
Gnomic Documentation

Release 1.0.1

Lars Schöning

Sep 21, 2017

Contents

1	Example usage	3
2	User's guide	5
2.1	Language grammar	5
2.2	Gnomic API	6
3	Indices and tables	7
	Python Module Index	9

Gnomic is a human- and computer-readable representation of microbial genotypes and phenotypes. The `gnomic` Python package contains a parser for the Gnomic grammar able to interpret changes over multiple generations.

The first formal guidelines for microbial genetic nomenclature were drawn up in the 1960s. These traditional nomenclatures are too ambiguous to be useful for modern computer-assisted genome engineering. The *gnomic* grammar is an improvement over existing nomenclatures designed to be clear, unambiguous and computer-readable and describe genotypes at various levels of detail.

A JavaScript (Node) version of the package is available on NPM as [gnomic-grammar](#).

CHAPTER 1

Example usage

In this example, we parse “*EcGeneA ΔsiteA::promoterB:EcGeneB ΔgeneC*” and “*ΔgeneA*” in *gnomic* syntax:

```
>>> from gnomic import Genotype
>>> g1 = Genotype.parse('+Ec/geneA(variant) siteA>P.promoterB:Ec/geneB -geneC')
>>> g1.added_features
{Feature(organism=Organism('Ec'), name='geneA', variant=('variant',)),
 Feature(organism=Organism('Ec'), name='geneB'),
 Feature(type='P', name='promoterB')}
>>> g1.removed_features
{Feature(name='geneC'),
 Feature(name='siteA')}

>>> g2 = Genotype.parse('-geneA', parent=g1)
>>> g2.added_features
{Feature(type='P', name='promoterB'),
 Feature(name='geneB', organism='Ec')}
>>> g2.removed_features
{Feature(name='siteA'),
 Feature(name='geneC')}
>>> g2.changes()
(Change(multiple=False,
        after=Fusion(annotations=(Feature(type='P', name='promoterB'),
↪Feature(organism='Ec', name='geneB'))),
        before=Feature(name='siteA')),
 Change(multiple=False, before=Feature(name='geneC')))
```

```
>>> g2.format()
'ΔsiteA P.promoterB:Ec/geneB ΔgeneC'
```


Language grammar

The grammar consists of a list of genotype or phenotype designations, separated by spaces and/or commas. The designations are described using the following nomenclature:

Designation	Grammar expression
feature deleted	-feature
feature at locus deleted	-feature@locus
feature inserted	+feature
site replaced with feature	site>feature
site (multiple integration) replaced with feature	site>>feature
site at locus replaced with feature	site@locus>feature
feature of organism	organism/feature
feature with type	type.feature
feature with variant	feature(variant)
feature with list of variants	feature(var1, var2) or feature(var1; var2)
feature with accession number	feature#GB:123456
feature by accession number	#GB:123456
accession number	#database:id or #id
fusion of feature1 and feature2	feature1:feature2
insertion of two fused features	+feature1:feature2
insertion of a list of features or fusions	+{..insertables}
fusion of a list and a feature	{..insertables}:feature
a non-integrated plasmid	(plasmid) or (plasmid ..insertables)
integrated plasmid vector with required insertion site	site>(vector ..insertables)

Gnomic API

Genotype

`class gnomic.Genotype` (*changes*, *parent=None*)

`classmethod is_valid` (*gnomic_string*, ***kwargs*)
Tests whether a gnomic genotype definition can be parsed.

Features

`class gnomic.Feature` (*name=None*, *type=None*, *accession=None*, *organism=None*, *variant=None*)
-Feature(“foo”) Feature(“site”) >> Feature(“insertion”)

`class gnomic.Plasmid` (*name*, *annotations=()*)

`class gnomic.Fusion` (**annotations*)

CHAPTER 3

Indices and tables

- `genindex`
- `modindex`
- `search`

g

gnomic, [6](#)

F

Feature (class in `gnomic`), [6](#)

Fusion (class in `gnomic`), [6](#)

G

Genotype (class in `gnomic`), [6](#)

`gnomic` (module), [6](#)

I

`is_valid()` (`gnomic.Genotype` class method), [6](#)

P

Plasmid (class in `gnomic`), [6](#)