simd-dna

Release 2021

Aaron Ong, David Doty

Mar 11, 2023

CONTENTS:

`simd-dna` is a simulator for the SIMD||DNA model.

https://github.com/UC-Davis-molecular-computing/simd-dna

Table of Contents

CHAPTER

ONE

SIMD-DNA.CLASSES MODULE

class simd_dna.classes.**Cell**(*domains: List[str]*, *strand_labels: Optional[List[Dict]] = None*)

This is a representation of a cell in the SIMD||DNA model. A cell is a unit of data in the register, and is further subdivided into domains, which consist of a small number of nucleotides.

Parameters

- **domains** – A list of strings representing the domains of the cell in left to right order
- **strand_labels** – A list of dictionaries that map strand patterns to string labels. A string label will be written underneath the cell in the SVG drawing if the strand pattern matches. Each dictionary has the following key-value pairs:

  strands: A 2D list, where the first index is an integer that represents the start index of a strand relative to the first domain of the cell, starting at 0. The second index is the string name of the strand type that should be present at that index for the pattern to match.

  label: A string that will be printed underneath the cell in the SVG if the strand pattern in 'strands' matches the cell's current contents.

  One example is the following dictionary:

  ```
  {'strands':  [[0, 'Zero-first'], [3, 'Zero-second']], 'label':  '0'}
  ```

  This means that if a strand of type 'Zero-first' has its leftmost domain attached to domain 0 of the cell, and a strand of type 'Zero-second' has its leftmost domain attached to domain 3 of the cell, then the label string '0' will be written underneath that cell in the SVG drawing.

add_strand_label(*coordinate_strand_pairs: List*, *string_label: str*) → None

Adds a new strand label to the cell type. See simd_dna.classes.Cell for a detailed breakdown of the strand label's data structure.

Parameters

- **coordinate_strand_pairs** – A list, where the first item is an integer that represents the start index of a strand relative to the first domain of the cell, starting at 0. The second item is the string name of the strand type that should be present at that index for the pattern to match.
- **string_label** – A string that will be printed underneath the cell in the SVG if the strand pattern in 'strands' matches the cell's current contents.

static decode_json(*domains: List[str]*, *strand_labels: Optional[List[Dict]] = None*, ***kwargs*) → Cell

Decodes a JSON object and returns an instance of simd_dna.classes.Cell .

Parameters

- **domains** – A list of strings corresponding to the domains field

- **strand_labels** – A list of dictionaries corresponding to the strand_labels field
- **kwargs** – kwargs is placed to avoid throwing errors in the decode step if excess data is present in the JSON object. Any excess data is ignored.

Returns
A `simd_dna.classes.Cell` object

class simd_dna.classes.**ObjectEncoder**(*, *skipkeys=False*, *ensure_ascii=True*, *check_circular=True*, *allow_nan=True*, *sort_keys=False*, *indent=None*, *separators=None*, *default=None*)

A JSONEncoder subclass that allows the json.dump() function to get the dictionary encoding of an object

default(*o*)

Implement this method in a subclass such that it returns a serializable object for `o`, or calls the base implementation (to raise a `TypeError`).

For example, to support arbitrary iterators, you could implement default like this:

```
def default(self, o):
    try:
        iterable = iter(o)
    except TypeError:
        pass
    else:
        return list(iterable)
    # Let the base class default method raise the TypeError
    return JSONEncoder.default(self, o)
```

class simd_dna.classes.**Register**(*cell_types: Optional[Dict[str]] = None*, *strand_types: Optional[Dict[str]] = None*)

This is a representation of a register in the SIMD||DNA model. A register, in practice, is a long DNA strand attached to a magnetic bead. This long DNA strand is referred to as the "bottom strand", and information is stored and encoded through nick patterns in the attached "top strands." Instruction strands are applied to a register, where the top strands are altered through DNA strand displacement. Waste products are washed away before the next instruction strands are applied.

Parameters

- **cell_types** – A dictionary of `simd_dna.classes.Cell` instances representing the possible cell types that can be part of this `simd_dna.classes.Register` instance. The dictionary maps strings, which represent the cell name, to the actual `simd_dna.classes.Cell` instance.
- **strand_types** – A dictionary of `simd_dna.classes.Strand` instances representing the possible strand types that can be part of this `simd_dna.classes.Register` instance. The dictionary maps strings, which represent the strand name, to the actual `simd_dna.classes.Strand` instance.

