Building a structure validation and standardization pipeline with RDKit

Riccardo Vianello

13th RDKit UGM, Zurich

September 12, 2024

NOVARTIS | Reimagining Medicine

Background and motivation

- The Small Molecule Registration system (SMR) at Novartis BR had been using the STRUCHK library since ~2012

- STRUCHK:
  - part of AvalonTools
  - very mature C library, but with some known issues/limitations
  - maintenance increasingly challenging
  - very difficult to adapt to evolving requirements

=> Implement an RDKit-based replacement for STRUCHK in SMR, and provide a more sustainable solution

Main requirements

- Focus on 2D Molfile representation (support both V2000 and V3000)
- Reject Molfile features not suitable for SMR (e.g., query atoms/bonds)
- Port STRUCHK's 2D layout and stereochemistry validation to RDKit
- Reimplement other STRUCHK's validation and standardization features using RDKit

- Preserve the explicit hydrogens and stereo bonds from the input

- Simple API (full execution wrapped by a single function call)
- Detailed error and information messages

API

```
// instantiate a Pipeline instance with default settings
MolStandardize::Pipeline pipeline;

// or

// configure the Pipeline with some specific options
MolStandardize::PipelineOptions options;
// [...] (options configuration)
MolStandardize::Pipeline pipeline(options);

// call the pipeline's "run" method on an input molblock
MolStandardize::PipelineResult result = pipeline.run(molblock);
```

API

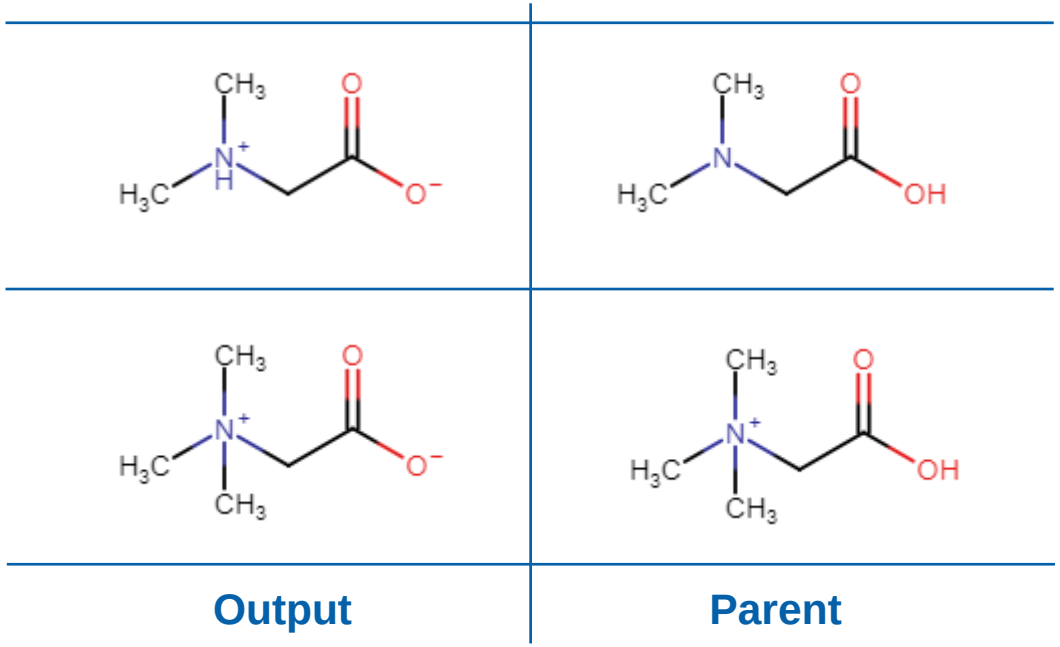
The returned MolStandardize::PipelineResult instance collects all the details about the pipeline execution:

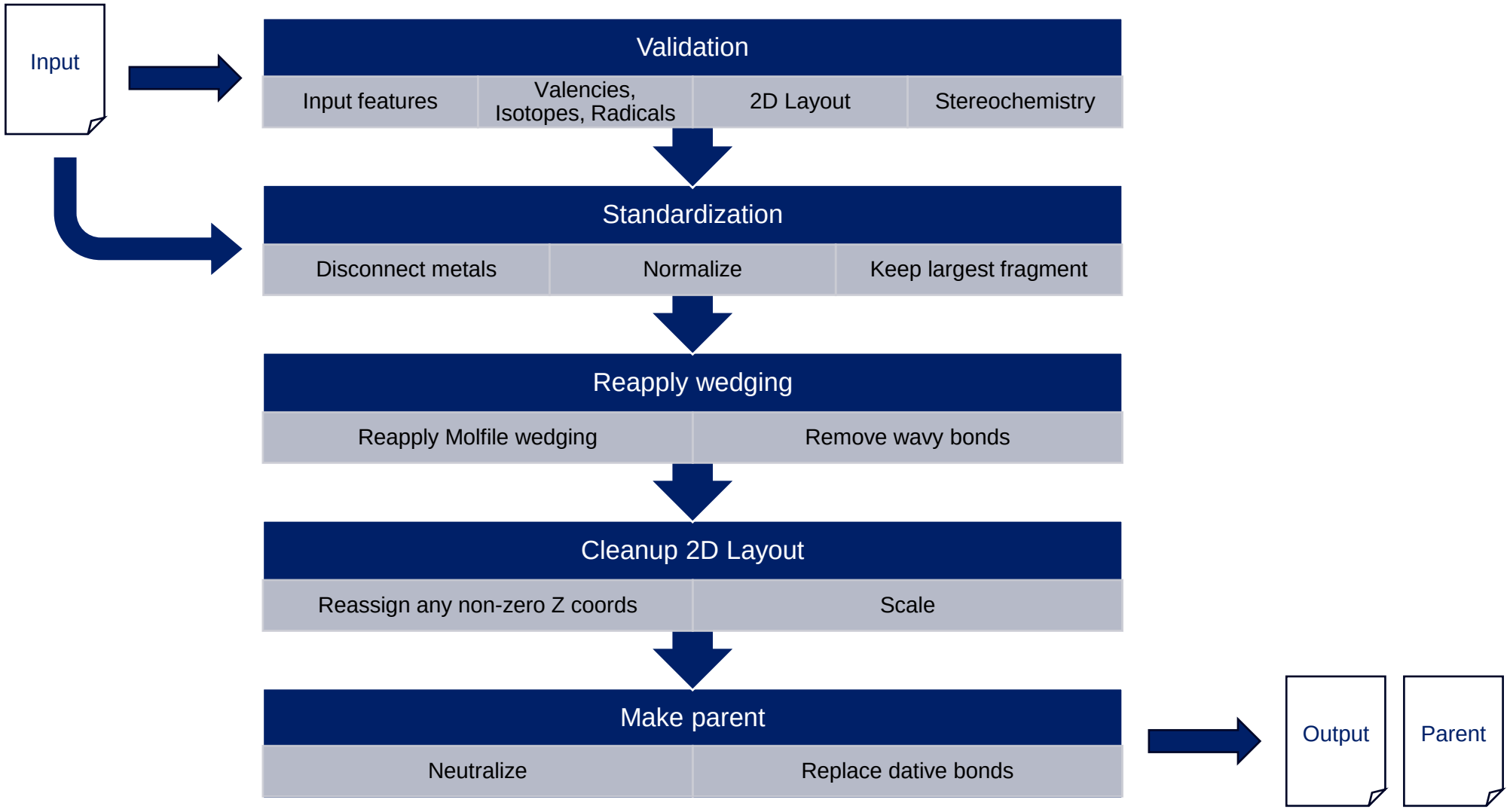
```
struct PipelineResult {
  PipelineStatus status; // bitmask summarizing all the errors and info
  std::uint32_t stage;   // did the pipeline run to completion?
  PipelineLog log;       // status flags and log messages of all events
  std::string inputMolData;
  std::string outputMolData;
  std::string parentMolData;
};
```

Output and Parent structure

- **Output: Represents the registered entry (=> the “display” molecule)**
- **Parent: Uniquely identifies the “concept” molecule (=> used for the registration key)**

Motivating use-case: zwitterions




Input
Validation
Input features
Valencies, Isotopes, Radicals
2D Layout
Stereochemistry
Standardization
Disconnect metals
Normalize
Keep largest fragment
Reapply wedging
Reapply Molfile wedging
Remove wavy bonds
Cleanup 2D Layout
Reassign any non-zero Z coords
Scale
Make parent
Neutralize
Replace dative bonds
Output
Parent

Current status

- Extensively tested on the internal collection of registered compounds (5M+ structures)
- Validation/standardization results were compared with STRUCHK, differences were carefully reviewed and either resolved or justified
- Internal RDKit build deployed to prod in June (~15K+ new compounds registered)
- Main PR merged into the upstream RDKit repo in July
- To be soon aligned to the upcoming RDKit 2024_09 release for better long-term support

Acknoledgements

Novartis

- Sandra Wildhaber
- Paolo Tosco
- Jason Elliott
- Markus Furegati
- Constanze Hartwieg
- John Isbell
- Maurice van Eis

RDKit

- Greg Landrum

Thank you

NOVARTIS | Reimagining Medicine

Configurability

```
struct PipelineOptions {

  // parsing
  bool strictParsing{false};

  // validation
  bool reportAllFailures{true};
  bool allowEmptyMolecules{false};
  bool allowEnhancedStereo{false};
  bool allowAromaticBondType{false};
  bool allowDativeBondType{false};
  double is2DZeroThreshold{1e-3};
  double atomClashLimit{0.03};
  double minMedianBondLength{1e-3};
  double bondLengthLimit{100.};
  bool allowLongBondsInRings{true};
  bool allowAtomBondClashExemption{true};

  // metal disconnector options
  std::string metalNof{"[Li,Na,K,Rb,Cs,Fr]~[#7,#8,F]"};
  std::string metalNon{};

  // normalizer options
  std::string normalizerData{"[...]"};

  // serialization
  bool outputV2000{false};

};
```

Customizability

The C++ API provides a basic mechanism for overriding the pipeline stages with custom functions

```
// Two example snippets from the MolStandardize unit tests

MolStandardize::Operations::PipelineVector ops{{1, &chargeParentLocal}};
pipeline.setStandardizationSteps(ops);
pipeline.setMakeParent(&parentNoOp);

// [...]

pipeline.setValidationSteps({});  // no validation
pipeline.setParse(&smilesParse);
pipeline.setSerialize(&smilesSerialize);
```