# Content

# [Logic axioms in DNA sequences from Below](#A)

# [Ontological axioms/rules in genome, gene, interval and ontology](#B)

# [Allen’s calculus for genetic interval relations updated](#C)

# Axioms in [DNA Sequences from Below: A Nominalist Approach](https://github.com/linikujp/ontology-for-genetic-interval/blob/master/publications/DNASequencesfromBelow.pdf)

# Nominalism about the real sequence

To give a formal account of the real sequence, we start from the ideas of element and sequence rather than element-kind and sequence-kind.

We assume there are some individuals which (because they are basic for genomic purposes) we call *elements*, and that each element belongs to exactly one element-kind, so that no two element-kinds have any element in common. Two element-kinds are called *coextensional* when the same elements belong to them. Element-kinds with at least one element in common are therefore coextensional.

Some elements occur in what we may call sequences. A sequence consists literally of several elements linked together one after the other. How they are linked is not part of the abstract theory except that the linking is assumed to be real (physical) and asymmetric, i.e. if one element **A** is linked to another **B**, the short sequence **AB** is such that there can be no sequence **BA** of the elements in the opposite order (at the same time). How things turn out later if the elements of a sequence are rearranged is a different matter.

We assume without further justification that the physical links between elements can all be treated as of the same kind. This may be a simplification in some cases but for the purposes of defining sequence we overlook it. In the case of DNA strands the linking is provided by bonding between carbon atoms in the phosphate deoxyribose backbone. The directionality is given by the difference between the 5' and 3' positions of carbon in a deoxyribose ring, giving rise to the upstream/ downstream asymmetry in DNA.

## A finite whole W of elements E

We define a whole of several elements **E** as an object which

1. has every element of **E** as its part
2. has no other types of elements than **E** as its parts

It may or may not have non-element parts such as parts of the backbone which are not bases in DNA. For a definition of sequence we do not need to take these into account.

A finite whole **W** of elements **E** is one where there are finitely many elements in **E**. In all practical applications this will be the case. Infinite collections may interest mathematicians but not real scientists.

Let **W** be a whole of elements.

Let **El(W)** be the collection of elements,

**W** is a single individual of which all elements of **El(W)** are parts. If an individual **A** is one of the elements of **W** we write:

**A** is one of **El(W)**.

The relation between **A** and **W** thereby defined is the element-of relation:

**A** is an element of **W.**

Note that if **A** is an element of **W** implies that **A** is a part of **W**. But not necessarily *vice versa*: **W** typically has other parts than its elements.

## Definition of L-string

We say **W** is a *string* of its elements **E** iff (Def.) there is a relation defined on the elements **E** which is connected, asymmetric, bi- directionally unique or non-splitting, and has two exactly free ends.

## Linking relation L

Let **A** and **B** both be elements of **W**:

If **A** links to **B**, we write ‘**ALB**’ .

We are using the schematic letter ‘**L**’ for such a linking relation. What the link consists of in any concrete case may vary. In the case of a DNA sequence it is the usual phosphodiester bonds between the fifth and third carbon atoms of adjacent sugar rings in the molecule. In the conventional notation for DNA the “upstream” base is written to the left of the “downstream” base and we are reflecting this in our choice of notation. We also stipulate that if **ALB** we may say that **A** is (immediately) to the left of **B** and **B** is (immediately) to the right of **A**. The use of ‘left’ and ‘right’ is here merely conventional and corresponds to no actual left–right orientation (which is in any case observer- and orientation-relative).

**L** is assumed to be asymmetrical. Ignoring the temporal aspect, the axiom of Asymmetry says:

Asym. For all **x** and **y** in **E**: if **xLy** then not **yLx** . (**0**) Corollary:

For no **x** in **E** is **xLx**.

Above axioms ensure the local direction of a sequence is from **x** to **y**.

If **ALB** then there exists a sequence of **A** linked immediately to **B** which we write ‘**AB**’. **A** and **B** are both parts of **AB** and neither is

identical to **AB**: they are proper parts. The asymmetry tells us that If **AB** exists **BA** does not exist .

We may extend the idea of linking from single elements to longer sequences of elements **E,** the idea being that one sequence **S** is **L**- linked to another **T** when the rightmost element of **S** is **L**-linked to the leftmost element of T. This explains what happens when we “paste’” together short subsequences resulting from modern sequencing methods.

