

Indigo: Universal Cheminformatics API

Capabilities

- **Support of popular data formats:** SMILES, SMARTS, Molfile, Rxnfile, SDF, RDF, GZip
- **Portability over modern platforms and languages:** Linux/Windows/Mac OS X, 32/64 bit, Java/Python/C#
- **Outstanding performance:** Original algorithms, fast C++ implementation

Functionality

Calculation of structure properties:

Canonical (isomeric) SMILES, molecular weight, molecular formula

Rendering of molecules and reactions:

PNG, SVG, PDF formats supported. Query features supported.

Also automatic SMILES layout, colors, highlighted fragments, titles, ...

Molecule and reaction search:

Exact/Substructure/SMARTS matching. Options for matching tautomers, resonance structures, Markush structures. Original fingerprints. Various similarity measures.

Scaffold detection and R-Group decomposition:

Maximum common substructure of arbitrary amount of input structures

Reaction atom-to-atom mapping:

Calculate a new AAM or alter an existing AAM

Combinatorial chemistry:

Enumeration of reaction products from available monomers.
Support of query features, perception of stereochemistry transformation.
Intramolecular reactions. Automatic screening of duplicates.

Design

```

// C++
auto getU(context fp, parameters.fingerprintSize())
    const auto u = context.fp.getU(fingerprintSize());
// by using the provided Sim()
auto sim_size = context.fp.parameters.fingerprintSizeSim();

// plain C
double freeIter;
// decomposing the structures...
deco = indexDecomposeMolecules(scaffold, structures);
// (optional)
if (outfile)
    int output = writeDecomposition(outfile, deco);
int writeDecomposition(outfile, deco);

```

```
OS_DEF(Molecule, mol,
OS_DEF(Array<int>, gross);

MoleculeAutoLoader loader(moffile);
loader.load(Molecule);

MoleculeAromatizer::aromatizeBonds(mol);
MangoExact::calculateHash(mol, _hash);

MoleculeFingerprintBuilder builder(mol, _context.fp_parameters
builder.process();

fp_size(builder.get(), _context.fp_parameters.fingerprintSize());
fp_size(_count = builder.count(), _context.fp_parameters.fingerprintSize());
outputSize(_count, _context.fp_parameters.fingerprintSize());

const int fp_size = (context.fp_parameters.fingerprintSizeSim());
int fp_size_sim = (context.fp_parameters.fingerprintSizeSim());

// Save gross formula to file
GrossFormula::calc(mol, _context);
GrossFormula::write(mol, _context,
_counted_elems, _clear);

ArrayOutput ce_output(_counted_elems_str);

for (int i = 0; i < int(NEMLEM(counted_elements)); i++)
ce_output.printf("%s", gross[counted_elements][i]);

ce_output.writeByte(0);

// Calculate molecular mass
MolecularMass mass_calculator;
mass_calculator.relative_atomic_mass_map;
context.relative_atomic_mass_map;
_molecular_mass = mass_calculator.calculateMolecularMass(mol,
context.relative_atomic_mass_map);

// Calculate molecular mass
MolecularMass mass_calculator;
mass_calculator.relative_atomic_mass_map;
context.relative_atomic_mass_map;
_molecular_mass = mass_calculator.calculateMolecularMass(mol,
context.relative_atomic_mass_map);
```

```
int outfile = IndigoWriteFile(outfile_alfscaffs);
int aliases = IndigoAlScaffolds(scaffoid);
int item, iter = indogenerateArray(alfscaffs);

printf("saving all obtained scaffolds (%d total) to %s\n",
    indogenerateArrayCount(alfscaffs), outfile_alfscaffs);

while ((item = indogoNext(iter)))
{
    indogoScaffoldAppend(output, item);
    indogFree(item);
}

indogoFree(iter);

Simplified A
printf("decomposing the structures...");
deco = indogoDecomposeMolecules(scaffoid, structures);

if (outfile)
{
    (plain C)
    int outfile, iter = 0;
    int item, iter = indogenerateDecomposedMolecules(deco);

    printf("saving the highlighted structures to %s\n",
        outfile.hi);

    while ((item = indogoNext(iter)))
    {
        indogoScaffoldAppend(output,
            indogoDecomposedMoleculeHighlighted(item));
        indogFree(item);
    }

    indogoFree(iter);
    indogFree(output);
}
```

Java wrapper

C# wrapper

Python

```

arr.arrayAdd(item);

scaf = indigo.extractCommonScaffold(arr, "exact")
deco = indigo.decompose(scaf)
scaf = deco.compose(deco.get("scaf"))
for item in deco.iterateDecomposedMolecules():
    print item.decomposedMoleculesHighlighted().smiles

