The Specification of Signal Track Query Language

1 Data Model

An **interval** is a DNA segment. Formally, an interval is a complex data type, which contains several fields as below:

- chr: The name of the chromosome on which the DNA segment is. Its type is string. It is a compulsory field.
- chrstart: The starting position of the DNA segment (inclusive). Its type is integer. It is a compulsory field. The first base in a chromosome is numbered 1.
- chrend: The ending position of the DNA segment (inclusive). Its type is integer. It is a compulsory field. It must be no less than the starting position (i.e., chrend $\geq$ chrstart), and cannot be larger than the length of the chromosome. If chrend is equal to chrstart, the interval is a single-position interval.
- value: The value associated to the DNA segment. Its type is float. It could be NULL. It is a compulsory field.
- strand: The strand on which the DNA segment is. Valid entries include "+", "−" or ".", meaning the positive strand, negative strand, do not know or do not care which strand, respectively. Its type is string. It is a optional field.
- other meta-data: other optional meta-data fields the DNA segment has. For example, a feature has a field called source showing the program that generated the feature. A DNA segment may has a name, such as a gene.

We define two value models to manipulate intervals' values: ***each*** and ***total***:

- each: Each position within the interval is associated with the value.
- total: All positions within the interval collectively have the value. E-quivalently, each position has an equally partial amount of the value.

A **signal track** is a set of intervals. Formally, we define a track as a table such that

1. It is a unary relation, containing only one column of type interval.

2. Every tuple in it has the exact same interval fields. It is not allowed that one tuple has some field but another one does not have that field.

The intervals in the same track could be overlapping, and are not necessarily ordered.

2 Interval Location Comparison Relations

All eleven location comparison relations are given in Table 1.

Location Relations	Definition	Remark
I_1 **is adjacent to** I_2	I_1.chr = I_2.chr and (I_1.chrend + 1 = I_2.chrstart or I_1.chrstart − 1 = I_2.chrend)	It is symmetric.
I_1 **coincides with** I_2	I_1.chr = I_2.chr and I_1.chrstart = I_2.chrstart and I_1.chrend = I_2.chrend	It is symmetric.
I_1 **overlaps with** I_2	I_1.chr = I_2.chr and I_1.chrstart $\leq$ I_2.chrend and I_1.chrend $\geq$ I_2.chrstart	It is symmetric.
I_1 **contains** I_2	I_1.chr = I_2.chr and I_1.chrstart $\leq$ I_2.chrstart and I_1.chrend $\geq$ I_2.chrend	It is the inverse relation of is-within.
I_1 **is within** I_2	I_1.chr = I_2.chr and I_1.chrstart $\geq$ I_2.chrstart and I_1.chrend $\leq$ I_2.chrend	It is the inverse relation of contains.
I_1 **is prefix of** I_2	I_1.chr = I_2.chr and I_1.chrstart = I_2.chrstart and I_1.chrend $\leq$ I_2.chrend	
I_1 **is suffix of** I_2	I_1.chr = I_2.chr and I_1.chrstart $\geq$ I_2.chrstart and I_1.chrend = I_2.chrend	
I_1 **precedes** I_2	I_1.chr = I_2.chr and I_1.chrend < I_2.chrstart	It is the inverse relation of follows.
I_1 **follows** I_2	I_1.chr = I_2.chr and I_1.chrstart > I_2.cheend	It is the inverse relation of precedes.
I_1 **is upstream of** I_2	I_1.chr = I_2.chr and ((I_2.strand = '+' and I_1.strand = '+' and I_1 precedes I_2) or (I_2.strand = '+' and I_1.strand = '.' and I_1 precedes I_2) or (I_2.strand = '−' and I_1.strand = '−' and I_1 follows I_2) or (I_2.strand = '−' and I_1.strand = '.' and I_1 is follows I_2))	Only when the two intervals have the same strand, it is the inverse relation of is-downstream-of; otherwise, it is not the inverse relation of is-downstream-of.
I_1 **is downstream of** I_2	I_1.chr = I_2.chr and ((I_2.strand = '+' and I_1.strand = '+' and I_1 follows I_2) or (I_2.strand = '+' and I_1.strand = '.' and I_1 follows I_2) or (I_2.strand = '−' and I_1.strand = '−' and I_1 precedes I_2) or (I_2.strand = '−' and I_1.strand = '.' and I_1 is precedes I_2))	Only when the two intervals have the same strand, it is the inverse relation of is-upstream-of; otherwise, it is not the inverse relation of is-upstream-of.