Variables

- **cells** (*List[str]*) – A list of `simd_dna.classes.Cell` type names that represent the cells that compose this `simd_dna.classes.Register`
- **top_strands** (*List[simd_dna.classes.TopStrand]*) – A list of `simd_dna.classes.TopStrand` s present on the register.
- **total_domains** (*int*) – The total number of domains in the register's bottom strand

add_cell(*cell_name: str*) → None

Adds a cell to the right of the register.

Parameters

cell_name – A string representing the cell type to be added

attempt_attachment(*domain_index: int*, *strand_type: str*, *unattached_matches: Optional[List[TopStrand]] = None*) → Optional[List[TopStrand]]

Attempts to attach a copy of strand_type, with its leftmost domain placed on top of the specified domain_index if it's complementary to the bottom strand (strand_type.is_complementary is False.) If complementary to the top strand, detaches all top strands that bind to strand_type from the register.

Parameters

- **domain_index** – The integer index of the register domain that the leftmost domain of strand_type will attempt to attach to (e.g. attach strand 'one_first' starting at domain index 56.) If strand_type is complementary to the top strand, this parameter is ignored.
- **strand_type** – The name of the `simd_dna.classes.Strand` that will be attached. The name must be one of the keys in the Register's strand_types instance variable
- **unattached_matches** – A list of `simd_dna.classes.TopStrand` instances that complement the domains underneath, but are inert because no open toeholds are available. If the current strand to be attached matches but is inert, it will be added to this list. If the caller isn't interested in getting the location of inert instruction strands, None can be provided.

Returns

A list containing `simd_dna.classes.TopStrand` instances of new strands if attachment was successful, or None if no strands attached

static decode_json(*cell_types: List[Cell]*, *strand_types: List[Strand]*, *cells*, ***kwargs*) → Register

Decodes a JSON object and returns an instance of `simd_dna.classes.Register`

Parameters

- **cell_types** – A list of `simd_dna.classes.Cell` types that can be present on this register
- **strand_types** – A list of `simd_dna.classes.Strand` types that can be present on this register
- **cells** – A string list of cell names that compose this Register instance, going from left to right as the list index increases
- **kwargs** – kwargs is placed to avoid throwing errors in the decode step if excess data is present in the JSON object. Any excess data is ignored.

Returns

A `simd_dna.classes.Register` object

displace_strands(*excluded_strands: Optional[List[TopStrand]] = None*) → List[TopStrand]

Simulates DNA strand displacement on the register, initiated by the new strands introduced by `simd_dna.classes.Register.attempt_attachment()`. By first attaching all possible new strands before displacing existing strands on the register, cooperative strand displacement can be simulated.

Parameters

excluded_strands – A list of `simd_dna.classes.TopStrand` s that should be exempt from displacement

Returns

A list of `simd_dna.classes.TopStrand` s that were displaced

get_cell_at_domain_index(*domain_index: int*) → Tuple[Optional[str], int]

Returns the name of the cell type at the given domain index, starting from index 0, as well as the numerical offset relative to the beginning of the enclosing cell, starting at 0. For example, if a register has cells of type A, B, C in that order, where each cell type has 3 domains, then domains 0-2 will return A, 3-5 will return B, and 6-8 will return C. The domain at index 3 is the 0th domain in cell B, so an offset of 0 will be returned.

Parameters

domain_index – The integer index of the domain in the register.

Returns

A tuple containing a string that represents the cell type name (or None if the domain index exceeds the total domain length of the register), and the integer offset of that domain relative to the start index of its enclosing cell (0 if the domain index exceeds the total domain length.)

get_top_strands_at_domain_index(*domain_index: int*, *include_orthogonal: bool = False*, *strand_set: Optional[List[TopStrand]] = None*) → Union[Tuple[List[TopStrand], List[TopStrand]], List[TopStrand]]

Returns the DNA top strand(s) present at a given domain index.

Parameters

- **domain_index** – The integer index of the domain in the register.
- **include_orthogonal** – A boolean specifying whether a separate list of DNA strands that are orthogonal at this domain index should be returned.
- **strand_set** – A list of DNA top strands to be inspected. The register's top_strands instance variable will be used if None.