```

```

    if (obj) {
        rp.mol.reset(new QueryMolecule());
    } else {
        rp.mol.reset(new Molecule());
    }
    rp.mol->clone(self.get<Object>());
}

```

Java wrapper

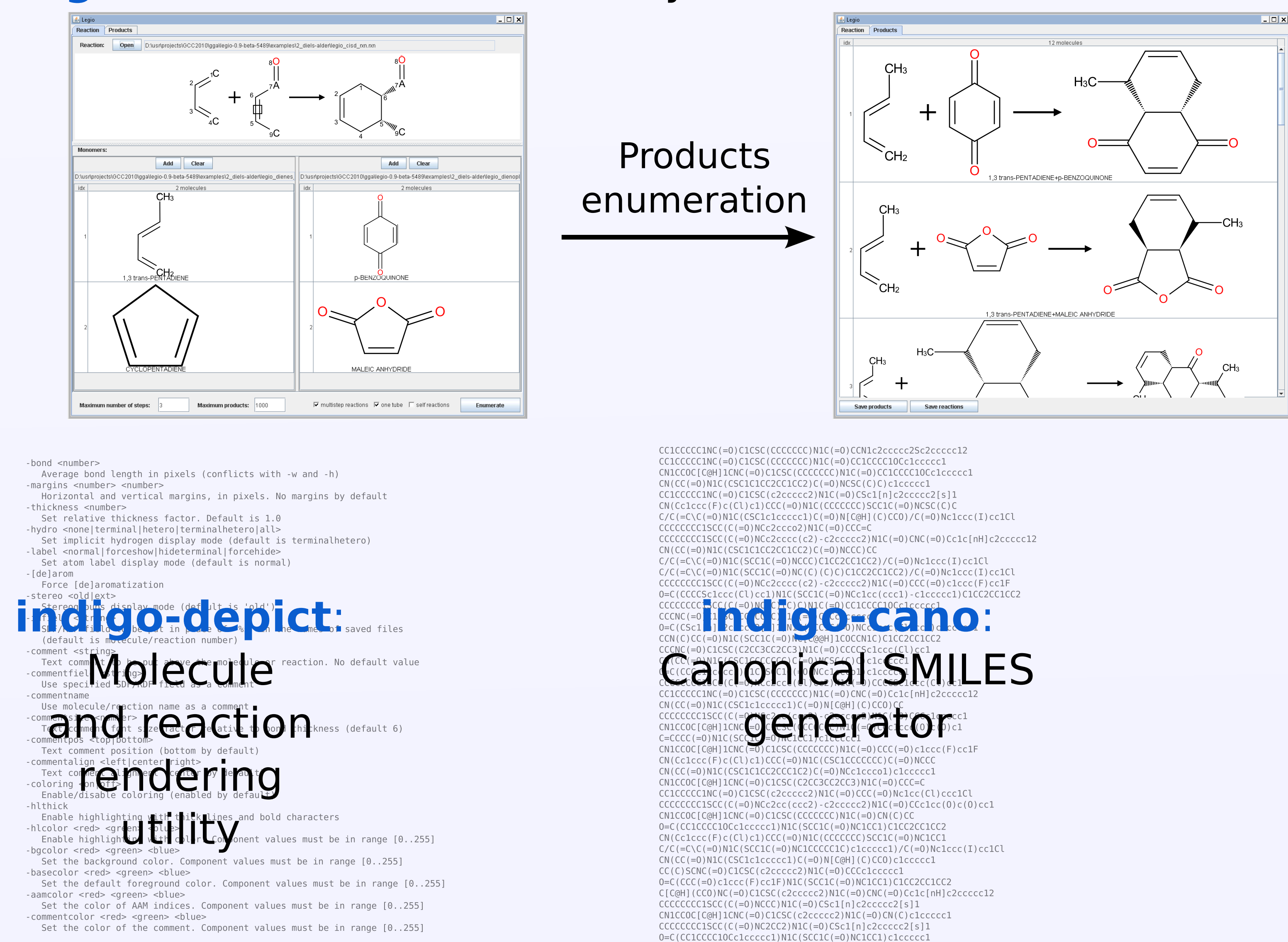
C# wrapper

Python

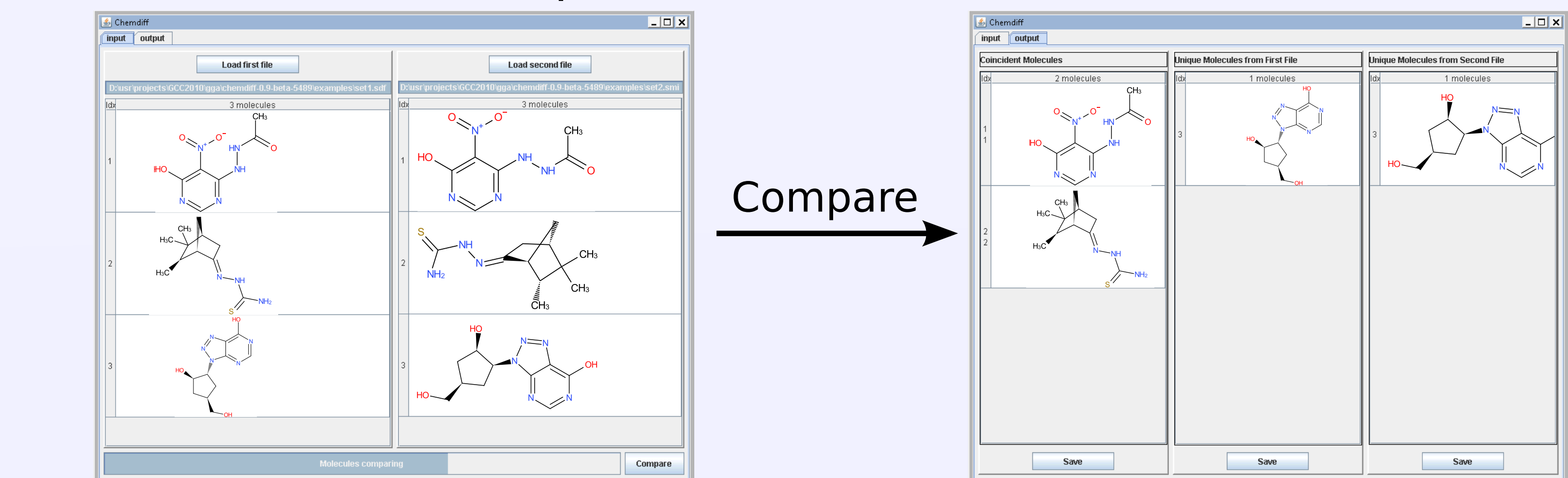
Open for third-party plugins
(either C, C++, Java, C#, or Python)

