bpforms documentation

Release 0.0.16

Karr Lab

Jul 09, 2020

Contents

BpForms is a set of tools for concretely representing the primary structures of non-canonical forms of biopolymers, such as oxidized DNA, methylated RNA, and acetylated proteins, and calculating properties of non-canonical biopolymers.

BpForms emcompasses five tools:

- A grammar for concretely describing the primary structures of non-canonical biopolymers. See Section 1.3.1 for detailed information. For example, the following represents a modified DNA molecule that contains a deoxyinosine monomeric form at the fourth position:

  ```
  ACG{dI}
  ```

  This concrete representation enables the *BpForms* software tools to calculate properties of non-canonical biopolymers.

- Tools for calculating properties of non-canonical biopolymers including their chemical formulae, molecular weights, charges, and major protonation and tautomerization states.
  - A web-based graphical interface: See https://bpforms.org and Section 1.3.2.
  - A JSON REST API: See https://bpforms.org/api and Section 1.3.3.
  - A command line interface: See Section 1.3.4.
  - A Python API: See Section 1.3.5.

BpForms was motivated by the need to concretely represent the biochemistry of DNA modification, DNA repair, post-transcriptional processing, and post-translational processing in whole-cell computational models. *BpForms* is also a valuable tool for experimental proteomics and synthetic biology. In particular, we developed *BpForms* because there were no notations, schemas, data models, or file formats for concretely representing modified forms of biopolymers, despite the existence of several databases and ontologies of DNA, RNA, and protein modifications, the ProForma Proteoform Notation, and the MOMODICS codes for modified RNA bases.

BpForms can be combined with BcForms to concretely describe the primary structure of complexes.

CHAPTER 1

Contents

1.1 Uses cases for epigenomics, proteomics, systems biology, synthetic biology, and proteomics

By concretely capturing the primary structure of biopolymers, *BpForms* can help facilitate a wide range of epigenomics, proteomics, proteomics systems biology, and synthetic biology research.

Please see https://www.bpforms.org for information about these use cases, including detailed examples.

1.2 Installation

The following is a brief guide to installing *BpForms*. The Dockerfile in the *BpForms* Git repository contains detailed instructions for how to install *BpForms* in Ubuntu Linux.

1.2.1 Prerequisites

First, install the third-party packages listed below.

- ChemAxon Marvin: optional to calculate major protonation and tautomerization states
  - Java >= 1.8
- Open Babel
- Pip >= 18.0
- Python >= 3.6

To use ChemAxon Marvin to calculate major protonation and tautomerization states, set `JAVA_HOME` to the path to your Java virtual machine (JVM) and add Marvin to the Java class path:

```
export JAVA_HOME=/usr/lib/jvm/default-java
export CLASSPATH=$CLASSPATH:/opt/chemaxon/marvinsuite/lib/MarvinBeans.jar
```

1.2.2 Latest release From PyPI

Run the following command to install the latest release from PyPI.:

```
pip install bpforms
```

1.2.3 Latest revision from GitHub

Run the following command to install the latest version from GitHub.:

```
pip install git+https://github.com/KarrLab/pkg_utils.git#egg=pkg_utils
pip install git+https://github.com/KarrLab/wc_utils.git#egg=wc_utils[chem]
pip install git+https://github.com/KarrLab/bpforms.git#egg=bpforms
```

1.2.4 Installing the optional features

To calculate major protonation and tautomerization states, *BpForms* must be installed with the *[protontation]* option:

```
pip install bpforms[protontation]
pip install git+https://github.com/KarrLab/bpforms.git#egg=bpforms[protontation]
```

To draw molecules, *BpForms* must be installed with the *[draw]* option:

```
pip install bpforms[draw]
pip install git+https://github.com/KarrLab/bpforms.git#egg=bpforms[draw]
```

To export the alphabets in OBO format, *BpForms* must be installed with the *[onto_export]* option:

```
pip install bpforms[onto_export]
pip install git+https://github.com/KarrLab/bpforms.git#egg=bpforms[onto_export]
```

To install the rest API, *BpForms* must be installed with the *[rest_api]* option:

```
pip install bpforms[rest_api]
pip install git+https://github.com/KarrLab/bpforms.git#egg=bpforms[rest_api]
```

1.3 Tutorial

This section contains tutorials for the grammar, web application, REST API, command line interface, and Python API. Additional interactive tutorials for the Python API are available as Jupyter notebooks at https://sandbox.karrlab.org.

1.3.1 *BpForms* grammar

The *BpForms* grammar unambiguously represents the primary structure of biopolymer forms that contain canonical and non-canonical monomeric forms using (a) a syntax similar to IUPAC/IUBMB and (b) extended alphabets for DNA, RNA, and proteins to describe monomeric forms.

BpForms.org contains a detailed overview of the grammar and examples.