Returns

The list of DNA top strands attached to the provided domain index, and the list of DNA top strands with orthogonal domains hanging above the provided domain index if include_orthogonal is set to true

print(*new_strands: Optional[List[TopStrand]] = None*, *unused_strands: Optional[List[TopStrand]] = None*) → None

Prints the register's current contents on the terminal.

Parameters

- **new_strands** – A list of `simd_dna.classes.TopStrand` s that will displace the current strands, which will be printed one level above the current strands on the register.
- **unused_strands** – A list of `simd_dna.classes.TopStrand` s that would've attached to the register but are inert, which will be printed two levels above the current strands on the register.

sanitize_inert_strands(*inert_strands: List[TopStrand]*, *new_strands: List[TopStrand]*) → List[TopStrand]

Removes inert strands that overlap with other inert strands or newly attached strands. Two strands are said to overlap if the have complementary domains in common on the register. This is done to reduce the cluttering in the visual representation of the register, in case the user chooses to print the register on the console or display its contents through an SVG representation.

Parameters

- **inert_strands** – A list of inert `simd_dna.classes.TopStrand` s after applying an instruction

- **new_strands** – A list of newly attached simd_dna.classes.TopStrand s after applying an instruction

Returns
A list of inert simd_dna.classes.TopStrand s after overlapping strands are removed

strands_intersect(*strand_1: TopStrand*, *strand_2: TopStrand*) → bool

Checks if two strands occupy the same domain location(s), where their domain segments at that location are complementary to the bottom strand. The two strands compete over that domain(s) if so.

Parameters

- **strand_1** – The first simd_dna.classes.TopStrand to compare
- **strand_2** – The second simd_dna.classes.TopStrand to compare

Returns
True if the strands intersect, False otherwise

class simd_dna.classes.**Strand**(*domains: List[str]*, *is_complementary: bool*, *color: str = '#000000'*)

This is a representation of a DNA strand in the SIMD||DNA model.

Parameters

- **domains** – A list of strings representing the domains of the strand in left to right order
- **is_complementary** – A boolean that indicates whether the strand is complementary to the top strand of the register or not. A top complementary strand in the SIMD||DNA model has the 3' end on the left and the 5' end on the right.
- **color** – A hexadecimal string that represents the strand's color when drawn in an SVG file

static decode_json(*domains: List[str]*, *is_complementary: bool*, *color: str = '#000000'*, ***kwargs*) → Strand

Decodes a JSON object and returns an instance of simd_dna.classes.Strand

Parameters

- **domains** – A list of strings corresponding to the domains field
- **is_complementary** – A boolean corresponding to the is_complementary field
- **color** – A string corresponding to the color field
- **kwargs** – kwargs is placed to avoid throwing errors in the decode step if excess data is present in the JSON object. Any excess data is ignored.

Returns
A simd_dna.classes.Strand object

class simd_dna.classes.**TopStrand**(*start_index: int*, *strand_name: str*)

This is a representation of a top strand currently attached to a register.

Attributes:

start_index: The index of the strand's leftmost domain on the register, starting from the register's leftmost domain at index 0

strand_name: The name of the strand at start_index's location

static decode_json(*start_index: int*, *strand_name: str*, ***kwargs*) → TopStrand

Decodes a JSON object and returns an instance of :class:simd_dna.classes.TopStrand

Parameters

- **start_index** – An integer representing the register position of the top strand's leftmost domain
- **strand_name** – A string representing the name/type of the top strand
- **kwargs** – kwargs is placed to avoid throwing errors in the decode step if excess data is present in the JSON object. Any excess data is ignored.

Returns

A `simd_dna.classes.TopStrand` object

CHAPTER

TWO

SIMD-DNA.SIMULATION MODULE

class simd_dna.simulation.**Simulation**(*step_by_step_simulation: bool = False*, *keep_results: bool = False*, *show_inert_instruction_strands: bool = False*)

This is an object that contains all settings and methods used in the SIMD||DNA simulation

Parameters

- **step_by_step_simulation** – A boolean that causes the console program to print each instruction one at a time if set to True. Otherwise, all instructions are printed at once in each cycle.
- **keep_results** – If set to True, the results after applying all instructions will overwrite the original contents of the register. Otherwise, the results will be discarded. Discarding the results can be helpful when testing out new instructions.
- **show_inert_instruction_strands** – If set to True, inert instruction strands will be printed on the terminal, alongside the applicable instructions. Otherwise, only applicable instruction strands are shown.