## Non-splitting or branching

Non-branching sequence from the left:

LUn. For all **x**, **y** and **z** in **E**: If **xLy** and **zLy** then **x** = **z** . (**2**) Non-branching sequence from the right:

RUn. For all **x**, **y** and **z** in **E**: If **xLy** and **xLz** then **y** = **z** . (**3**)

## Two free ends

A sequence is left-ending:

LEnd**.** For some **x** in **E** there is no **y** in **E** such that **yLx** . (**4**) A sequence is right-ending:

REnd. For some **x** in **E** there is no **y** in **E** such that **xLy** . (**5**)

## Ancestral of L-relation

To exclude other cases such as two unconnected strings, or one string and one cycle in **W**, we require that the string be *connected,* that is, that it be possible to pass from any element to any other element by a finite sequence of steps in the **L**-direction or in the direction opposite to **L**.

For this purpose, we start from a definition of the *ancestral* of the **L**- relation.

Intuitively, **A** stands in the ancestral of the **L**-relation to **B** if **ALB** or **ALx** and **xLB** for some **x** or **ALx** and **xLy** and **yLB** for some **x** and **y** … and so on. Thus, we get from **A** to **B** by doing one **L**-step after another.

To define ancestral we start by taking **C** to be a collection of elements.

**C** is **L**-hereditary: Her(**L**)(**C**) iff (Def.) for all **x** and y: if x is one of C and **xLy** then y is one of **C**.

So **C** collects up all the things lying **L**-downstream of **x**. We now define the ancestral **L\*** of **L** as follows:

For all **x** and **y**: **xL\*y** iff (Def.) for all **C**, if **x** is one of **C** and Her(**L**)(**C**) then **y** is one of **C**.

This takes all the **L**-hereditary collections and extracts the smallest or minimal one, since an arbitrary **L**-hereditary collection may contain extraneous junk. If **xL\*y** then **y** is **L**-downstream of **x** by a finite number of steps.

**L**-Conn. A collection **C** of elements is **L**-connected iff (Def.)

for all **x** and **y** in **C**, either (i) **x** = **y** or (ii) **xL\*y** or (iii) **yL\*x**. (**5**)

A whole made of an **L**-connected collection of elements may not contain two disconnected sub-collections, but it could be cyclic.

Therefore, we give a final definition for an **L**-string object:

An **L**-*string* of elements **E** is a finite whole made of elements linked by the **L**-relation such that it satisfies Asym, LUn, RUn, LEnd, REnd and L-Conn.

## Substring and L-strand

One string is a substring of another under the following conditions.

Let **S** be a string of elements **E** under linking relation **L** and **T** be a string of elements **F** under linking relation **M**.

We say **S** is a substring of **T** iff (Def.)

1. **S** is a string of **E** under **L**
2. **T** is a string of **F** under **M**
3. **L** = **M**
4. **E** is a sub-collection of **F**, i.e. every one of **E** is one of **F**.

We now define an **L-strand** of elements **E** as a maximal L-string.

Let S be an L-string of elements **E.** Then **S** is an **L-strand** iff (Def.) there is no **T** and no **F** such that **T** is an L-string of elements **F** and **F** contains all elements of **E** and **S** is a substring of T.

We then define an **L**-strand having two ends:

Let **S** be an **L**-string, then **L**-top and **L**-bottom elements of **S** are the

two ends respectively:

**L**Top(**S**) = that element **x** of **S** such that for no **y** in S is **yLx L**Bot(**S**) = that element **y** of **S** such that for no **x** in **S** is **yLx**

## Sequences or string-kinds

We assume every element **A** belongs to at least one element-kind **EK(A)**, and that if two element kinds **K** and **H** have any element in common, they are one and the same element-kind. Therefore, distinct element-kinds are disjoint, i.e. have no common members (for if they did, some element would belong to two distinct kinds). Thus, every element belongs to exactly one element-kind.

Let **A** and **B** be elements. **A** and **B** are *isogenic* iff they belong to the same element-kind.

Let **S** and **T** be two **L**-strings. It is supposed that they are equally long,

i.e. the collection **El(S)** and **El(T)** have the same number of members. We then define a correspondence relation **L***-corr* between elements of the strings as follows:

1. **L**Top(**S**) **L**-corresponds to **L**Top(**T**) and vice versa, and neither **L**- corresponds to anything else
2. For all **x** and **y** in S and for all **z** and **w** in **T**:

if **x L**-corresponds to **z** and **xLy** and **zLw** then **y L**-corresponds to w and vice versa and neither **L**-corresponds to anything else.

By this means we set up a one-one correspondence between elements of **S** and the elements of **T** that correspond to them in position down from the top. First in **S** corresponds to first in **T**, second in **S** to second in **T** and so on.

