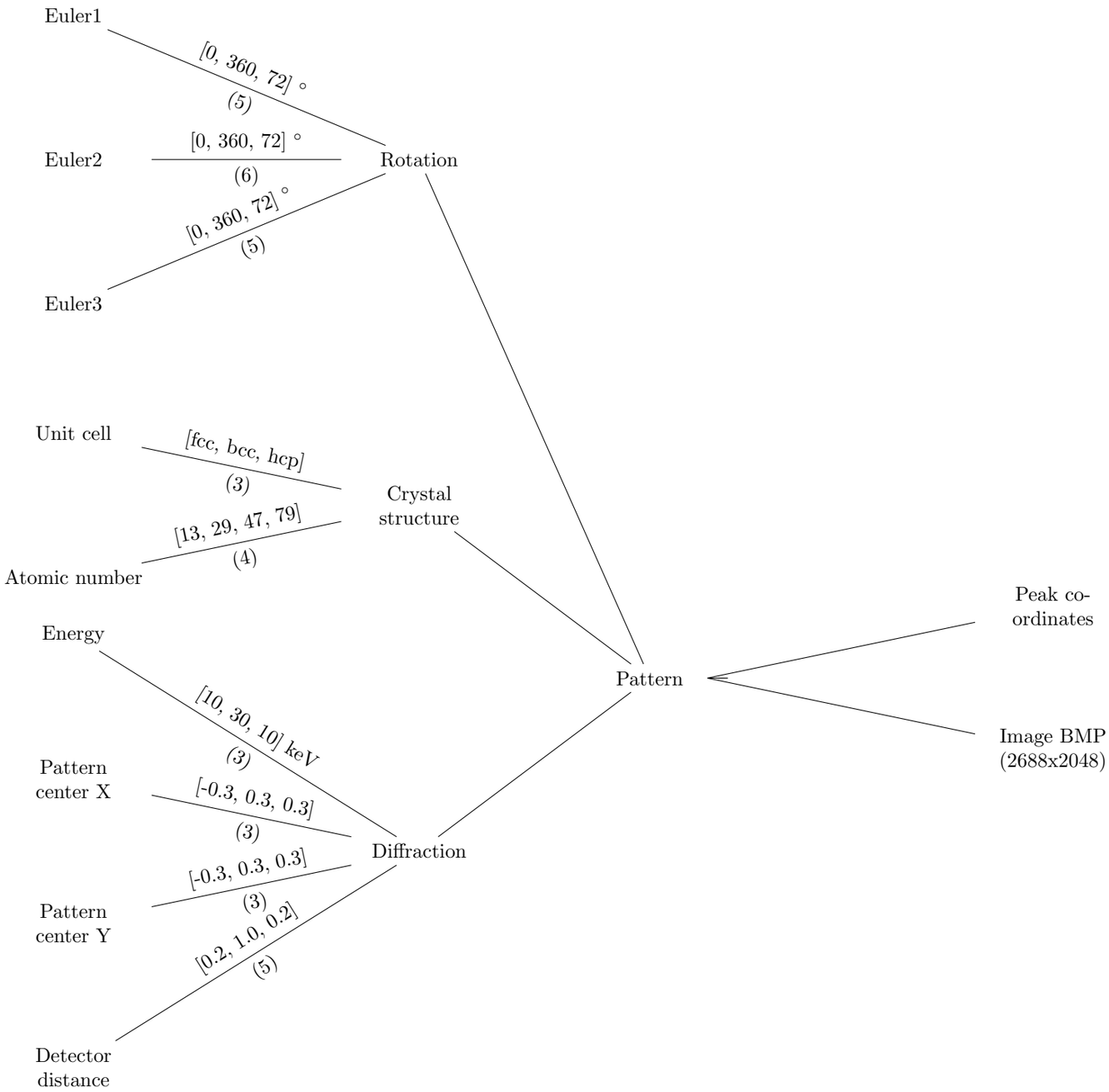
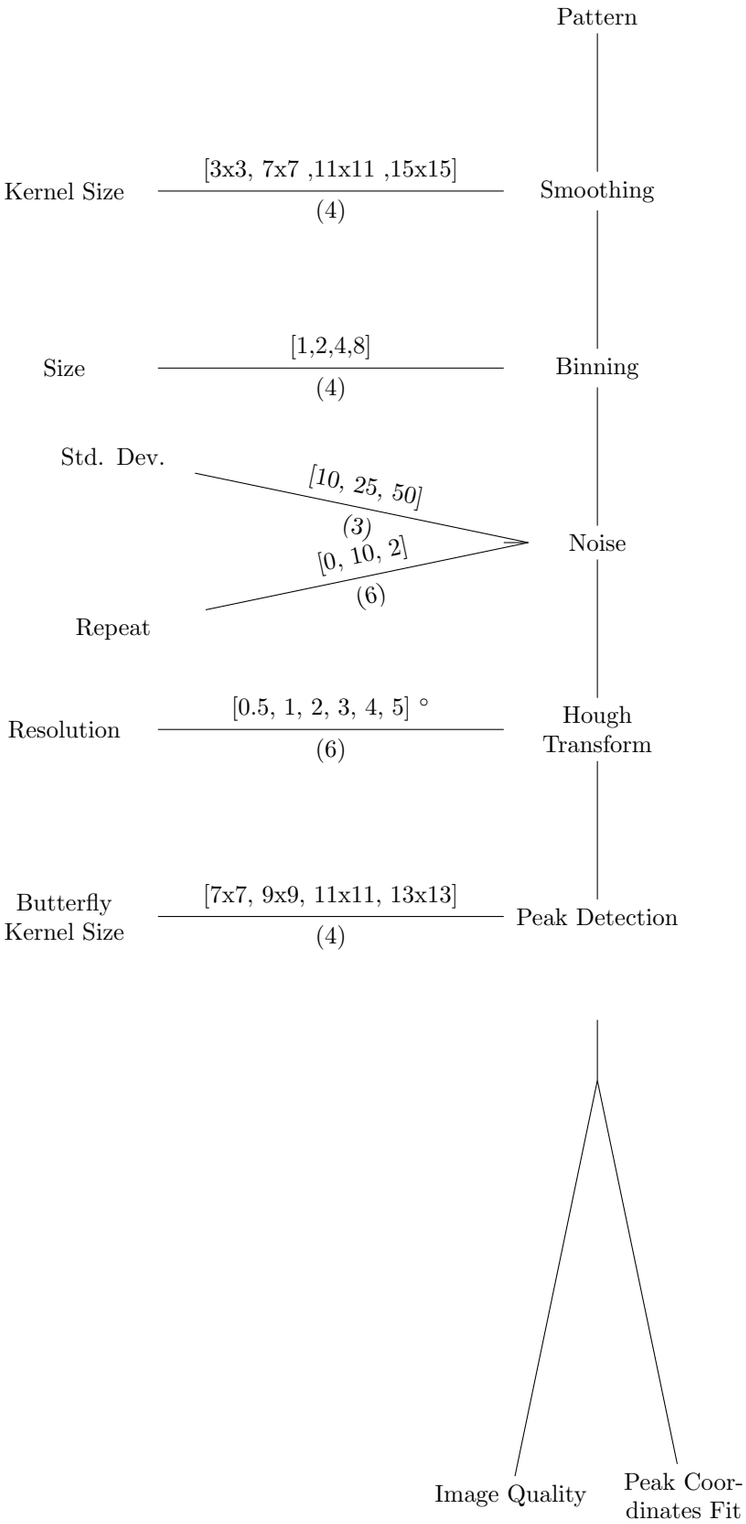
1 Pattern Simulation



