Reaction Perception and Protein Handling using RDKit

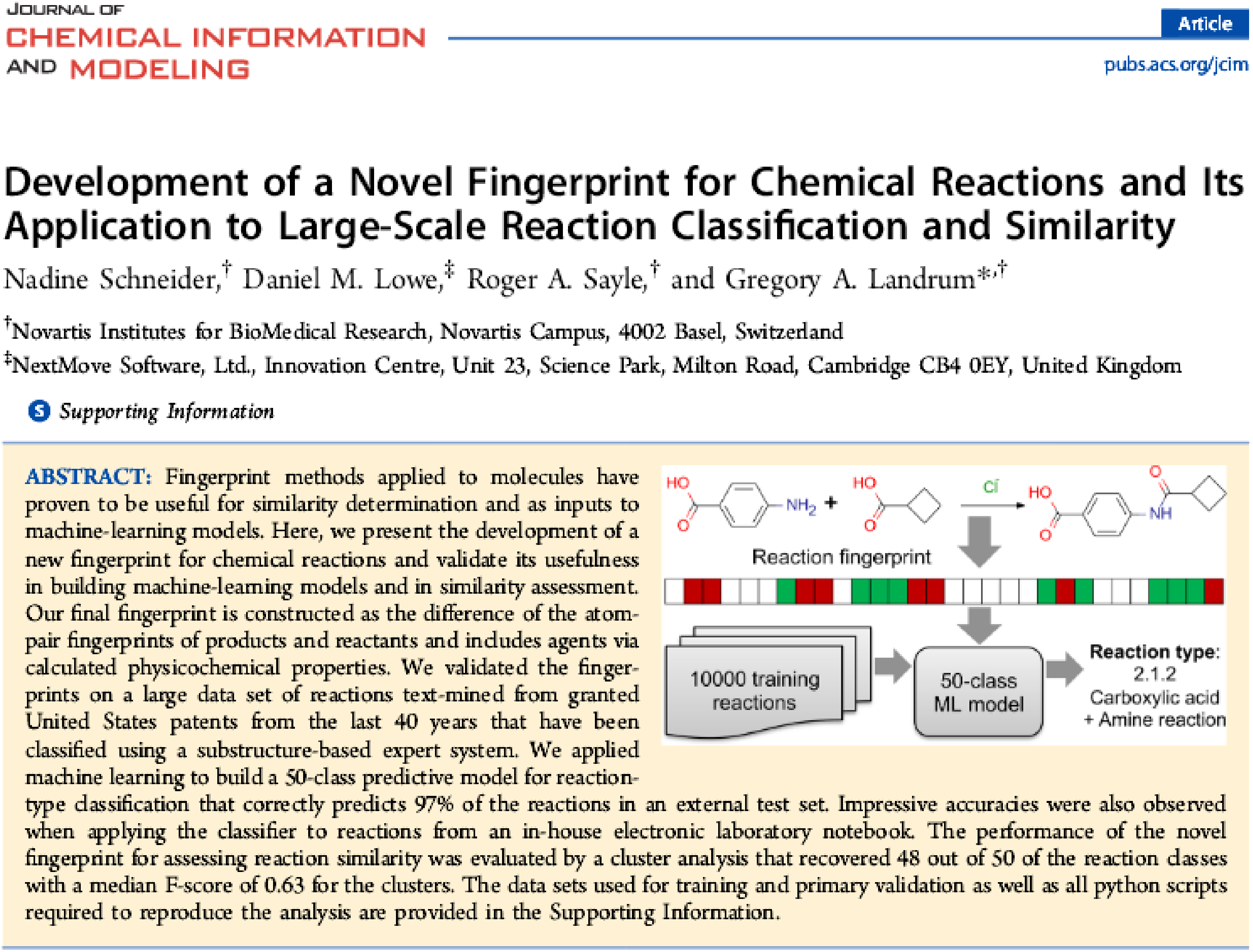
Roger Sayle

NextMove Software, Cambridge, UK

4th RDKit User Group Meeting, ETH, Zurich, Switzerland, Wednesday 2nd September 2015

QUICK OVERVIEW

- Reaction Perception
  - SMARTS pre-compilation
  - Hashing
  - Improved Canonicalization
- Protein Handling
  - Algorithm scaling/complexity
  - (PDB) Residue Representation
  - FASTA sequences
  - Pistoia HELM support