Table 1: Location Comparison Relations

3 Interval Operations

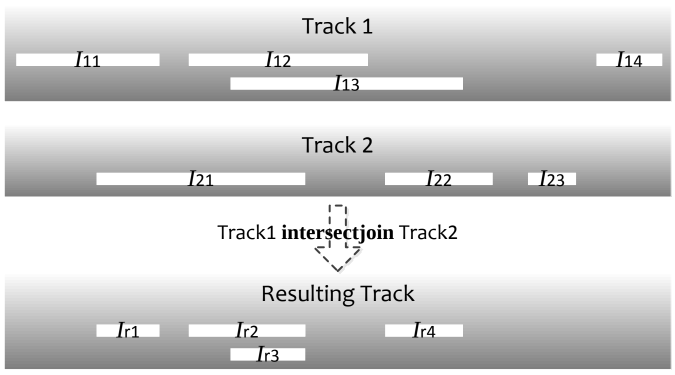
Length of an interval Given one interval I, its length, i.e., length(I), is defined as I.chrend − I.chrstart + 1.

Distance between two intervals Given two intervals I_1, I_2, their distance, i.e., distance(I_1, I_2), is defined as below:

$$\text{distance}(I_1, I_2) = \begin{cases} I_2.\text{chrstart} - I_1.\text{chrend} & \text{if } I_1 \text{ precedes } I_2 \\ 0 & \text{if } I_1 \text{ overlaps with } I_2 \\ I_1.\text{chrstart} - I_2.\text{chrend} & \text{if } I_1 \text{ follows } I_2 \\ \text{NaN} & \text{if } I_1.\text{chr} \neq I_2.\text{chr} \end{cases}$$

4 Track Operations

Intersectjoin Given two tracks T_1, T_2, for each pair of overlapping intervals I_1, I_2, where I_1 is from T_1 and I_2 is from T_2, we compute the common fragment of I_1 and I_2. We denoted the operation of finding the common fragment of I_1 and I_2 by intersect(I_1, I_2). Let I_r = intersect(I_1, I_2).



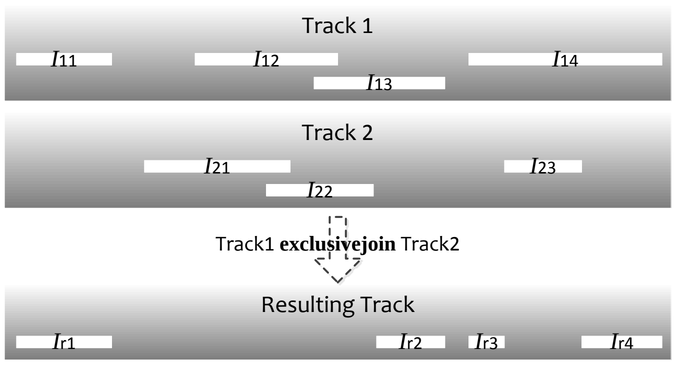
Assume that I_1's value is v_1 and I_2's value is v_2. If we use each model, we can derive the value of I_r, v_r, by using any of the nine types of value-derivation in Table 2.

If we use total model, let v_{f1} $(= v_1 \times \frac{\text{length}(I_r)}{\text{length}(I_1)})$, v_{f2} $(= v_2 \times \frac{\text{length}(I_r)}{\text{length}(I_2)})$, we can derive the value of I_r, v_r, by using any of the nine types of value-derivation in Table 3.

