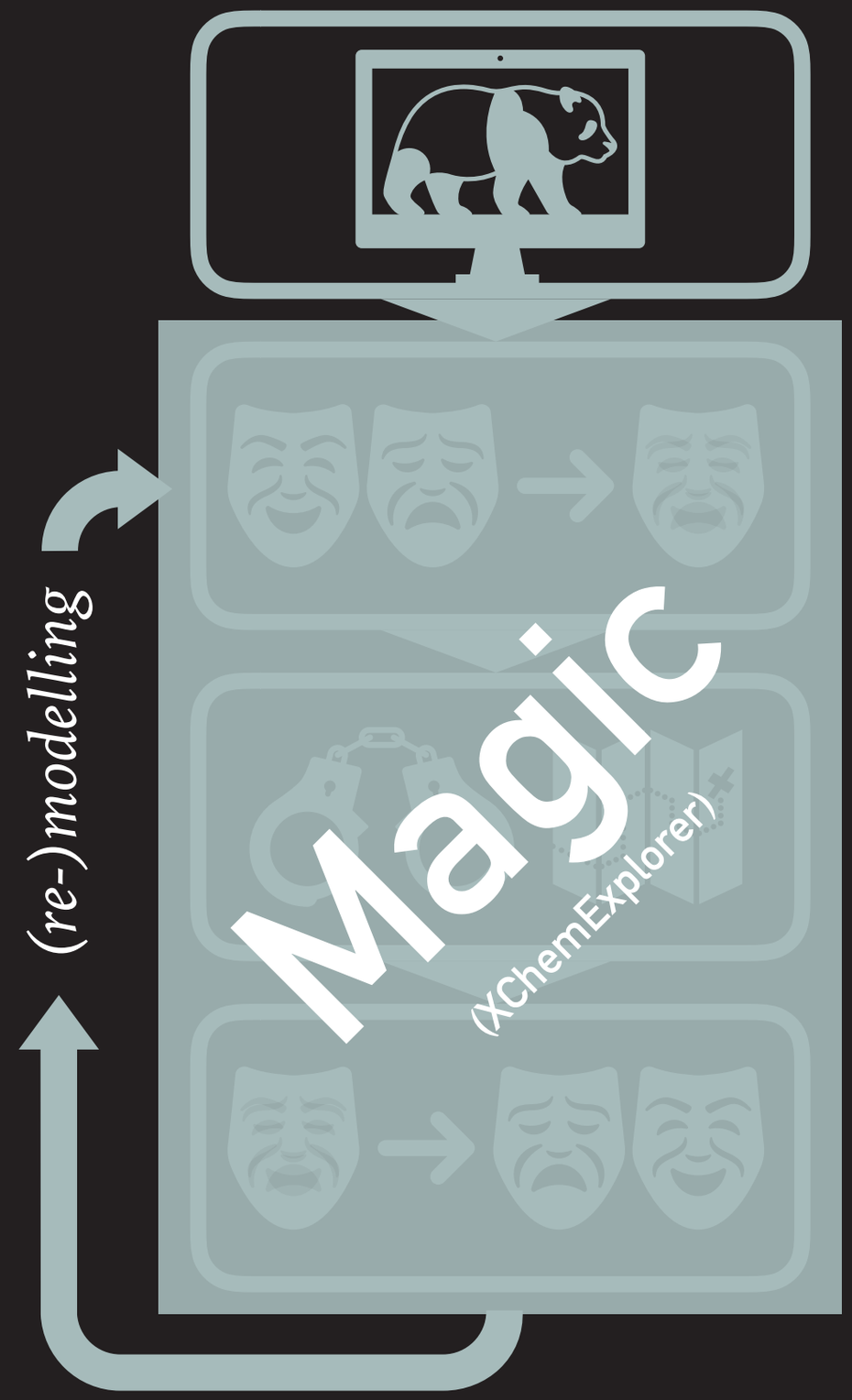
- Also generate a list of “PROSMART” distance restraints to preserve the local structure within each conformer.

Separation of Ensemble States

giant.split_conformations

- Performs the inverse of giant.merge_conformations.
- Can split by:
  - a conformer -> (A), (B), (C), (D), (E)
    - one model for each conformer in the structure
  - a residue -> (A,B,C), (D,E)
    - all conformers with a residue (e.g. LIG) in one model.
    - e.g. ligand-bound states and ligand-unbound states
- Once split, duplicated conformers are removed as in merging step 6).

INTERACTIONS AS A USER



<u>Analysis & Modelling</u>

Model the bound-state <u>only</u> in pandda.inspect

<u>Refinement & Re-modelling</u>

- *The merged models are complicated to work with, thus all modelling should be done on the bound- and the ground-state separately.*
- *Merging, restraint-generation and refinement should be performed automatically (and invisibly) in XCE so that you only interact with the <u>unmerged</u> models.*

<u>The final merged model is deposited in the PDB</u>

SIMPLE USAGE

(run with --show-defaults for all options)

pandda.analyse [options] …

pandda.inspect

giant.merge_conformations model1.pdb model2.pdb

(runs automatically in pandda.export)

giant.make_restraints model.pdb

(can run automatically after giant.merge_conformations)

giant.split_conformations model.pdb

(can run automatically after giant.quick_refine)