JOURNAL OF
CHEMICAL INFORMATION
AND MODELING

Article

pubs.acs.org/jcim

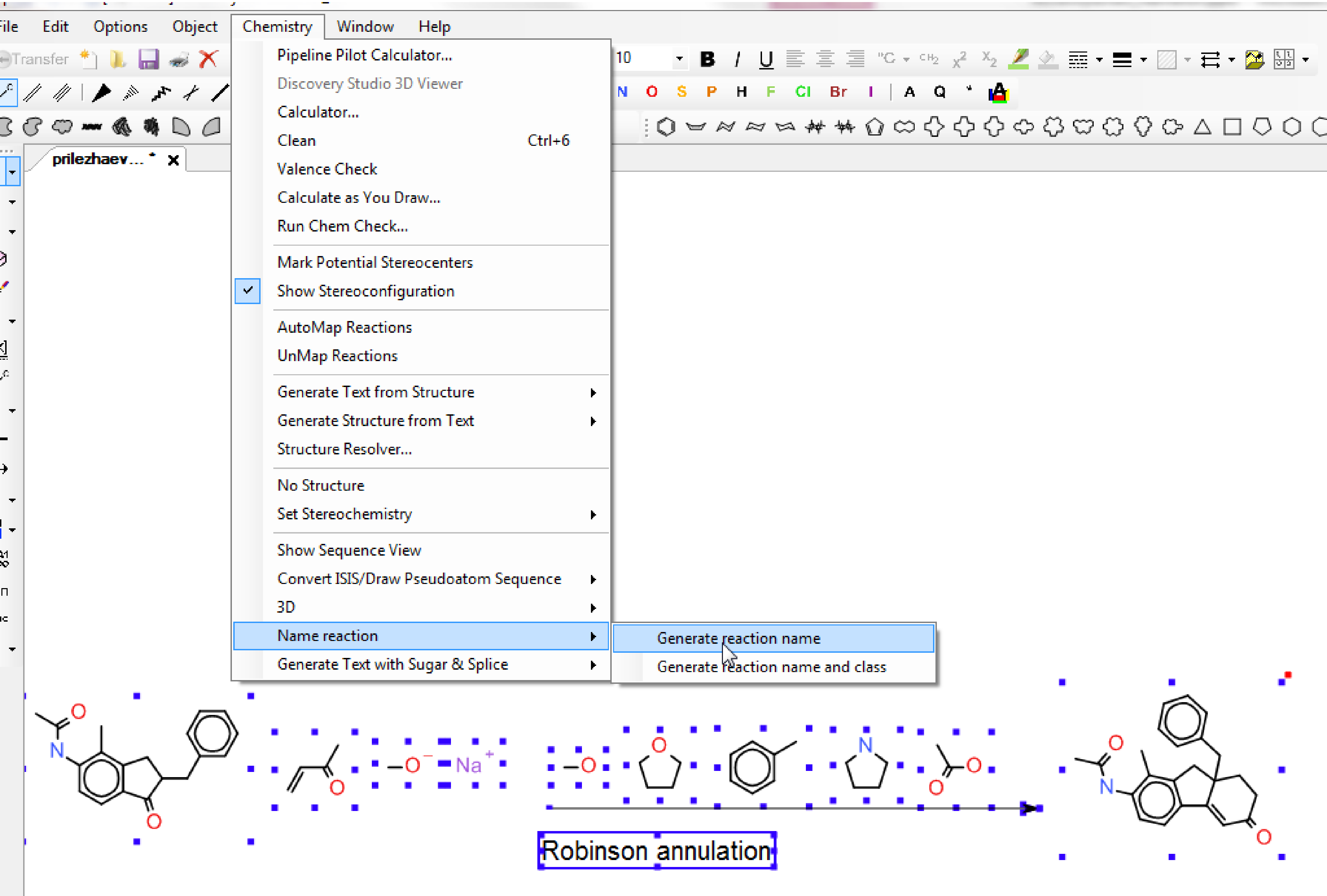
Development of a Novel Fingerprint for Chemical Reactions and Its Application to Large-Scale Reaction Classification and Similarity

Nadine Schneider,[†] Daniel M. Lowe,[‡] Roger A. Sayle,[†] and Gregory A. Landrum*,[†]

[†]Novartis Institutes for BioMedical Research, Novartis Campus, 4002 Basel, Switzerland
[‡]NextMove Software, Ltd., Innovation Centre, Unit 23, Science Park, Milton Road, Cambridge CB4 0EY, United Kingdom

Supporting Information

ABSTRACT: Fingerprint methods applied to molecules have proven to be useful for similarity determination and as inputs to machine-learning models. Here, we present the development of a new fingerprint for chemical reactions and validate its usefulness in building machine-learning models and in similarity assessment. Our final fingerprint is constructed as the difference of the atom-pair fingerprints of products and reactants and includes agents via calculated physicochemical properties. We validated the fingerprints on a large data set of reactions text-mined from granted United States patents from the last 40 years that have been classified using a substructure-based expert system. We applied machine learning to build a 50-class predictive model for reaction-type classification that correctly predicts 97% of the reactions in an external test set. Impressive accuracies were also observed when applying the classifier to reactions from an in-house electronic laboratory notebook. The performance of the novel fingerprint for assessing reaction similarity was evaluated by a cluster analysis that recovered 48 out of 50 of the reaction classes with a median F-score of 0.63 for the clusters. The data sets used for training and primary validation as well as all python scripts required to reproduce the analysis are provided in the Supporting Information.