Grammar

The following is the definition of the *BpForms* grammar. The grammar is defined in Lark syntax which is based on EBNF syntax.

```
start: seq (attr_sep global_attr)* WS?
seq: (WS|monomer)((WS? nick)? (WS|monomer))*
?monomer: alphabet_monomer | inline_monomer
alphabet_monomer: CHAR | DELIMITED_CHARS
inline_monomer: OPEN_SQ_BRACKET WS? inline_monomer_attr (attr_sep inline_monomer_
↪attr)* WS? CLOSE_SQ_BRACKET
?inline_monomer_attr: id | name | synonym | identifier | structure | backbone_bond_
↪atom | backbone_displaced_atom | r_bond_atom | l_bond_atom | r_displaced_atom | l_
↪displaced_atom | delta_mass | delta_charge | position | base_monomer | comments
id: ID_FIELD_NAME field_sep ESCAPED_STRING
name: NAME_FIELD_NAME field_sep ESCAPED_STRING
synonym: SYNONYM_FIELD_NAME field_sep ESCAPED_STRING
identifier: IDENTIFIER_FIELD_NAME field_sep identifier_id identifier_sep identifier_ns
?identifier_ns: ESCAPED_STRING
?identifier_id: ESCAPED_STRING
structure: STRUCTURE_FIELD_NAME field_sep QUOTE_DELIMITER SMILES QUOTE_DELIMITER
backbone_bond_atom: BACKBONE_BOND_ATOM_FIELD_NAME field_sep atom
backbone_displaced_atom: BACKBONE_DISPLACED_ATOM_FIELD_NAME field_sep atom
r_bond_atom: RIGHT_BOND_ATOM_FIELD_NAME field_sep atom
l_bond_atom: LEFT_BOND_ATOM_FIELD_NAME field_sep atom
r_displaced_atom: RIGHT_DISPLACED_ATOM_FIELD_NAME field_sep atom
l_displaced_atom: LEFT_DISPLACED_ATOM_FIELD_NAME field_sep atom
atom: atom_element INT atom_charge?
?atom_element: /[A-Z][a-z]?/
?atom_charge: /[\+\-][0-9]+/
delta_mass: DELTA_MASS_FIELD_NAME field_sep DALTON
delta_charge: DELTA_CHARGE_FIELD_NAME field_sep CHARGE
position: POSITION_FIELD_NAME field_sep INT? DASH INT? monomers_position?
monomers_position: WS? OPEN_SQ_BRACKET (CHAR|CHARS) (WS? "|" WS? (CHAR|CHARS))* CLOSE_
↪SQ_BRACKET
base_monomer: BASE_MONOMER_FIELD_NAME field_sep QUOTE_DELIMITER (CHAR|CHARS) QUOTE_
↪DELIMITER
comments: COMMENTS_FIELD_NAME field_sep ESCAPED_STRING
?global_attr: crosslink | circular

crosslink: CROSSLINK_FIELD_NAME field_sep OPEN_SQ_BRACKET WS? (onto_crosslink |␣
↪inline_crosslink) WS? CLOSE_SQ_BRACKET

onto_crosslink: onto_crosslink_attr (attr_sep onto_crosslink_attr)*
onto_crosslink_attr: onto_crosslink_type | onto_crosslink_monomer
onto_crosslink_monomer: onto_crosslink_monomer_type field_sep INT
onto_crosslink_monomer_type: /[lr]/
onto_crosslink_type: "type" field_sep QUOTE_DELIMITER CHARS QUOTE_DELIMITER

inline_crosslink: (inline_crosslink_attr (attr_sep inline_crosslink_attr)*)?
inline_crosslink_attr: inline_crosslink_atom | inline_crosslink_order | inline_
↪crosslink_stereo | inline_crosslink_comments
inline_crosslink_atom: inline_crosslink_atom_type field_sep INT atom_element INT atom_
↪charge?
inline_crosslink_atom_type: (LEFT_BOND_ATOM_FIELD_NAME | RIGHT_BOND_ATOM_FIELD_NAME |␣
↪LEFT_DISPLACED_ATOM_FIELD_NAME | RIGHT_DISPLACED_ATOM_FIELD_NAME)
inline_crosslink_order: "order" field_sep QUOTE_DELIMITER /
↪(single|double|triple|aromatic)/ QUOTE_DELIMITER
```

(continues on next page)