We now define *isotypy* between two **L**-strings.

Let **S** and **T** be two **L**-strings. Then **S** is **L**-*isotypic* with **T** iff (Def.)

1. **S** and **T** are equally long.
2. For all **x** in **S** and **y** in **T**, **x** is isogenic to **y** (they are of the same element-kind), and **x L**-corresponds to **y**

We may now say: two **L**-strings **S** and **T** exemplify the same string-type if and only if they are **L**-isotypic.

We may specify a string-type by giving:

* 1. The linking relation **L**
  2. The number of elements in any instance
  3. The element-kind of the **L**-top of each instance
  4. The element-kind of each subsequent element as we pass along the string, or equivalently

(4') For each position in the string-kind, the element-kind of the occupier of that position in any instance.

The positions in a string-kind derive from the number of **L**-steps that we go from the **L**-top of any string of that kind to the element that number of steps along. First, second, third and so on. These can be notated by the standard numerals.

## Pairs in a DNA molecule

The above notion of correspondence is not the same as that of pairing used in genetics. There we talk about the actually linking of distinct elements in the two strands of DNA that form up to make a double helix, such that G pairs with C and A with T. What this means is that not just any conceptually possible strand of A, C, G and T bases can be a DNA strand in a double helix DNA molecule, because the two strands have to be anti-parallel. That imposes a constraint on the sequence of bases, as follows:

Call paired element type *counterparts*: so A and T are each other's counterparts, and G and C are each other's counterparts.

A strand of bases is only a candidate for a double DNA strand if:

for all n ≥ 0: the position n **L**-steps *down* from **L**Top (first position, 5' end) of one strand is in its element-kind the counterpart of the position n steps *up* from **L**Bot (last position, 3' end) of the other strand.

We call such a sequence *Definitely Nicely Arranged*.

Two Definitely Nicely Arranged and isotypic strands can pair in an anti-parallel fashion to form a complete DNA molecule: any actual DNA molecule has two strands which are DNA (Definitely Nicely Arranged), isotypic, and do actually pair up by hydrogen bonds.

# Ontological definitions in [Genome, Gene, Interval and Ontology](https://github.com/linikujp/ontology-for-genetic-interval/blob/master/publications/genome_gene_interval_ontology_2009.pdf)

## Genetic Interval

In mathematics, an interval is a certain subset of an ordered set. Allen JF has devel- oped an Interval Temporal Logic [Allen et al. 1984, Allen et al. 1990] to present a concise, formal axiomatization of “interval-based” time. Hobbs, J. R. and Pan, F. de- veloped a time ontology to describe the temporal entity and relations for semantic web. [Hobbs and Pan 2004] In biological investigations, many genetic entities, such as genes, alleles, haplotypes and genetic markers are based on their sequence information, which is a physical subset of an ordered DNA or RNA base set. Based on their similarities, and the needed of co-localization of some genetic entities with genes, we presented genetic intervals as physical material intervals, which start and end at a certain point or boundary.

Here, we use a universal named “*Biological Interval*” to describe the spatial con- tinuous physical entity which contains ordered biological sets (DNA segments, Genetic Markers, Nucleic Acid Base Residues, RNA segments, Protein segments) between two boundaries: start boundary and end boundary on a chromosome, RNA or protein. *Bio- logical Interval* is distinct by three intervals: *Genetic Interval*, *Interval Base Residue* and *Protein Interval*; *Genetic Interval* is distinct by *DNA Interval* and *RNA Interval*. *DNA Interval* has primary DNA sequence structure by its definition; whereas RNA Interval has primary RNA sequence structure.

We discuss mainly *Genetic Interval* in this paper. *Genetic Interval* is continuous, so that neither a genome (a collective of chormosomes), nor a genotype of a diploid (a collective alleles coming from different chromosomes), nor a gene family (a collective of genes of same homolog located to different chromosomes) are *Genetic Interval*. Whereas gene clusters which are juxtaposition genes on chromosomes can be a sub- class of *Genetic Interval*. By length, the longest *DNA Interval* is the interval with the same start point and end point as the chromosome, and the smallest is when one start point and end point are equal (one residue of DNA or RNA).