4th RDKit User Group Meeting, ETH, Zurich, Switzerland, Wednesday 2nd September 2015

NAMERXN REACTION NAMING

- Many reaction classification algorithms are dependent upon atom-atom mapping assignments.
- Alas MCS-based atom mapping algorithms are often slow and/or inaccurate [Lowe & Sayle 2012 & 2013].
- NameRXN is a mechanism-based atom-mapper.
- All reactants and reagents are placed in a single pot (molecule) and sets of SMIRKS applied in turn.
- If the desired product is generated, the reaction (its mechanism and mapping) is identified.
  - Rationalization is easier than prediction!

REACTION ONTOLOGY

- Reactions are classified into a common subset of the Carey et al. classes and the RSC's RXNO ontology.
- There are 12 super-classes
  - e.g. 3 C-C bond formation (RXNO:0000002).
- These contain 84 class/categories.
  - e.g. 3.5 Pd-catalyzed C-C bond formation (RXNO:0000316)
- These contain ~300 named reactions/types.
  - e.g. 3.5.3 Negishi coupling (RXNO:0000088)
- These require ~800 SMIRKS-like transformations.

EXAMPLE SMARTS/SMIRKS

```
# NOZAKI_HIYAMA_KISHI_REACTION
[#6v4+0;X4,X3:1][BrD1h0+0:2].[Ni].[Cr].[OD1h
0+0:3]=[CD2h1v4+0:4]>>[#6:1][C:4]-[Oh1:3]

# PAAL_KNORR_THIOPHENE_SYNTHESIS
[OD1h0+0:1]=[CX3v4+0:2][CX4v4+0:3]([H])[CX4v
4+0:4]([H])[CX3v4+0:5]=[OD1h0+0:6]>>[S:1]1[C
:2]=[C:3][C:4]=[C:5]1
```

- Writing SMIRKS is both an art and a science.

SMARTS PATTERN COMPILATION

```
bool atom_1(const RDKit::Atom *aptr) {
  return aptr->getAtomicNum() == 6 &&
         ringinfo->numAtomRings(aptr->getIdx()) != 0 &&
         aptr->getDegree() == 3 &&
         aptr->getTotalNumHs(false) == 0 &&
         aptr->getExplicitValence()+aptr->getImplicitValence() == 4 &&
         aptr->getFormalCharge() == 0;
}

bool atom_28(const RDKit::Atom *aptr) {
  if (aptr->getAtomicNum() != 6 ||
      aptr->getExplicitValence()+aptr->getImplicitValence() != 4 ||
      aptr->getFormalCharge() != 0)
    return false;
  return (aptr->getDegree() == 2 && aptr->getTotalNumHs(false) == 1) ||
         (aptr->getDegree() == 2 && aptr->getTotalNumHs(false) == 2) ||
         aptr->getDegree() == 3;
}
```

SMARTS PATTERN COMPILATION

```
  RDKit::ROMol::OBOND_ITER_PAIR biter[25];
…
      case_7:
        biter[0] = mol->getAtomBonds(atom[1]);
        goto case_9;
      case_8:
        avisit[atom[0]->getIdx()] = 0;
        ++biter[0].first;
      case_9:
        if (biter[0].first != biter[0].second) {
          bptr = (*mol)[*biter[0].first].get();
          if (bond_1(bptr)) {
            aptr = bptr->getOtherAtom(atom[1]);
            aidx = aptr->getIdx();
            if (avisit[aidx] == 0 && atom_1(aptr)) {
              avisit[aidx] = 1;
              atom[0] = aptr;
              goto case_10;
            } else ++biter[0].first;
          } else ++biter[0].first;
        } else goto case_5;
        goto case_9;
```

Recent Improvements Insights

- Profiling NameRxn in 2014 to diagnose performance problems with RDKit revealed that pattern matching and transformation accounted for <1% of runtime.
- The bottleneck is actually in canonicalization.
- The Ah-ha experience was to use hash filtering.
- Check molecular formula: $C_iH_jBr_kCl_lF_mI_nN_oO_pP_qS_r$
- Additional cleverness allows pre-sanitization hashing.
  - Triple bond count, but not single or double bond count.
  - Perhaps there's something in InChI-style hashing after all.

CHEMINFORMATICS INSIGHT #5

- There are two ways of testing for molecule equality:
  - Generate canonical SMILES for each and compare strings.
  - Test whether each is a substructure of the other.
- In both approaches, hash filtering (such as atom and bond counts) can be used.
- Which is most appropriate depends upon the relative performance of canonicalization to pattern matching for a particular toolkit.
- Use of both approaches in a single toolkit (such as RDKit usually indicates an inefficiency).