(continued from previous page)

```
inline_crosslink_stereo: "stereo" field_sep QUOTE_DELIMITER /(wedge|hash|up|down)/␣
↪QUOTE_DELIMITER
inline_crosslink_comments: COMMENTS_FIELD_NAME field_sep ESCAPED_STRING

nick: ":"

circular: "circular"

ID_FIELD_NAME: "id"
NAME_FIELD_NAME: "name"
SYNONYM_FIELD_NAME: "synonym"
IDENTIFIER_FIELD_NAME: "identifier"
STRUCTURE_FIELD_NAME: "structure"
BACKBONE_BOND_ATOM_FIELD_NAME: "backbone-bond-atom"
BACKBONE_DISPLACED_ATOM_FIELD_NAME: "backbone-displaced-atom"
LEFT_BOND_ATOM_FIELD_NAME: "l-bond-atom"
LEFT_DISPLACED_ATOM_FIELD_NAME: "l-displaced-atom"
RIGHT_BOND_ATOM_FIELD_NAME: "r-bond-atom"
RIGHT_DISPLACED_ATOM_FIELD_NAME: "r-displaced-atom"
DELTA_MASS_FIELD_NAME: "delta-mass"
DELTA_CHARGE_FIELD_NAME: "delta-charge"
POSITION_FIELD_NAME: "position"
BASE_MONOMER_FIELD_NAME: "base-monomer"
COMMENTS_FIELD_NAME: "comments"
CROSSLINK_FIELD_NAME: "x-link"

?attr_sep: WS? "|" WS?
?field_sep: WS? ":" WS?
?identifier_sep: WS? "@" WS?

CHAR: /[^\[\]\{\}":\| \t\f\r\n]/
CHARS: CHAR /[^\[\]\{\}":\|\t\f\r\n]*/ CHAR
DELIMITED_CHARS: "{" (CHAR|CHARS) "}"
SMILES: /[^"]+/
DALTON: /[\-\+]?[0-9]+(\.[0-9]*)?/
CHARGE: /[\-\+]?[0-9]+/
OPEN_SQ_BRACKET: "["
CLOSE_SQ_BRACKET: "]"
QUOTE_DELIMITER: "\""
DASH: "-"
WS: /[ \t\f\r\n]+/
INT: /[0-9]+/
ESCAPED_STRING: /"(?:[^"\\]|\\.)*"/
```

1.3.2 Graphical web interface

The *BpForms* website provides a convenient interface to explore *BpForms* and calculate (bio)chemical properties of biopolymer forms.

The website provides a single input box to enter biopolymer forms. After clicking submit, the website will perform three actions

1. The website will validate the form and report any syntax errors.
2. Optionally, the website will calculate the major protonation and tautomerization states at the chosen pH.
3. The website will calculate the length, chemical formula, mass, and charge of the submitted form.

1.3.3 REST API

The REST JSON API enables researchers to programmatically validate and calculate the properties of biopolymer forms.

The root URL for the REST API is https://bpforms.org/api/. The REST API provides endpoints to get a list of available alphabets, to get the monomeric forms in an alphabet, and to calculate properties of biopolymers, optionally, at a specific pH. Please see the inline documentation for more information.

1.3.4 Command line interface

The command line interface provides four functions to easily manipulate *BpForms*-encoded descriptions of biopolymers.

- **Get help with the 'BpForms' command line interface.** The following commands return inline help information about the command line interface:

```
bpforms
bpforms -h
bpforms --help
```

- **Validate a 'BpForms'-encoded description of a form of a biopolymer.** The following command can be used to verify if a sequence of monomeric forms is syntactically and semantically valid. The command line interface will print any errors to the standard error:

```
bpforms validate --help

bpforms validate <alphabet_name> <bpform_sequence>

bpforms validate dna 'ACGT | circular'
# Form is valid

bpforms validate protein 'CRATUG'
# Form is valid
```

- **Calculate the protonation of a form of a biopolymer.** The following command can be used to calculate the major protonation and tautomerization state of each monomeric form in a biopolymer form. This command will print the sequence of the major protonation/tautomerization states. Note, this function requires a structure for each monomeric form:

```
bpforms get-major-micro-species --help

bpforms get-major-micro-species <alphabet_name> <bpform_sequence> <ph_value>

bpforms get-major-micro-species dna 'ACGT | circular' 7
# Cc1cn(C2CC(O)C(COP(=O)([O-])OC3CC(OC3COP(=O)([O-]
↪])OC3CC(OC3COP(=O)([O-])[O-
↪])n3cnc4c(N)ncnc34)n3ccc(N)nc3=O)n3cnc4c3nc(N)[nH]c4=O)O2)c(=O)[nH]c1=O

bpforms get-major-micro-species protein 'CRATUG' 7
#␣
↪C[C@@H](O)[C@H](NC(=O)[C@H](C)NC(=O)[C@H](CCCNC(=[NH2+])N)NC(=O)[C@@H]([NH3+])CS)C(=O)N[C@@H](
↪]
```

- **Calculate physical properties of a form of a biopolymer.** The following command can be used to calculate the length, formula, mass, and charge of a bipolymer form. The optional `ph` argument can be used to calculate

the major protonation and tautomerization state of the biopolymer at a specific pH. Note, this function requires a structure for each monomeric form:

```
bpforms get-properties --help

bpforms get-properties <alphabet_name> <bpform_sequence> [--ph ph]

bpforms get-properties dna 'ACGT | circular'
# Length: 4
# Structure: O(C1CC(OC1COP(=O)([O-])[O-
↪])n1cnc2c1ncnc2N)P(=O)(OCC1C(OP(=O)(OCC2C(OP(=O)(OCC3C(O)CC(O3)n3cc(C)c(=O)[nH]c3=O)[O-
↪])CC(O2)n2cnc3c2nc(N)[nH]c3=O)[O-])CC(O1)n1ccc(nc1=O)N)[O-]
# Formula: C39H46N15O25P4
# Molecular weight: 1248.772047992
# Charge: -5

bpforms get-properties protein 'CRATUG'
# Length: 6
# Structure:␣
↪C(=O)([C@@H]([NH3+])CS)N[C@H](C(=O)N[C@@H](C)C(=O)N[C@@H]([C@@H](C)O)C(=O)N[C@H](C(=O)NCC(=O)O
# Formula: C21H41N9O8SSe
# Molecular weight: 658.645
# Charge: 2
```

1.3.5 Python API

The following tutorial illustrates how to use the *BpForms* Python API. An interactive version of this tutorial is also available in the whole-cell modeling sandbox.