Legend:
$[x, y, z] \Rightarrow [start, end, step]$
$(n) \Rightarrow$ number of analyses

2 Detection Simulation



3 Experiments

3.1 Classes

3.1.1 Experiment

- Store all information to run **one** experiment (pattern and detection parameters)
- Load and save information in *.exp* files
  - key=value
  - simulate only pattern (optional)
  - outputs requested (fit, image quality, etc.)
  - database connection information (optional)

3.1.2 Experiments

- Create an `ArrayList` of `Experiment` based on the lists of values
- Use `MultipleLoop` to construct the `ArrayList`
- Load values from constructor or,
- Load values from *.exps* files
  - key=value1, value2

3.1.3 ExperimentsCasir

1. Split experiments based on the number of nods
2. Create list of *.exp* in *.txt* files
3. Launch *qsub* `patterns.java` with the *.txt* files

3.2 Mains

3.2.1 Pattern.java

- Process

  1. Create a `Experiment` class from a *.exp* file or the command line arguments (arguments override *.exp* file)
  2. Create a pattern (see class `Pattern`)
  3. Analyse pattern (based on argument)
  4. Save results in prop file (based on argument)
  5. Save image (based on argument)
  6. Save results in MySQL database (based on argument)

- Arguments

  -f *.exp* filename

  -s save image

  -r save results

  etc.

3.2.2 ExperimentsRun.java

- Process

  1. Load the values contained in a *.exps* to create a `Experiments` class
  2. Create *.exp* for all experiments
  3. Launch each `Experiment` based on the argument (see below)

- Arguments

  -f *.exps* filename

  -c launch on casir (i.e. call `ExperimentsCasir`) with the number of nods desired

  -l launch locally

3.2.3 Patterns.java

- Similar than `Pattern.java` but load a file (*.txt*) containing a list of *.exp*.
- Run each *.exp* like `Pattern.java`

4 Interfaces and inputs