By ontological definition, *Genetic Interval* is a *Biological Interval*, which hasEnd- Point *End\_Boundary\_of\_Interval* and hasEndPoint Exactly 1, and hasStartPoint *Start\_Boundary\_of\_Interval* and hasStartPoint Exactly 1, and hasIntervRelations with *Genetic Interval*, and is (*DNA Interval* or *RNA Interval*). The first fold subclasses of *DNA Interval* are: *DNA segment*, *Genetic Marker*, *Probe, Amplifier,* and *Flanking Sequences*. *DNA Segment* has such subclasses as: *Gene*, *Allele*, *Exon*, *Intron*, *Gene Regulatory Elements*, *Haplotype*, *Intergenic Segment*, *Linkage Interval*, *LD block* and so on. *DNA segment* is a *DNA Interval* that is located on a chromosome.

## Interval Relations

Based on the interval relations we will introduce later, the locus of a susceptibility gene or region in a chromosome can be inferred from the location of other genetic markers.

The following axiom states one of the object properties of Genetic\_Interval : hasIn- tervalRelation

<owl:ObjectProperty rdf:about=”#hasIntervalRelation”>

<rdfs:range rdf:resource=”#Genetic\_Interval”/>

<rdfs:domain rdf:resource=”#Genetic\_Interval”/>

</owl:ObjectProperty>

We use logic to define genetic interval and its relations:

*geneticInterval(X)* X is genetic interval

*hasStartPoint*(x1,X) X has start point x1

*hasEndPoint*(x2,X) X has end point x2

If the start and end of a genetic interval are identical, the genetic interval is a point interval.

*pointInterval*(X) ↔ *geneticInterval*(X) ∧ *hasStartPoint*(x,X) ∧ *hasEndPoint*(x,X)

The start point and end point is point interval. *pointInterval*(x1) ↔ *geneticInterval*(X) ∧ *hasStartPoint*(x1,X) *pointInterval*(x2) ↔ *geneticInterval*(X) ∧ *hasEndPoint*(x1,X)

### isLocatedBefore

isLocatedBefore is both a Class-Class and an Instance-Instance relation.

*isLocatedBefore*(X,Y) genetic interval X is located before genetic interval

Y

If genetic interval X is located before genetic interval Y, then the end of X is before

the start of Y.

*isLocatedBefore*(X,Y) ↔ *hasEndPoint*(x,X) ∧ *hasStartPoint*(y,Y) ∧ *isLocatedBe- fore*(x,y)

### isLocatedAfter

*isLocatedAfter*(X,Y) genetic interval X is located after genetic interval Y

If genetic interval X is located after genetic interval Y, then the start of X is after the end of Y; or the end of Y is before the start of X. If *isLocatedBefore*(X,Y), then *isLocatedAfter*(Y,X)

*isLocatedAfter*(X,Y) ↔ (*hasStartPoint*(x1,X) ∧ *hasEndPoint*(y1,Y) ∧ *isLocate- dAfter*(x1,y1))

∨(*hasEndPoint*(x2,X) ∧ *hasStartPoint*(y2,Y) ∧ *isLocatedBefore*(y2,x2)) *isLocatedAfter*(X,Y) ≡ *isLocatedBefore*(Y,X)

### isStartsWith

*isStartsWith*(X,Y) genetic interval X starts with genetic interval Y

If genetic interval X starts with genetic interval Y, then the start of X and the start of Y are identical and the end of X is before the end of Y.

*isStartsWith*(X,Y) ↔ ((*hasStartPoint*(x1,X) ∧ *hasStartPoint*(x1,Y))

∧ *(hasEndPoint(x2,X)* ∧ *hasEndPoint(y,Y)* ∧ *isLocatedBefore(x2,y))*

### isEndsWith

*isEndsWith*(X,Y) genetic interval X ends with genetic interval Y

If genetic interval X ends with genetic interval Y, then the end of X and the end of Y are identical and the start of X is after the start of Y.

*isEndsWith*(X,Y) ↔ ((*hasEndPoint*(x1,X) ∧ *hasEndPoint*(x1,Y))

∧ *(hasStartPoint(x2,X)* ∧ *hasStartPoint(y,Y)* ∧ *isLocatedAfter(x2,y))*

### isOverlapStartWith

*isOverlapStartWith*(X,Y) genetic interval X overlaps the start of genetic interval Y

If genetic interval X overlaps the start of genetic interval Y, then the start of X is be- fore the start of Y and the end of X is before the end of Y and the end of X is distinct from the start of Y.