Importing *BpForms*

Run this command to import *BpForms*:

```
import bpforms
```

Creating biopolymer forms

Use the *BpForms* grammar and the `bpforms.BpForm.from_str` method to create an instance of `bpforms.BpForm` that represents a form of a biopolymer:

```
dna_form = bpforms.DnaForm().from_str('ACG{m2C}AC')
```

Getting and setting monomeric forms

Individual monomeric forms and slices of monomeric forms can be get and set similar to lists:

```
dna_form[0]
    => <bpforms.core.Monomer at 0x7fb365341240>

dna_form[1] = bpforms.dna_alphabet.monomers.A

dna_form[1:3]
```

(continues on next page)

(continued from previous page)

```
    => [<bpforms.core.Monomer at 0x7fb365341240>, <bpforms.core.Monomer at␣
↪0x7fb365330cf8>]

dna_form[1:3] = bpforms.DnaForm().from_str('TA')
```

Getting and setting the base of a monomeric form

Optionally, *BpForms* can track the monomeric forms that are generated from a monomeric form (e.g. m2A is generated from A). This can be get and set using the `bpforms.Monomer.base_monomers` attribute. This attribute is a `set` of `bpforms.Monomer`:

```
di_monomer = dna_form[3]
di_monomer.base_monomers
    => set(<bpforms.core.Monomer at 0x7fb365341240>)
```

Protonation and tautomerization

Calculate the major protonation and tautomerization state of each monomeric form in the biopolymer form:

```
dna_form.get_major_micro_species(8., major_tautomer=True)
```

Calculation of physical properties

Use these commands to calculate the length, formula, molecular weight, and charge of the biopolymer form:

```
len(dna_form)
    => 6

dna_form.get_formula()
    => AttrDefault(<class 'float'>, False, {'C': 59.0, 'N': 23.0, 'O': 35.0, 'P': 6.0,
↪ 'H': 72.0})

dna_form.get_mol_wt()
    => 1849.193571988

dna_form.get_charge()
    => -7
```

Generating IUPAC/IUBMB sequences for *BpForms*

The `get_canonical_seq` method generates IUPAC/IUBMB representations of *BpForms*. Where annotated, this method uses the `base_monomers` attribute to represent modified monomeric forms using the code for their root (e.g. m2A is represented as "A"). Monomeric forms that don't have their base annotated are represented as "N" and "X" for nucleic acids and proteins, respectively:

```
dna_form.get_canonical_seq()
    => ATANAC
```

Determine if two biopolymers describe the same structure

Use the following command to determine if two instances of `BpForm` describe the same biopolymer:

```
dna_form_1 = bpforms.DnaForm().from_str('ACGT')
dna_form_2 = bpforms.DnaForm().from_str('ACGT')
dna_form_3 = bpforms.DnaForm().from_str('GCTC')

dna_form_1.is_equal(dna_form_2)
    => True

dna_form_1.is_equal(dna_form_3)
    => False
```

1.4 Alphabets of residues

BpForms comes with six alphabets:

- Canonical DNA: The canonical DNA nucleotide monophosphates.
- Canonical RNA: The canonical RNA nucleotide monophosphates.
- Canonical protein: The 20 canonical protein residues.
- DNA: The canonical DNA nucleotide monophosphates, plus non-canonical DNA nucleotide monophosphates based on DNAmod.
- RNA: The canonical RNA nucleotide monophosphates, plus non-canonical RNA nucleotide monophosphates based on MODOMICS and the RNA Modification Database.
- Protein: The 20 canonical protein residues, plus the non-canonical protein residues in the PDB Chemical Component Dictionary and RESID.

1.4.1 Construction of the public alphabets

The alphabets of DNA, RNA, and protein residues were developed by merging residues from multiple databases:

- The DNA alphabet was developed by combining the deoxyribose nucleotide monophosphates and 3' and 5' DNA ends from the PDB Chemical Component Dictionary (PDB CCD) with the verified DNA nucleobases from DNAmod and the deoxyribose nucleosides from REPAIRtoire that had concrete structures.
- The RNA alphabet was developed by combining the ribose nucleotide monophosphates and 3' and 5' RNA ends from the PDB CCD with the ribose nucleosides from MODOMICS and the RNA Modification Database that had concrete structures.
- The protein alphabet was developed by merging residues and ends from the PDB CCD and RESID.

Specifically, the alphabets were constructed as follows:

1. We downloaded, scraped, and manually extracted residues from DNAmod, MODOMICS, the PDB CCD, REPAIRtoire, RESID, the RNA Modification Database.
2. We parsed each database into a list of residues.
3. We rejected residues with incompletely defined structures, as well as inconsistent residues such as nucleotides from DNAmod.