WHILST I'M PREACHING . . .

- Beware of APIs such as:

```
RWMol *SmilesToMol(std::string smi, …)
```

- Many programmers fail to realize that the above idiom requires a memory allocation, copy, and deallocation on every single innvocation.
- In threaded environments, these may be serializing.
- A better equivalent idiom:

```
RWMol *SmilesToMol(const std::string &smi, …)
```

- A better idiom still (think string literals) is the use of:

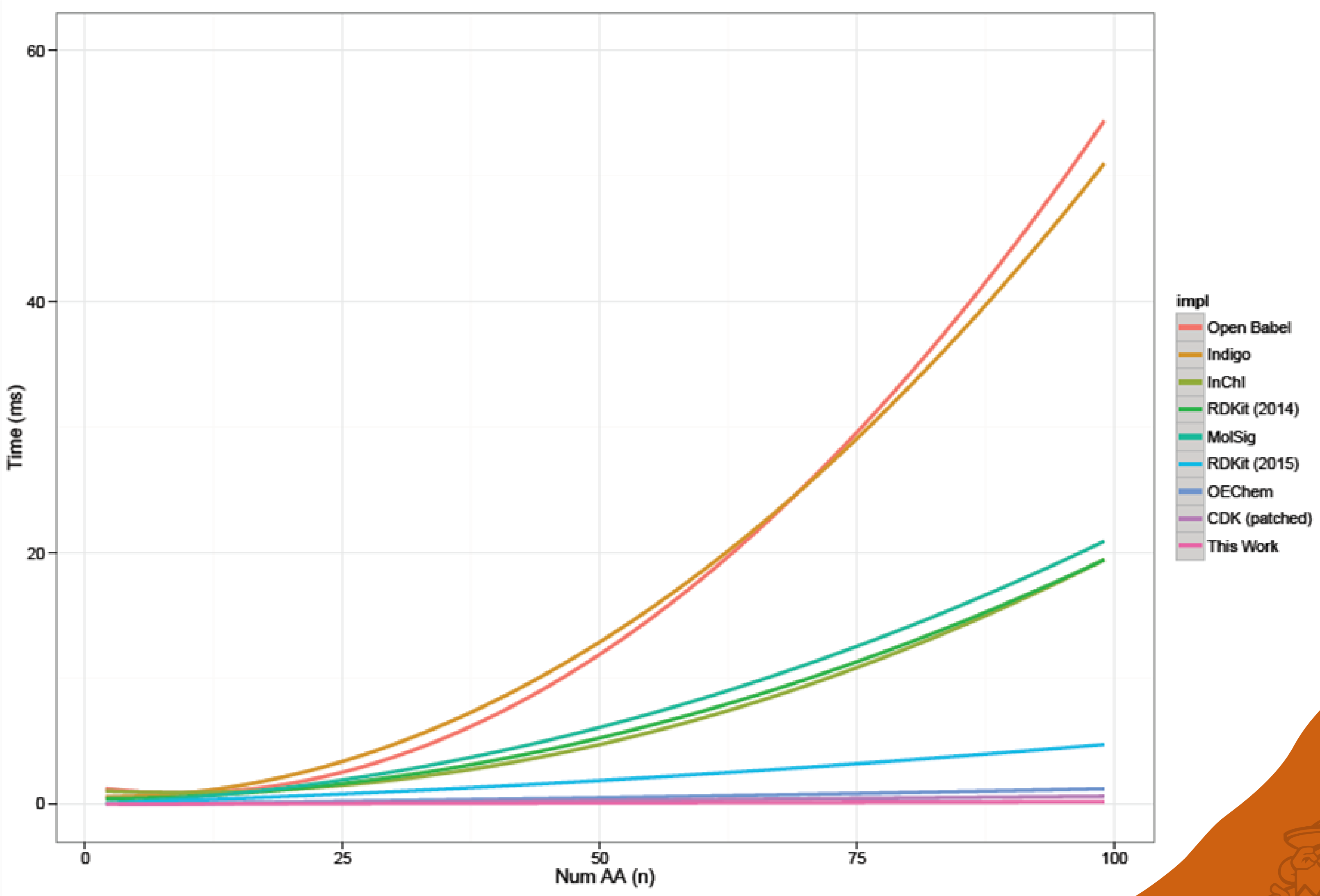
```
RWMol *SmilesToMol(const char *smi, …)
```

- Finally, internally convert C++-to-C, not C-to-C++.

