# Atom-Parser documentation

Benjamin Schrauf

September 15, 2017

**Overview**

This python script prepares the input data for the DFTB+ electron transport density functional theory code. The program takes a molecule, given in the form of atom positions, as input. Also provided are the maximum distances at which the atoms interact, and the contact atoms through which electric current is introduced to the molecule. Using this information, the program divides the atoms into groups ("bins"). Each bin's atoms may only be in contact with atoms from at most two other bins. To satisfy this contraint, the program must generate an equivalent one dimensional structure by merging together the appropriate bins and contacts, until only two contacts are left.

# 1 Program description

The DFTB+ code uses tridiagonal Hamiltonian matrices to solve molecular electron transport problems. To generate a tridiagnal Hamiltonian matrix, the atoms need to be sorted into groups ("bins") in such a way that the atoms from one bin only interact with atoms from at most two other bins.

## 1.1 Algorithm

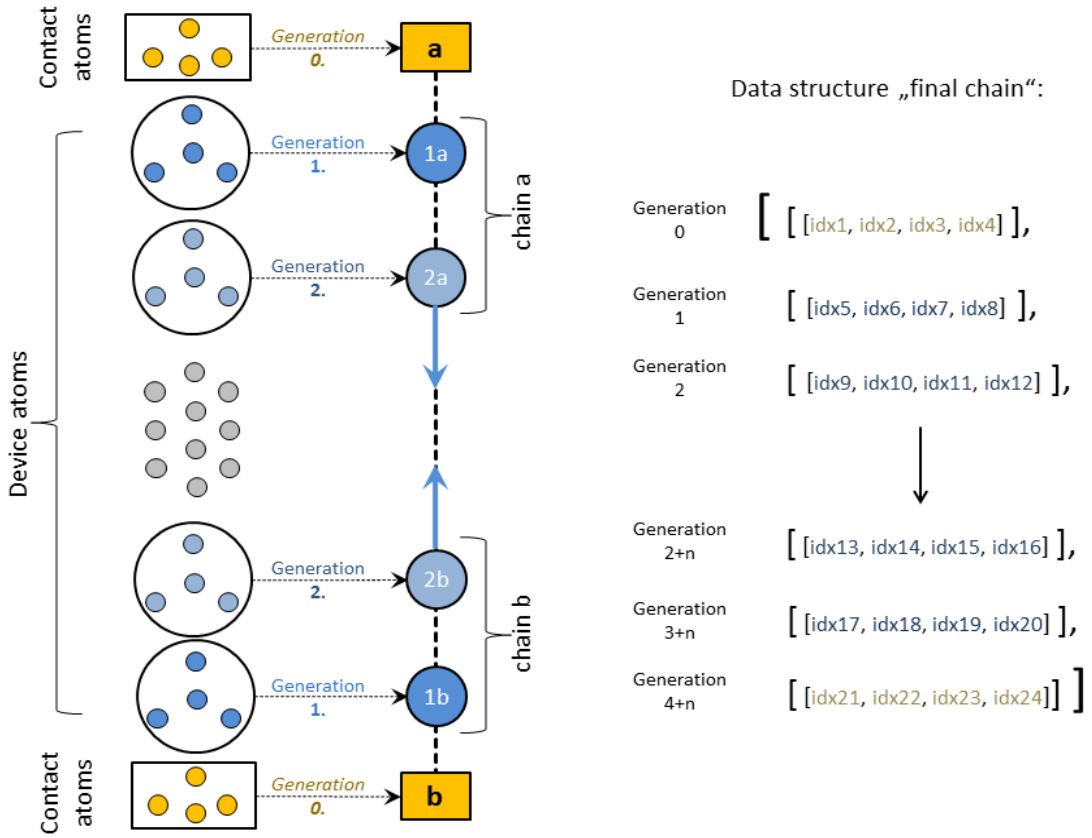
The program assigns as index to each atom, and associates this indes with a position by a list. The program creates a list of atom indices for the atoms in each contact, as well as a list of atoms indices for the device atoms.

### 1.1.1 Data preparation

First the program generates two $n \times n$ -matrices, where $n$ is the number of atoms in the input molecule. The first matrix (dist_mtrx) gives the distances between each atom in the molecule, the second matrix (interact_mtrx) is composed of truth values that indicate whether two atoms are within interaction distance.

### 1.1.2 Sorting atoms into bins

The atoms are sorted into bins starting from from all contacts simultaneously. First, the script creates lists ("contact_bins") containing the atoms in each contact that are interacting with device atoms. These lists are then combined into a nested list, called "generation zero". Next, a nested list containing lists of all device atoms interacting with each contact_bin is generated. This nested list, called "generation one", is then appended to to the the nested list of generation zero. Each sucessive generation of bins is generated analogously, and appended to the previous generation. The data structure thus creates is called the "chain". The creation precess is depicted in figure 1.




1

Figure 1: Generating the chain data structure. Left side: Molecule with colored dots representing atoms. Center: Sorted chains a and b, consisting of bins. Right side: Data structure "final_chain".

### 1.1.3 Merging bins