4. We normalized the DNA and RNA residues to nucleotide monophosphates and normalized the protein residues to amino acids. For example, we transformed the DNAmod entries to nucleotides by adding deoxyribose monophosphate to each nucleobase.
5. We merged the repeated residues. This included residues that had the same molecular structure, that the upstream sources annotated were equivalent, or that had similar names.
6. We identified the atom indices of the left/preceding and right/following bonding sites in each residue. For DNA and RNA, the left and right bonding sites comprised the phosphorus atom in the phosphate group bonded to the 5' carbon and the oxygen atom bonded to the 3' carbon, respectively. For protein residues, these comprised the nitrogen atom in the amino group and the acidic oxygen atom in the carboxyl group.

The above steps are implemented by `bpforms.util.build_alphabets()`.

Updating the public alphabets

`bpforms.util.build_alphabets()` can be used to pull updates to DNAmod, MODOMICS, the PDB CCD, REPAIRtoire, RESID, and the RNA Modification Database into the alphabets. We plan to use this function to update the alphabets annually or as needed.

1.4.2 Contributing to the public alphabets

We welcome contributions to the alphabets, as well as new alphabets. See Section 1.12 for information about how to contribute to *BpForms*.

1.4.3 Building additional "user" alphabets

Users can also create additional alphabets by creating additional instances of `bpforms.Alphabet`. Users can either add monomeric forms programmatically or load them from YAML files. Users can add monomeric forms programmatically by creating instances of `bpforms.Monomer` and adding them to the `monomers` attribute of `bpforms.Alphabet`, which is a dictionary that maps the character codes of monomeric forms to monomeric forms. Users can load monomeric forms from YAML files by using the `from_yaml` method of `bpforms.Alphabet`. Please see bpforms/alphabet/dna.yml for an example of the YAML alphabet format.

1.5 Ontology of crosslinks

As of March 2020, the ontology of crosslinks contains 36 crosslinks between protein residues.

We developed the ontology based on entries in RESID which represent crosslinked dipeptides:

1. We searched RESID for entries that represent crosslinked dipeptides.
2. We identified the individual residues which participate in each dimer.
3. We used ChemAxon Marvin to identify the atoms involved in each crosslink.
4. We used Open Babel to determine the indices of these atoms in the canonical SMILES ordering of the atoms.
5. We manually assigned an id and name to each crosslink.

We invite the community to help us curate additional crosslinks, including DNA-DNA, RNA-RNA, DNA-protein, and RNA-protein crosslinks. Please contact us at info@karrlab.org to get involved, or please submit additional crosslinks via GitHub pull requests or issues.

1.6 Managing revisions to the grammar and the ontologies and user ontologies

Over time, we will likely introduce new capabilties into the grammar (e.g., to represent additional types of uncertainty) and expand (and, if necessary, correct) the ontologies. In addition, we anticipate that users will develop their own ontologies of residues and crosslinks. Consequently, there will likely be needs to (a) compare molecules that are described with different versions of the grammar, different versions of the ontologies, and different ontologies and (b) convert descriptions of molecules between versions of the grammar, versions ontologies, and different ontologies.

1.6.1 Comparing descriptions of molecules described with different versions

Because each combination of (a) a version of the grammar and (b) a set of versions of the ontologies will be capable of representing the molecular structures (atoms and bonds) of polymers, it will be possible to compare molecules described with different versions of the grammar, different versions of the ontologies, and different ontologies at the molecular level.

Below are our plans for faciliating such comparisons:

- Separate the alphabets of residues and ontology of crosslinks into their own Git repositories so their revisions can be separately tracked from that of the *BpForms* software.
- Create a repository for user ontologies. This could be implemented as a Git repository.
- Expand the grammar and data structures to capture the ontologies (including user ontologies) used to describe each molecule.
- Expand the grammar and data structures to capture the versions of the grammar and the versions of the ontologies used to describe each molecule.
- Implement a method for comparing molecules describe with different versions of the grammar, different ontologies, or different versions of the ontologies. This method will use the Git repositories to resolve the specific versions of the grammar and ontologies.

1.6.2 Migrating descriptions of molecules between different ontologies and versions

To make it easier for researchers to work with revisions to the grammar and ontologies, it will also be useful to develop a utility for migrating descriptions of molecules between different versions of the grammar and/or different ontologies and versions of the ontologies. Below are our plans for facilitating such migrations.

- Implement data structures for capturing changes to the grammar and ontologies.
- Use these data structures to implement a method for migrating descriptions of molecules between successive versions of the grammar or ontologies.
- Implement a second method for migrating descriptions of molecules between arbitrary versions of the grammar and ontologies by chaining together multiple executions of the first method.