I_r can inherits all meta-data from I_1. Specifically, if I_1 has n (≥ 0) meta-data fields, f_1, f_2, $\ldots$, f_n, I_r also has these n fields and $I_r.f_1 = I_1.f_1$, $I_r.f_2 = I_1.f_2$, $\ldots$, $I_r.f_n = I_1.f_n$.

Exclusivejoin Given two tracks T_1, T_2, for each interval I_1 in T_1, we compute the fragment(s) of I_1 that are not covered by any interval in T_2. Suppose that I_r is such one of the fragments of I_1.

Here we have only one type of value-derivation, i.e., vd_left, which means that we derive I_r.value from the value of I_1. Assume that I_1's

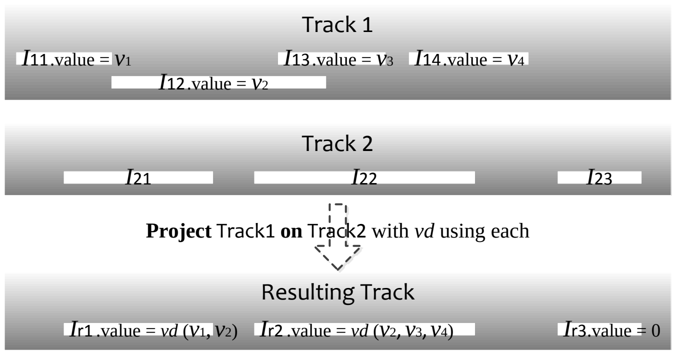


value is v_1. If we use each model, I_r.value $= v_1$. If we use total model, I_r.value $= v_1 \times \frac{\text{length}(I_r)}{\text{length}(I_1)}$.

I_r can inherits all meta-data from I_1.

Project-on Given two tracks T_1, T_2, for each interval I in T_2, we generate an interval I_r that coincides with I, and derive a value for I_r from the values of the intervals in T_1 that overlaps with I_r (or I). It implies that we calculate a value for I_r based on its corresponding positions in T_1. We propose five types of value-derivation.

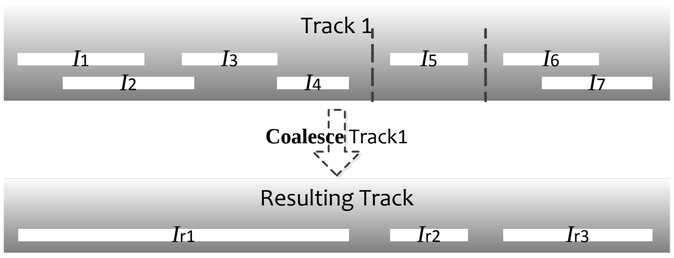
If we use each model, we can derive the value of I_r by using any type of value-derivation in Table 4.



If we use total model, we can derive the value of I_r by using any type of value-derivation in Table 5.

I_r can inherits all meta-data from I.

Coalesce Given a track T, we merge all overlapping and/or adjacent intervals. Consider a subset S of T such that the intervals in S span continuous positions, we generate an interval I_r, whose starting position is the minimum of the starting positions of all intervals in S, and whose ending position is the maximum of the ending positions of all intervals in S. We can also derive a value for I_r. We have five types of value-derivation for coalesce as we have for project-on.

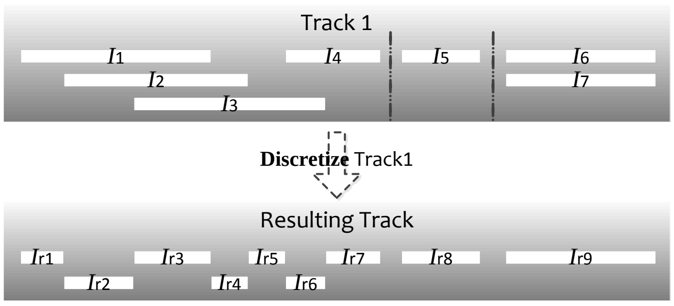


If we use each model, like project-on, for each position p within I_r, we first compute the value at p based on the values of the intervals in S that covers p, by using one type of value-derivation. Then, we compute the average of values across all positions of I_r.