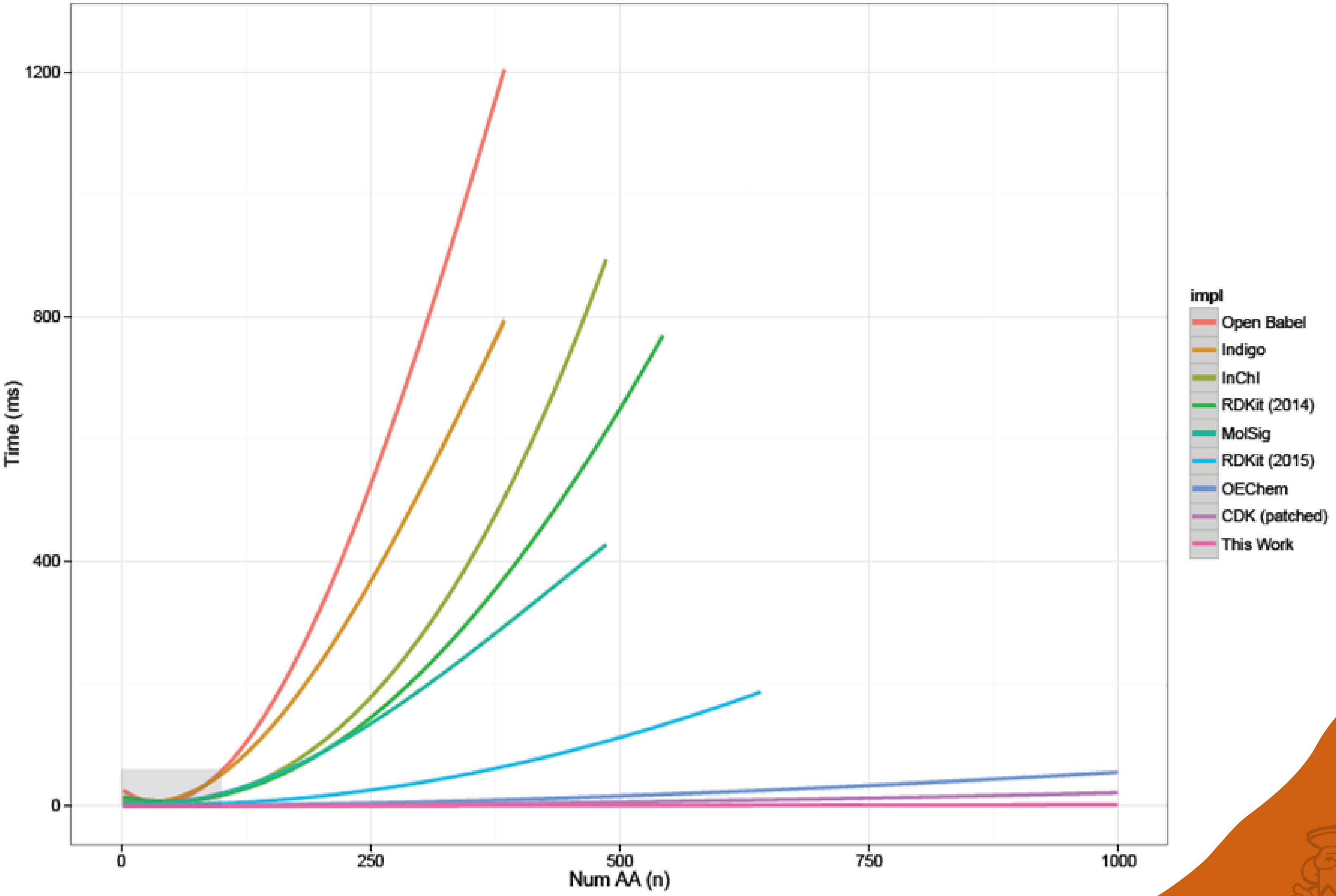
CANONICALIZATION IMPROVEMENTS

- As Nadine will present tomorrow, there have also been dramatic improvements (and bug fixes) in SMILES canonicalization performance between last year’s RDKit UGM and now.
- Amongst the records that get set straight are the use of primes in molecular canonicalization algorithms.
- A manuscript describing these improvements has been submitted to JCIM and fingers-crossed will appear in print later this year.