1.7 Interoperating between ontologies and with other formats

1.7.1 Interoperating between user alphabets of residues

The goal of *BpForms* is to faciliate precise communication about polymers. From a technical perspective, the easiest way to achieve this would be to require all users to use the same alphabets, require each character in each alphabet

to represent a unique molecular structure, and not allow users to define residues inline or to define their own alphabets. With only one set of alphabets, it would be easy to compare descriptions of molecules and generate high-level descriptions of any differences. However, we believe that *BpForms* should enable users to define their own residues and even entire alphabets because (a) even the simplest process for contributing residues to the public alphabets could be a barrier for some users, and we prefer users to use *BpForms* rather than not describe polymers precisely at all and (b) some fields may prefer to manage their own alphabets (without a large number of users yet, we think it may be difficult to influence every field – structural biology, transcriptomics, proteomics, systems biology, synthetic biology – to converage on one set of alphabets).

Consequently, there will likely be a need to interoperate between alphabets. Because *BpForms* describes concrete molecular structures (i.e. atoms and bonds), *BpForms* can compare molecules described with different alphabets and convert descriptions of molecules between alphabets using the molecular meanings of the descriptions of molecules. Below are our plans for facilitating such interoperability:

- Expand the grammar and data structures to capture the alphabet (including user alphabets) used to describe each molecule.
- Implement a method for comparing molecules described with different alphabets by generating the molecular structures for the molecules and comparing these structures.
- Implement a method for migrating descriptions of molecules between alphabets by (i) using the molecular structures of the residues in the alphabets to build a mapping from the residues of the source alphabet to the residues of the target alphabet, (ii) using this mapping to migrate all mapped residues, and (iii) replacing all unmapped residues with user-defined residues.

1.7.2 Interoperating between user ontologies of crosslinks

Interoperability between ontologies of crosslinks can be managed similar to the interoperating between alphabets of residues as discussed above.

1.7.3 Interoperating with other formats

Similarly, *BpForms* can interoperate (e.g., compare, calculate differences) with any other molecularly-precise format, such as the InChI, SMILES, and BigSMILES, through converting each format to a common molecular representation.

1.8 Integrating BpForms into pathway databases, models, and genetic designs

Please see *https://www.bpforms.org* for information about integrating BpForms with commonly used standards for genomics, transcriptomics, proteomics, systems biology and synthetic biology:

- BioPAX: knowledge of pathways
- CellML: kinetic models
- FASTA: sets of sequences
- PDB: 3D structures of macromolecules
- SBML: kinetic models, and
- SBOL: in silico designs.

1.9 Resources for reconstructing modified DNA, RNA, and proteins

The following are valuable resources for reconstructing DNA, RNA, and protein modifications:

- DNA
  - DNAmod: Database of covalently modified nuceleobases
  - MethDB: Database of DNA methylation
  - MethSMRT: Database for DNA 6mA and 4mC methylomes
- RNA
  - Modomics: Databases of modified NMPs and modified RNA
  - RMBase: Database of posttranscriptionally modified RNAs
  - RNA Modification Database: Database of posttranscriptionally modified RNA nucleosides
- Proteins
  - dbPTM: Database of modified amino acids and proteins
  - Delta Mass: Database of modified amino acids
  - FindMod: Database of post-translational modifications
  - iPTMnet: Database of post-translational modifications
  - PDB Chemical Components: Database of modified amino acids
  - PDB in Europe Chemical Components: Database of modified amino acids
  - Protein Ontology: Database of modified proteins
  - PSIMOD: Ontology of modified amino acids
  - RESID: Database of modified protein residues
  - UniMod: Database of modified amino acids
  - UniProt: Database of modified amino acids in proteins
  - UniProt Controlled Vocabulary of Posttranslational Modifications: Database of modified amino acids
- Drawing structures of monomeric forms
  - ChemAxon Marvin: Software for drawing structures of monomeric forms
  - Open Babel: Software for calculating the numbers of the atoms in monomeric forms

1.10 Comparison of *BpForms* with other formats, databases, and software

1.10.1 Comparison with BigSMILES

- *BpForms* is backward compatible with the IUPAC/IUBMB format
- *BpForms* supports high-level naming of residues, crosslinks, nicks, and circularity
- With *BcForms*, *BpForms* can be compared into yet more abstract descriptions of complexes
- *BpForms* can capture missing or uncertain knowledge about molecules

- There are software tools for working with *BpForms*

1.10.2 Comparison with BioPAX

- *BpForms* can represent non-canonical DNA, RNA, and proteins
- *BpForms* can represent any modification and, therefore, is not limited to modifications that have been previously enumerated in the *PSI Molecular Interation ontology <https://www.ebi.ac.uk/ols/ontologies/mi>*. This is necessary to represent the combinatorial complexity of non-canonical DNA, RNA, and proteins.
- *BpForms* concretely represents the primary structure of biopolymers. In constrast, the chemical structure implied by BioPAX features and chemical reactions is unclear.
- *BpForms* is easier to embed into other files such as SBML-encoded models.
- *BpForms* includes software for interpreting descriptions of non-canonical biopolymers. This enables calculations of properties such as chemical formulae, masses, and charges, which is essential for modeling and other applications.

1.10.3 Comparison with DNAmod

- The *BpForms* DNA alphabet is internally consistent. Each monomeric form represents a nucleotide monophosphate. This ensures the monomeric forms can be composed into polymers, as the semantic meaning of the polymers is well-defined.