If we use total model, how we derive I_r.value is shown in Table 6.

The coalesce operation does not support reservation of meta-data.

Discretize Given a track T, we break down overlapping intervals using their starting and ending positions so that all resulting intervals are disjoint. Consider a subset S of T such that the intervals in S span continuous positions, we generate a list of adjacent intervals, each spanning from the starting position of some interval in S, to the ending position of some interval in S. Consider one interval in such a list, I_r. Here we also propose five types of value-derivation, each of which can be used to derive the value of I_r.



Suppose that the set $\hat{S}$ comprises of all intervals in S that contains I_r. The derivation of I_r.value under each different value model is shown in Table 7, Table 8, respectively.

The discretize operation does not support reservation of meta-data.

5 STQL Grammar Rules

QUERY := SELECT_STAT FROM_STAT (WHERE_STAT)? (GROUPBY_STAT)? (ORDERBY_STAT)?

FROM_STAT := from FROM_SOURCE

FROM_SOURCE := TRACK UNION TRACK | MULTIPLETRACK

TRACK := RAW_TRACK | TRANSFORM_RES | OVERLAPJOIN_RES | SUBQUERY

MULTIPLETRACK := TRACK (, TRACK)*

UNION := union all

RAW_TRACK := Identifier (TRACKALIAS)?

TRACKALIAS := Identifier

TRANSFORM_RES := TRANSFORM_OP | LBracket TRANSFORM_OP RBracket TRACKALIAS

TRANSFORM_OP := TRANSFORM (with VALUE_DER)?

TRANSFROM := COALESCE TRACK | DISCRETIZE TRACK

COALESCE := coalesce

DISCRETIZE := discretize

OVERLAPJOIN_RES := OVERLAPJOIN_OP | LBracket OVERLAPJOIN_OP RBracket TRACKALIAS

OVERLAPJOIN_OP := OVERLAPJOIN (with (VALUE_DER_METADATA | VALUE_DER | META_DATA))?

OVERLAPJOIN := INTERSECTJOIN | EXCLUSIVEJOIN | PROJECT

INTERSECTJOIN := TRACK intersectjoin TRACK

EXCLUSIVEJOIN := TRACK exclusivejoin TRACK

PROJECT := project TRACK on (TRACK | CREATE_BINS)

CREATE_BINS := generate bins with length Integer

VALUE_DER_METADATA := VALUE_DER, META_DATA | META_DATA, VALUE_DER

VALUE_DER := VD_TYPE using VALUE_MODEL

VD_TYPE := vd_sum | vd_diff | vd_product | vd_quotient | vd_avg | vd_max | vd_min | vd_left | vd_right

META_DATA := metadata

VALUE_MODEL := VM_TYPE model

VM_TYPE := each | all

SELECT_STAT := select ((distinct)? FIELD (, FIELD)* | SELALLEXP)

SELALLEXP := *

FIELD := ARITH_FUNC | AGG

ARITH_FUNC := (MUL_DIV | Number) ((+ | −) (MUL_DIV | Number))?

MUL_DIV := (ELEM | Number) ((* | /) (ELEM | Number))?

ELEM := INTERVAL_ATTR | LBracket ARITH_FUNC RBracket

INTERVAL_ATTR := INTERVAL.ATTRNAME | TRACKNAME.ATTRNAME

INTERVAL := TRACKNAME.interval

TRACKNAME := Identifier | TRACKALIAS

ATTRNAME := chr | chrstart | chrend | value | Identifier

AGG := AGG_FUNC LBracket INTERVAL_ATTR RBracket | COUNT_ALL

AGG_FUNC := count | max | min | avg | sum

COUNT_ALL := count LBracket SELALLEXP RBracket

WHERE_STAT := where OR_PREDICATE

OR_PREDICATE := AND_PREDICATE (or AND_PREDICATE)?