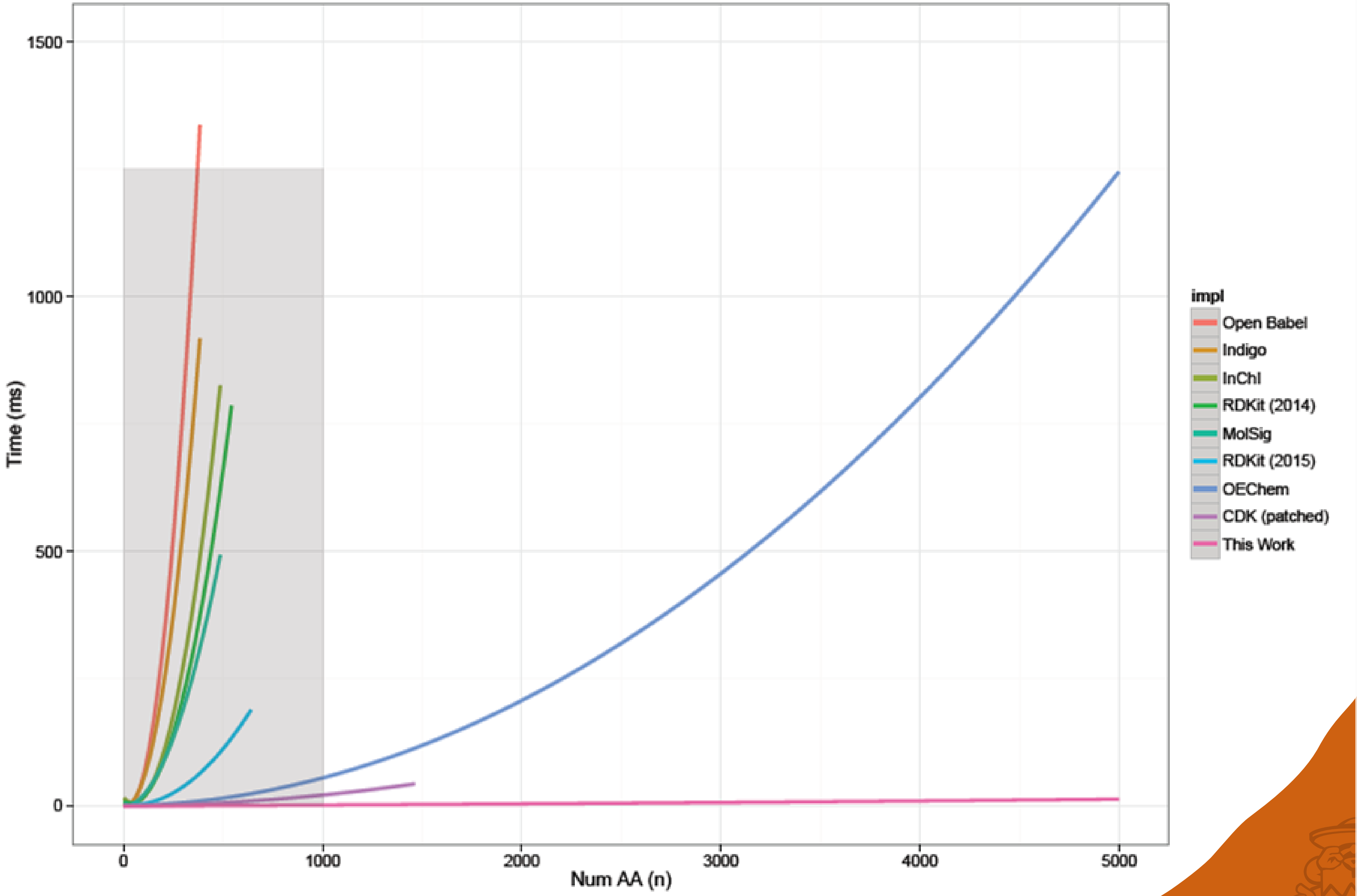
ALGORITHM SCALING TO 100AA



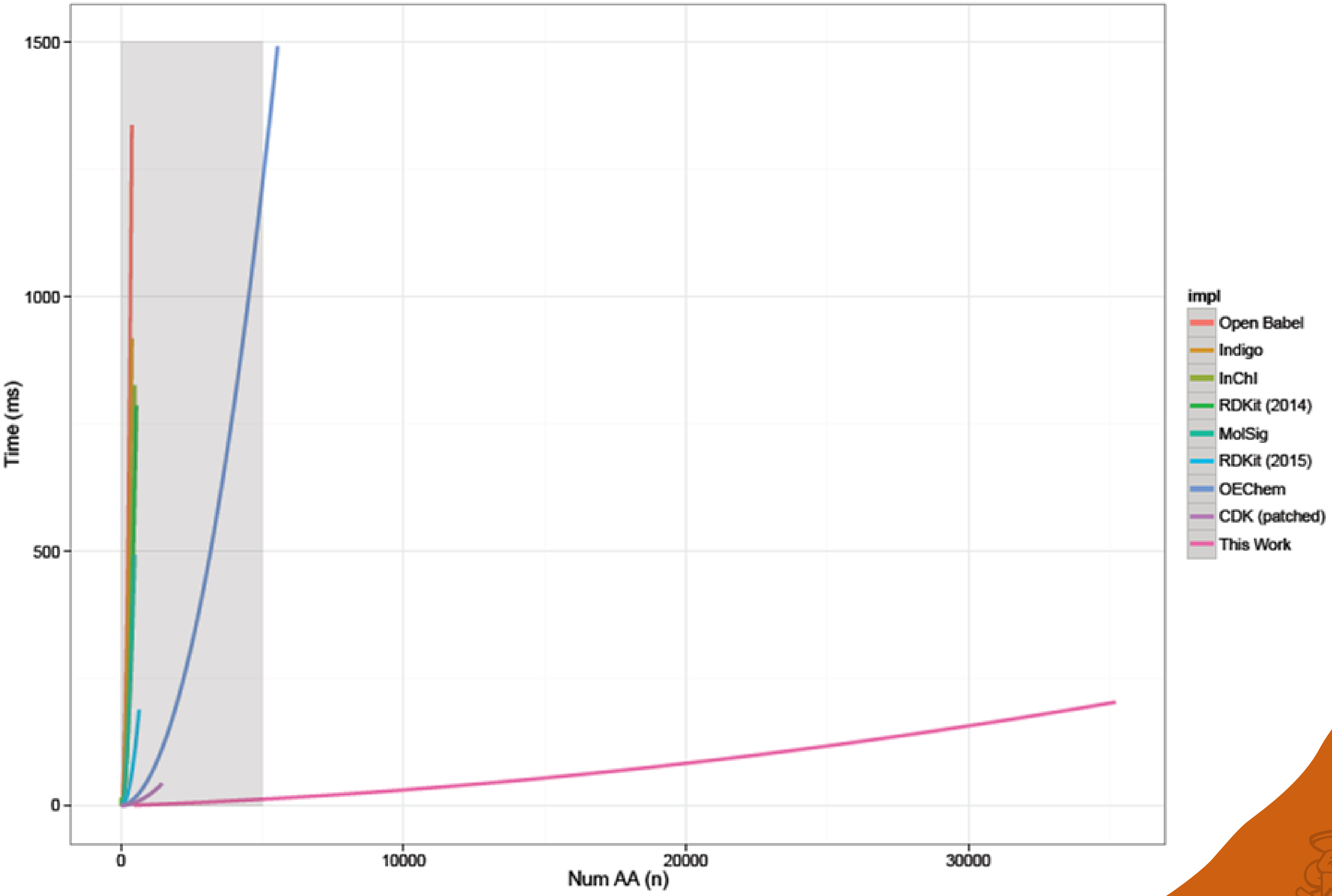
ALGORITHM SCALING TO 1000AA



ALGORITHM SCALING TO 5000AA



ALGORITHM SCALING TO MAXIMUM



PROPOSED NEW SUPPORT IN RDKIT

```
>>> m = Chem.MolFromSequence(“GREG”)
>>> Chem.MolToSmiles(m)
‘N=C(N)NCCC[C@H](NC(=O)CN)C(=O)N[C@@H](CCC(=O)O)C(=O)NCC(=O)O’
>>> print Chem.MolToSequence(m)
‘GREG’
>>> print Chem.MolToHELM(m)
`PEPTIDE1{G.R.E.G}$$$$’
>>> print Chem.MolToPDBBlock(m)
ATOM      1  N   GLY A   1       0.000   0.000   0.000  0.00
ATOM      2  CA  GLY A   1       0.000   0.000   0.000  0.00
ATOM      3  C   GLY A   1       0.000   0.000   0.000  0.00
ATOM      4  O   GLY A   1       0.000   0.000   0.000  0.00