1.10.4 Comparison with HELM

- The *BpForms* grammar is fully backward compatible with the IUPAC/IUBMB format. While the HELM format is similar to the IUPAC/IUBMB format, the HELM representation of an unmodified sequence is not equivalent to its IUPAC/IUBMB representation.
- *BpForms* supports inline definitions of residues. This enables users to describe molecules that involve new residues without having to create their own alphabet or edit the public alphabets. In contrast, HELM has a more hierarchical design that requires users to describe all new residues in a custom alphabet. We believe this is more cumbersome, particular for use cases where users want to embed descriptions of molecules into other files (e.g., Excel workbooks or PDF documents) without having to deal with references to files that define alphabets.
- *BpForms* provides a high-level representation of crosslinks. HELM does not support this.
- *BpForms* can capture missing or uncertain information about molecules, which is essential for biological research. HELM does not support this.
- With *BcForms*, *BpForms* can be composed into yet higher-level descriptions of complexes. In contrast, HELM can only capture complexes whose subunits are linked by crosslinks. *BcForms* is more flexible, and can represent a complex as a bag of subunits. This flexibility is essential for representing complexes that do not involve crosslinks or for which crosslink information is not available.

1.10.5 Comparison with MODOMICS

- *BpForms* can represent DNA, RNA, and proteins including the diversity of ids of monomeric forms used by MODOMICS, RESID, and other databases.
- *BpForms* can represent any modification, and is not limited to the modifications catalogued in MODOMICS.

- *BpForms* concretely captures the bonds between adjacent monomeric forms, avoiding ambiguity about how monomeric forms are composed into polymers. This is particularly important for monomeric forms that have multiple 3' and 5' sites.
- *BpForms* can represent bonds between non-adjacent monomeric forms, such as crosslinks.
- *BpForms* can represent circular biopolymers.
- *BpForms* can capture uncertainity in the structures of biopolymers. This is essential for proteomics.
- *BpForms* has a concrete grammar.
- All of the monomeric forms in the *BpForms* RNA alphabet have defined structures, unlike several MODOMICS entries which have undefined *BASE* groups. This gaurantees that *BpForms* polymers specify concrete structures.
- The *BpForms* RNA alphabet consistenty represents only nucleotide monophosphates. This gaurantees that the monomeric forms are composable. In constract, MODOMICS includes both nucleosides and bases, which makes the composition of the MODOMICS monomeric forms ill-defined.
- The *BpForms* RNA alphabet encompasses monomeric forms from both MODOMICS and the RNA Modification Database. This adds two additional monomeric forms that are not present in MODOMICS.
- *BpForms* includes software for error checking descriptions of non-canonical biopolymers. This includes verifying that monomeric forms that only have left bonding sites (e.g. 5' caps) only appear at the first position and that monomeric forms that only have right bonding sites (e.g. 3' caps) only appear at the last position.
- *BpForms* includes software for interpreting descriptions of non-canonical biopolymers. This enables calculations of properties such as chemical formulae, masses, and charges, which is essential for modeling and other applications.

1.10.6 Comparison with the PDB Chemical Component Dictionary/Ligand

- The *BpForms* protein alphabet includes numerous additional monomeric forms from RESID.
- The *BpForms* protein alphabet consistently represents the C termini as COH. In constrast, the PDB CCD represents C termini as both COOH and COH.
- The composition of *BpForms* monomeric forms into sequences is well defined. The *BpForms* protein alphabet explicitly describes the location of the C and N termini of each monomeric form. This enables the *BpForms* software to verify that BpForms describe valid atomic structures. The semantics for combining PDB CCD monomeric forms into sequences is unclear. Consequently, sequences of PDB CCD monomeric forms cannot be validated. In particular, monomeric forms with multiple C and N termini have ambiguous bonding becuase the PDB CCD monomeric forms do not capture the left and right bonding sites.

1.10.7 Comparison with PDB format

- *BpForms* is a more compact, human readable description of the primary structure of biopolymers.
- It is easier to compose monomeric forms into *BpForms* than the PDB format.
- *BpForms* is easier to embed into other standards and formats.

1.10.8 Comparison with PDB SEQRES annotations

- *BpForms* can capture any monomeric form, including monomeric forms which are not part of the PDB Chemical Component Dictionary.

- *BpForms* concretely captures the bonds between successive monomeric forms. PDB SEQRES annotations are ambiguous for monomeric forms with multiple C and N termini.
- *BpForms* can capture circularity.
- *BpForms* can capture crosslinks.
- There is no defined semantics for generating atomic structures from PDB SEQRES annotations.
- The *BpForms* software can verify that *BpForms* describe valid atomic structures. PDB SEQRES annotations cannot be verified because there is no defined semantics for generating atomic structures from these annotations.