*isOverlapStartWith*(X,Y) ↔ (*hasStartPoint*(x1,X) ∧ *hasStartPoint*(y1,Y) ∧ *isLo- catedBefore*(x1,y1))

∧ (*hasEndPoint*(x2,X) ∧ *hasEndPoint*(y2,Y) ∧ *isLocatedBefore*(x2,y2))

∧ ((*pointInterval*(x2,X) ≠ *pointInterval*(y1,X))

### isOverlapEndWith

*isOverlapEndWith*(X,Y) genetic interval X overlaps the end of genetic interval

Y

We can define isOverlapEndWith by isOverlapStartWith, if *isOverlapStart-*

*With*(X,Y) , then *isOverlapEndWith*(Y,X) *isOverlapStartWith*(X,Y) ↔ *isOverlapEndWith*(Y,X)

Another approach is straightforword. If genetic interval X overlaps the end of ge- netic interval Y, then the start of X is after the start of Y and the end of X is after end of Y and the start of X is distinct from the end of Y.

*isOverlapEndWith*(X,Y) ↔ (*hasStartPoint*(x1,X) ∧ *hasStartPoint*(y1,Y)

∧ *isLocatedAfter*(x1,y1)) ∧ (*hasEndPoint*(x2,X) ∧ *hasEndPoint*(y2,Y) ∧

*isLocatedAfter*(x2,y2))

∧ ((*pointInterval*(x1,X) ≠ *pointInterval*(y2,X))

### isContainedIn

*isContainedIn*(X,Y) genetic interval X is contained inside genetic interval Y

If genetic interval X is contained in genetic interval Y, then start of X is after the start of Y and the end of X is before the end of Y and the start of X is unique with the start of Y, the end of X is unique with the end of Y.

*isContainedIn*(X,Y)

*isContainedIn(X,Y) ↔ (hasStartPoint(x1,X)* ∧ *hasStartPoint(y1,Y)* ∧ *is- LocatedAfter(x1,y1))*

∧ (*hasEndPoint*(x2,X) ∧ *hasEndPoint*(y2,Y) ∧ *isLocatedBefore*(x2,y2))

∧ ((*pointInterval*(x1,X) ≠ *pointInterval*(y1,X))

∧ ((*pointInterval*(x2,X) ≠ *pointInterval*(y2,X))

### isAdjacentBefore

*isAdjacentBefore*(X,Y) genetic interval X is adjacent to the start of genetic interval Y

If genetic interval X is adjacent to the start of genetic interval Y, then the start of X is before the start of Y and the end of X is before the end of Y, and the end of X is identical with the start of Y.

*isAdjacentBefore*(X,Y) ↔ (*hasStartPoint*(x1,X) ∧ *hasStartPoint*(y1,Y) ∧ *isLocat- edBefore*(x1,y1))

∧ (*hasEndPoint*(x2,X) ∧ *hasEndPoint*(y2,Y) ∧ *isLocatedBefore*(x2,y2))

∧ ((*pointInterval*(x2,X) ≡ *pointInterval*(y1,X))

### isAdjacentAfter

*isAdjacentAfter*(X,Y) genetic interval X is adjacent the end part of genetic interval Y

If genetic interval X is adjacent to the end part of genetic interval Y, then the start of X is after the start of Y and the end of X is after the end of Y, and the end of X is the same as the start of Y.

*isAdjacentAfter*(X,Y) ↔ (*hasStartPoint*(x1,X) ∧ *hasStartPoint*(y1,Y) ∧ *isLocate- dAfter*(x1,y1))

∧ (*hasEndPoint*(x2,X) ∧ *hasEndPoint*(y2,Y) ∧ *isLocatedAfter*(x2,y2))

∧ ((*pointInterval*(x2,X) ≡ *pointInterval*(y1,X))

# Updated Allen’s calculus in [poster from OGI to gene assembly using NGS](https://github.com/linikujp/ontology-for-genetic-interval/blob/master/publications/Poster2_OGI_sequence_assembly.pdf)

Table 1. Genetic Interval Relations

|  |  |  |
| --- | --- | --- |
| Relations in Allen Interval | Illustration | Relations of Genetic Interval |
| X<Y Y>X |  | isLocatedBefore (xLBy) isLocatedAfter (yLAx) |
| XmY YmiX |  | isAdjacentBefore (xABy) isAdjacentAfter (yAAx) |
| XoY YoiX |  | isOverlapStartWith (xOSy) isOverlapEndWith (yOEx) |
| XsY YsiX |  | isStartsWith (xSWy,ySWx) (symmetric property) |
| XdY YdiX |  | isContainedIn (xCIy) (transitive property) |
| XfY YfiX |  | isEndWith (xEWy, yEWx) (symmetric property) |
| X=Y |  | isEqualTo (xEy, yEx) (symmetric property) |
|  |  | isReverseCompleteOf (xRCy) (symmetric property) |