ATOM      5  N   ARG A   2       0.000   0.000   0.000  0.00
ATOM      6  CA  ARG A   2       0.000   0.000   0.000  0.00
ATOM      7  C   ARG A   2       0.000   0.000   0.000  0.00
ATOM      8  O   ARG A   2       0.000   0.000   0.000  0.00
```

Proposed New Support in RDKit

```
>>> m = Chem.MolFromHELM("PEPTIDE1{C.Y.I.Q.N.C.P.L.G.[am]}$PEPTID
E1,PEPTIDE1,1:R3-6:R3$$$"))
>>> Chem.MolToSequence(m)
‘CYIQNCPLG’
>>> Chem.MolToSmiles(m)
‘CC[C@H](C)[C@@H]1NC(=O)[C@H](Cc2ccc(O)cc2)NC(=O)[C@@H](N)CSSC[C@
@H](C(=O)N2CCC[C@H]2C(=O)N[C@@H](CC(C)C)C(=O)NCC(N)=O)NC(=O)[C@H]
(CC(N)=O)NC(=O)[C@H](CCC(N)=O)NC1=O’
>>> Chem.MolToHELM(m)
`PEPTIDE1{C.Y.I.Q.N.C.P.L.G.[am]}$PEPTIDE1,PEPTIDE1,1:R3-6:R3$$$’
```