1.10.9 Comparison with ProForma Proteoform Notation

- *BpForms* can represent DNA, RNA, and proteins.
- *BpForms* can represent any modification and, therefore, is not limited to modifications that have been previously enumerated in databases and ontologies. This is necessary to represent the combinatorial complexity of non-canonical DNA, RNA, and proteins.
- *BpForms* concretely captures the bonds between adjacent monomeric forms, avoiding ambiguity about how monomeric forms are composed into polymers. This is particularly important for monomeric forms that have multiple C and N termini, which affects numerous entries in RESID.
- *BpForms* can represent monomeric forms which can only bond to the right and left or which don't have backbones such as 3' and 5' caps.
- *BpForms* can represent bonds between non-adjacent monomeric forms, such as disulfide bonds.
- *BpForms* can represent circular biopolymers.
- *BpForms* separates the representation of non-canonical biopolymers from the chemical processes which generate them.
- *BpForms* can capture additional uncertainty in the structures of biopolymers: uncertainty in the position of a non-canonical monomeric form within a sequence, and uncertainty in the chemical identity of a non-canonical monomeric form (e.g., deviation from its expected mass or charge).
- *BpForms* has a concrete grammar.
- *BpForms* includes software for error checking descriptions of non-canonical biopolymers.
- *BpForms* includes software for interpreting descriptions of non-canonical biopolymers. This enables calculations of properties such as chemical formulae, masses, and charges, which is essential for modeling and other applications.

1.10.10 Comparison with RESID

- Each monomeric form in the *BpForms* protein alphabet has a defined structure. This gaurantees that polymers have well-defined structures. In constrast, RESID has numerous entires without defined structures.
- The composability of the monomeric forms in the *BpForms* protein alphabet is well-defined. Each form has at most one left-bonding-terminus (C) and at most one right-bonding-terminus (N). This eliminates confusion about the meaning of composition monomeric forms with multiple N and C-termini. In contrast, RESID has numerous entries with multiple N or C-termini whose composition into polymers is ill-defined.
- The *BpForms* protein alphabet encompasses entries from additional databases.

1.10.11 Comparison with the RNA Modification Database

- Each monomeric form in the *BpForms* protein alphabet has a machine-readable structure. This gaurantees that polymers have well-defined structures. In constrast, RESID has numerous entires without defined structures. In contrast, the RNA Modification Database only provides images and CAS ids, neither or which can easily be converted into SMILES.

1.10.12 Comparison with the Synthetic Biology Open Language (SBOL)

- *BpForms* concretely captures the primary structure of non-canonical biopolymers. In particular, it concretely captures the covalent bonds between monomeric forms. In contrast, SBOL's sequence annotations capture insufficient information to define the primary structure of a non-canonical biopolymer. The chemical meaning of these sequence annotations are unclear.
- *BpForms* directly captures the primary structure of biopolymers. In constract, SBOL indirectly captures structures via the reactions that produce them via operations such as cutting.
- *BpForms* can capture uncertainity in the structures of biopolymers. This is essential for proteomics.
- *BpForms* is easier to embed into other files such as SBML-encoded models.
- *BpForms* includes software for interpreting descriptions of non-canonical biopolymers. This enables calculations of properties such as chemical formulae, masses, and charges, which is essential for modeling and other applications.

1.10.13 Comparison with the World-wide Monomer Reference Database

- The *BpForms* alphabets include many more residues.
- The residues in the *BpForms* alphabets include synonyms, comments, and unification links with other databases.

1.11 Limitations and alternatives

BpForms has a few known limitations. Here we describe these limitations and, where possible suggest alternative solutions.

- *BpForms* cannot represent the secondary structure (e.g., base pairing) of biopolymers. Extending *BpForms* to represent secondary structure would enable *BpForms* to represent more DNA forms involved in DNA damage and repair.
- *BpForms* cannot represent branched biopolymers such as carbohydrates. Other formats such as WURCS are better suited for this purpose. See Glycopedia for more information.
- *BpForms* cannot represent the three-dimensional structure of a polymer. The PDB format is better suited for this purpose.

1.12 Contributing to *BpForms*

We welcome contributions to *BpForms*, including to the software, alphabets of residues, and ontology of crosslinks. Please use GitHub pull requests to contribute to *BpForms* or contact us by email.

1. Create a fork of the *BpForms* Git repository. Please see the GitHub documentation for more information.
2. Edit the code and/or alphabets.

3. Commit your changes to your fork of the *BpForms* repository.
4. Push your changes to GitHub.
5. Use the GitHub website to create a pull request for your changes. Please see the GitHub documentation for more information.

1.13 About

1.13.1 License

The software is released under the MIT license

1.13.2 Development team

This package was developed by the Karr Lab at the Icahn School of Medicine at Mount Sinai in New York, USA.

- Yassmine Chebaro
- Jonathan Karr
- Paul Lang
- John Sekar

1.13.3 Citing BpForms

Lang PF, Chebaro Y & Jonathan R. Karr. BpForms: a toolkit for concretely describing modified DNA, RNA and proteins. arXiv:1903.10042.

1.13.4 Acknowledgements

This work was supported by a National Institute of Health P41 award [grant number 1 P41 EB023912] and a National Science Foundation INSPIRE award [grant number 1649014].

1.13.5 Questions and comments

Please contact the Karr Lab with any questions or comments.