Applications

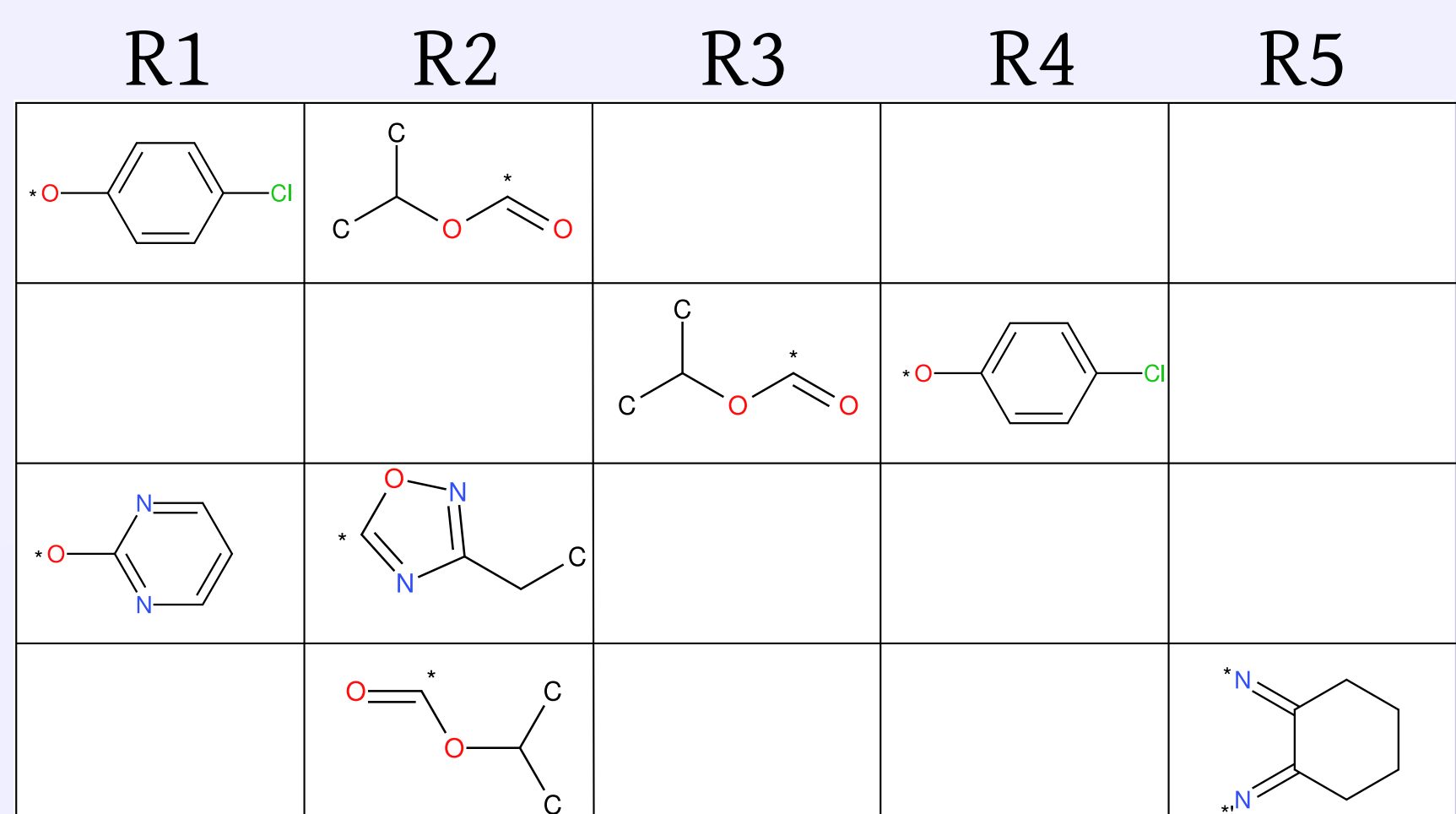
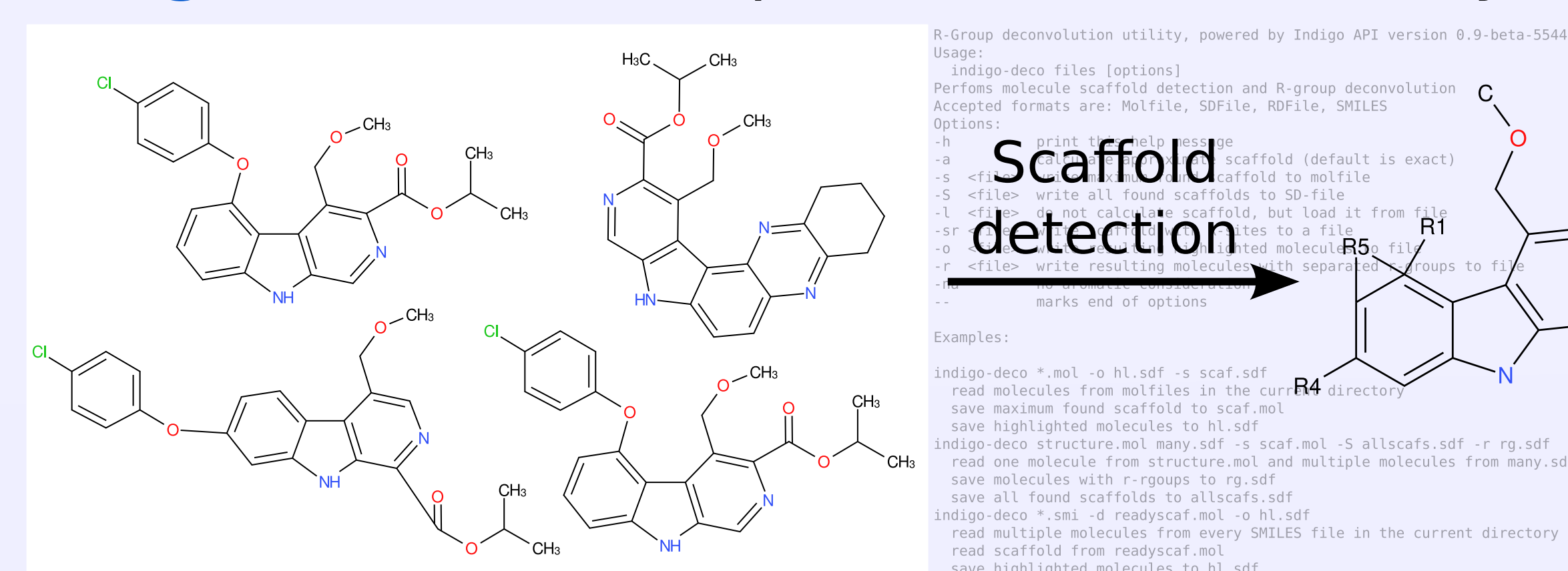
Legio: Combinatorial chemistry GUI tool



chemdiff: Visual comparison of two SDF or SMILES files



indigo-deco: R-Group deconvolution utility



Community

[illegible]

http://scitouch.net
Product information,
documentation, downloads

<http://groups.google.com/group/indigo-general>
<http://groups.google.com/group/indigo-dev>
<http://groups.google.com/group/indigo-bugs>
 Public discussion lists

[illegible]

<http://github.com/ggasoftware>
Complete source code
(GPL v3)

[illegible]

<http://ctr.wikia.com>
Code examples
for different toolkits

<http://blueobelisk.shapado.com>
BlueObelisk question board