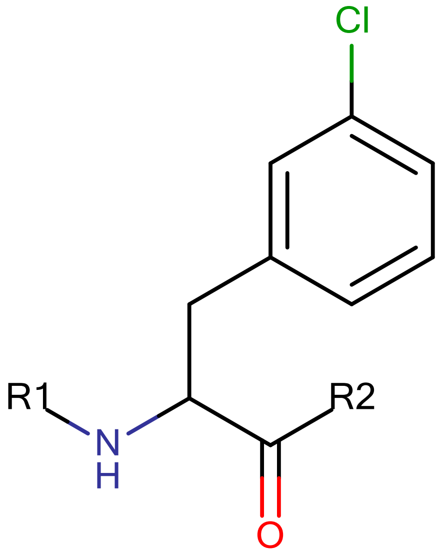
How this works under the hood

- Each RDKit::Atom contains an optional pointer to an AtomPDBResidueInfo structure for tracking residues.
- The field may be retrieved Atom::getMonomerInfo and set with Atom::setMonomerInfo.
- Internally, sequence information is encoded via PDB residue codes (ALA, ARG, ASN, ASP, CYS…).
- This is more expressive than 1-letter bioinformatics
  - D-amino acids: DAL, DAR, DAS, DSN, DCY…
  - Non-standard amino acids: SAR, ORN, NLE, NVA…
  - N-terminal acetyl: ACE, C-terminal amide: NH2

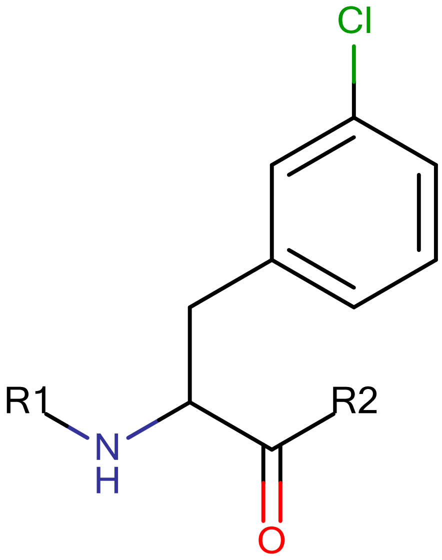
PROPOSED NEW SUPPORT IN RDKIT

```
>>> m = Chem.MolFromHELM("PEPTIDE1{L.[dF].P.V.[Orn].L.[dF].P.V.[O
rn]}$PEPTIDE1,PEPTIDE1,10:R2-1:R1$$$"))
>>> m.GetNumAtoms()
82
>>> Chem.MolToSequence(m)
‘LfPVXLfPVX’
>>> Chem.MolToSmiles(m)
‘CC(C)C[C@@H]1NC(=O)[C@H](CCCN)NC(=O)[C@H](C(C)C)NC(=O)[C@@H]2CCC
N2C(=O)[C@@H](Cc2ccccc2)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCCN)NC(=O
)[C@H](C(C)C)NC(=O)[C@@H]2CCCN2C(=O)[C@@H](Cc2ccccc2)NC1=O’
>>> print Chem.MolToHELM(m)
`PEPTIDE1{L.[dF].P.V.[Orn].L.[dF].P.V.[Orn]}$PEPTIDE1,PEPTIDE1,10
:R2-1:R1$$$’
```

HELM ISSUES (CHEMAXON)



ClC1=CC=CC(CC(N[*])C([*])=O)=C1
|$;;;;;;;;;_R1;;_R2;;$,c:3,12,t:1|



ClC1=CC(CC(N[*])C([*])=O)=CC=C1
|$;;;;;;;_R1;;_R2;;;;$,c:10,12,t:1|

RDKIT UGM HACKATHON SUGGESTION

- **Reaction atom-mapping in RDKit**
- The FMCS algorithm in RDKit could potentially be used as the basis for an atom-mapping algorithm in RDKit.
- Reaction atom mapping is a rare feature in cheminformatics toolkits, GGA's indigo and ChemAxon are exceptions.
- Benchmarks exist [Lowe & Sayle 2012 and 2013] for evaluating atom mapping algorithms based on fraction of reactions mapped, the number of bonds broken/formed, and the number of Carbon-Carbon bonds broken.
    - https://www.chemaxon.com/app/uploads/2013/07/NexMove.pdf
    - https://www.nextmovesoftware.com/products/AtomMapping.pdf

RDKit UGM Hackathon Support

- **Simple physiological ionization functionality**
- A recurrent suggestion of an RDKit Hackathon is to reproduce MOE’s “sdwash” functionality for adjusting protonation.
- A relatively small subset of this functionality (carboxylic acids and amines) would be extremely useful in combination with the new peptide/protein functionality described in this talk.
- Currently, proteins/peptides are generated in neutral form, which is appropriate for registration in databases.
- It would be extremely useful to have single function call to convert this form into the ionized form seen at physiological pH for 3D modelling, etc.

ACKNOWLEDGEMENTS

- The RDKit Hackers at Novartis
  - Greg Landrum
  - Nadine Schneider
  - Joann Prescott-Roy
- The team at NextMove Software
  - John May
  - Noel O’Boyle
  - Daniel Lowe
- And many thanks for your time!