Part of **Age Concern**, a joint project between **Oxford** and **Durham** Universities

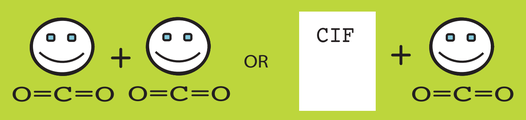
N David Brown | Mustapha Sadki | Amber L Thompson

James Haestier | David J Watkin | *..the Oxford team!*

CHEMICAL STRUCTURE MATCHING - AN IMPROVED ALGORITHM

STEP 1

Input two chemical structures in either CIF or SMILES format.

CIF + CIF OR  OR CIF +

STEP 2

Specify which elements should match with which, and how each atom's environment should correspond to the matched atom's environment for a match to be successful.

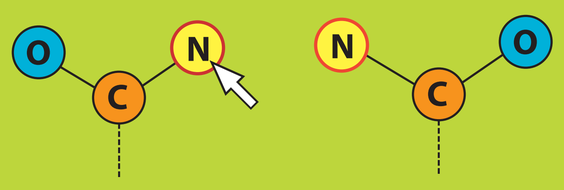
```
global_options=
      [chemical_identical] # atom pairs considered for mapping must be the same element
      [chemical_similar]   # ..these candidates must have chemically similar environments
      [graph_identical]    #..and these environments must match elements present and
                           #  their frequency
```

STEP 3

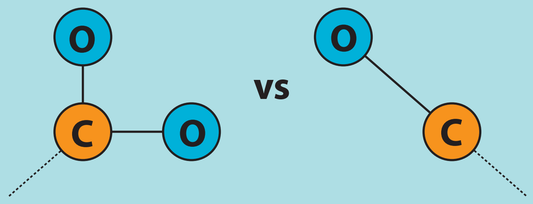
The algorithm begins to explore possible mappings between the two input structures, where each mapping is simply a set of atom pairs, and each atom pair consists of one atom from each input structure. We expand each mapping as much as possible, exhausting one search before beginning another (commonly known as depth first search).

STEP 4

Receive results rapidly from text report displayed upon algorithm completion. View mappings in human-readable text form, listed in descending order of size. Even better, display and cycle through mappings in a bespoke 3D renderer, and mouse over an atom in one structure to highlight its mapped counterpart in the other structure.

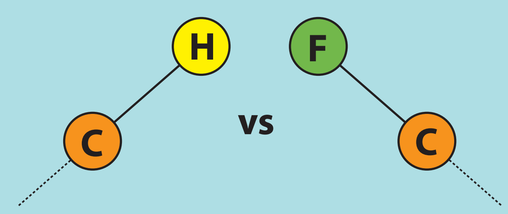


Graph Similarity



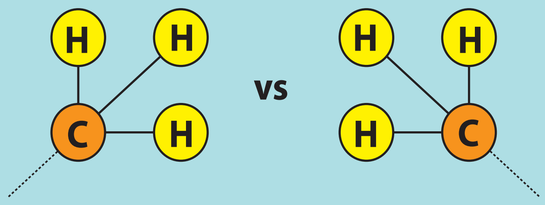
Our algorithm builds on an expansion *1 of the classic Ullmann algorithm*2. By the nature of Ullmann's method, connectivity is maintained during structure matching, i.e. there must be the same bonding configuration between the matched atoms in one structure and their counterpart mapped atoms in the other structure. Additionally, we allow tailoring of this graph comparison to successfully match only 'graph identical' environments (i.e. same number of bonds), or 'graph similar' environments where the definition of similarity is arbitrary to the implementation.

Chemical Similarity



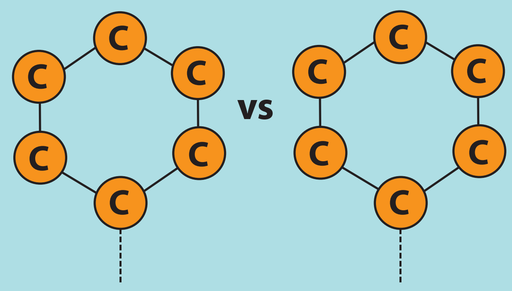
We can tailor which elements will match with which in a similar manner to tailoring our graph matching behaviour. By specifying that only 'chemical identical' candidates and environments should be considered, we limit mapping candidates to those atom pairs where both atoms - one from each input structure - have the same element type. Alternatively, specifying 'chemical similar' for candidates and environments, along with text-described sets of 'similar' environments, allows us to consider equivalent those environments with similar pharmacological effects, for example.

Arbitrary Mappings



Our algorithm purposefully does not consider any spatial information during its search, and instead relies solely on connectivity to match components of our two structures. One consequence of this is that entirely equivalent matches will be generated between the environments of two matched atoms, if one environment contains more than one atom which is chemically identical or similar to the atoms in the first environment. We handle these so-called 'arbitrary mappings' by clustering the arbitrary atoms in each structure, and associating a cluster in one structure with the relevant cluster in the other; we then know that all mappings where each atom in one cluster maps to one of the atoms in the other cluster are valid.

Feature Detection *currently under development....*



Certain moieties which exhibit high local symmetry may prolong chemical structure matching, since many mapping permutations may be possible between two such equivalent moieties in the input structures. One related problem is that of 'ring shuffling' - a linear carbon chain in one structure will repeatedly match with a carbon ring (e.g. benzene) in the other structure, shuffling round the ring by one atom to generate each new match. Our implementation is currently able to rapidly detect all rings present in the input structures. We are putting the final touches to code which detects features described by the user (including rings), and allows more efficient structure matching by avoiding complications such as ring shuffling.

Procedure Flowchart

References:

1. **L.P. Cordella, P. Foggia, C. Sansone, M. Vento (2004). IEEE Trans. On Patt. Anal. & Mach. Intell., 26, 10, 1367-1372**
2. **J.R. Ullmann (1976). Journal of the Association for Computing Machinery, 23, 31-42 .**