Variables

- **strand_types** (*Mapping[str, Strand]*) – A dict of possible strand types in the simulation, mapping the strand name to the `simd_dna.classes.Strand` instance.
- **cell_types** (*Mapping[str, Cell]*) – A dict of possible cell types in the simulation, mapping the cell name to the `simd_dna.classes.Cell` instance.
- **registers** (*Mapping[str, Register]*) – A dict of registers present in the solution, mapping the register name to the `simd_dna.classes.Register` instance.
- **instructions** (*List[List[str]]*) – A list of instructions to be applied. Each instruction is a list of strings, which are the names of strand types present in that instruction. The instructions are applied in the order of their indices, starting from 0.

add_cell_strand_label(*cell_name: str*, *coordinate_strand_pairs: List[List]*, *string_label: str*) → None

Adds a new strand label to a cell. See `simd_dna.classes.Cell` for a detailed description of strand labels.

Parameters

- **cell_name** – The name of the cell to add a strand label to.
- **coordinate_strand_pairs** – A list of pairs of coordinates and strand names, representing the locations of the leftmost domains of each strand.
- **string_label** – A string that will be printed underneath the cell in the SVG drawing if the top strands match the coordinates and strands indicated.

add_cell_type(*name: str*, *domains: List[str]*) → None

Adds a new `simd_dna.classes.Cell` type to the simulation.

Parameters

- **name** – The name of the new cell type.
- **domains** – A list of strings representing the names of the domains that comprise the new cell type.

add_cells_to_register(*register_name: str*, *cell_name: str*, *top_strands: Optional[List[TopStrand]] = None*, *copies: int = 1*) → None

Adds new cells to a register, starting from the right side of the register (the bottom strand's 5' end.)

Parameters

- **register_name** – The name of the `simd_dna.classes.Register` to add cells to. If the register doesn't already exist in the simulation, a new one is created.
- **cell_name** – The name of the cell type to be added to the register.
- **top_strands** – A list of top strands present on the cells to be added. If None, the added cells will be bare.
- **copies** – An integer representing the number of cell copies to be added.

add_instruction(*instruction_strands: List[str]*) → None

Adds a new instruction to the simulation.

Parameters

instruction_strands – A list of strings representing the strand names present in the instruction.

add_strand_type(*name: str*, *domains: List[str]*, *is_complementary: bool = False*, *color: str = '#000000'*) → None

Adds a new `simd_dna.classes.Strand` type to the simulation.

Parameters

- **name** – The name of the new strand type.
- **domains** – A list of strings representing the names of the domains that comprise the new strand type.
- **is_complementary** – A boolean determining if the strand is complementary to the top strand. If True, the strand is designed to bind to and remove top strands from the register, instead of binding to the register's bottom strand.
- **color** – A string hexadecimal color code representing the color of this strand when drawn in SVG.

run_instruction(*register_name: str*, *inst_num: int*) → Tuple[Register, Register, List[TopStrand], Optional[List[TopStrand]]]

Applies an instruction to a register.

Parameters

- **register_name** – The name of the affected `simd_dna.classes.Register`
- **inst_num** – The integer index of the applicable instruction.

Returns

A tuple describing the results of applying the instruction: the `simd_dna.classes.Register` after applying the instruction, a copy of the `simd_dna.classes.Register` before the instruction, the list of applicable instruction strands, and (optionally) the list of inert instruction strands. The function will not keep track of inert instruction strands if show_inert_instruction_strands is set to False.

CHAPTER

THREE

SIMD-DNA.REGISTER_SVG MODULE

CHAPTER

FOUR

SIMD-DNA.TM MODULE

CHAPTER

FIVE

SIMD-DNA.FUNCTIONS MODULE

simd_dna.functions.**convert_hex_to_rgb**(*hex_rgb: str*) → str

Returns an RGB string representation of a hexadecimal color code

Parameters

hex_rgb – A 6-digit hex color code

Returns

An RGB string representation of the provided hex color code

simd_dna.functions.**convert_rgb_to_hex**(*r: float*, *g: float*, *b: float*) → str

Returns a hexadecimal color code representation of a set of RGB values

Parameters

- **r** – A float representing the red component
- **g** – A float representing the green component
- **b** – A float representing the blue component

Returns

A 6-digit hex color code representing the provided RGB values

CHAPTER

SIX

INDICES AND TABLES

- genindex
- modindex
- search

INDEX