AND_PREDICATE := NOT_PREDICATE (and NOT_PREDICATE)?

NOT_PREDICATE := PREDICATE | not (PREDICATE | LBracket OR_PREDICATE RBracket)

PREDICATE := NUMERIC_COMP | LOCATION_COMP | PATTERN_MATCHING

NUMERIC_COMP := (INTERVAL_ATTR | INTERVAL_LENGTH | INTERVAL_DIS | Number) COMP_OP (INTERVAL_ATTR | INTERVAL_LENGTH | INTERVAL_DIS | Number)

INTERVAL_LENGTH := length LBracket (INTERVAL | CONS_INTERVAL) RBracket

INTERVAL_DIS := distance LBracket (INTERVAL | CONS_INTERVAL) , (INTERVAL | CONS_INTERVAL) RBracket

COMP_OP := < | = | ! = | > | <= | >=

LOCATION_COMP := (INTERVAL | CONS_INTERVAL) LOC_COMP_OP (INTERVAL | CONS_INTERVAL)

LOC_COMP_OP := overlaps with | precedes | follows | matches | is prefix of | is suffix of | coincides with | is within | contains | is upstream of | is downstream of | is closest to

CONS_INTERVAL := LeftSquareBracket CHR, CHRSTART, CHREND (,STRAND)? RightSquareBracket

CHR := Idendifier

CHRSTART := Integer

CHREND := Integer

STRAND := + | −

PATTERN_MATCHING := (not)? like RegularExpression

GROUPBY_STAT := group by INTERVAL_ATTR (, INTERVAL_ATTR)*

ORDERBY_STAT := order by INTERVAL_ATTR (, INTERVAL_ATTR)*

SUBQUERY := LBracket QUERY RBracket Identifier

6 STQL Query Examples

Q1 Find the locations of all intervals in a track T.

SELECT T.interval.chr, T.interval.chrstart, T.interval.chrend
FROM T

Q2 Divide the whole genome into 100bp bins, and compute the value for each bin based on the values at corresponding positions in a track T using vd_sum and each model.

```
SELECT  *
FROM    PROJECT T ON GENERATE BINS
        WITH LENGTH 100 WITH vd_sum USING each model
```

Q3 Find all overlapping interval pairs, one from T_1, the other from T_2.

SELECT $*$
FROM T_1, T_2
WHERE T_1.interval ovelaps with T_2.interval

Q4 Find pairs of intervals such that (1) the first interval is from a track T_1, covering a position p in chr1, and the second one is from T_2; (2) the first interval precedes the second one; (3) their distance is no larger than 10kbp.

```
SELECT  *
FROM    T1, T2
WHERE   T1.interval contains [chr1, p, p] and
        T1.interval precedes T2.interval and
        distance(T1.interval, T2.interval) <= 10000
```

Q5 Find the number of intervals in T with value larger than v for each chromosome.

SELECT T.interval.chr, count($*$)
FROM T
WHERE T.interval.value $>$ v
GROUP BY T.interval.chr

Q6 Find the locations of all intervals in T_1 and T_2.

```
SELECT  *
FROM    (SELECT T1.interval.chr, T1.interval.chrstart, T1.interval.chrend
        FROM T1
        UNION ALL
        SELECT T2.interval.chr, T2.interval.chrstart, T2.interval.chrend
        FROM T2) t
```

value-derivation	$v_r =$	remark
vd_sum	$v_1 + v_2$	The sum of v_1 and v_2
vd_avg	$(v_1 + v_2)/2$	The average of v_1 and v_2
vd_diff	$v_1 - v_2$	The difference of v_1 and v_2
vd_product	$v_1 \times v_2$	The product of v_1 and v_2
vd_quotient	$v_1 \div v_2$	The quotient of v_1 and v_2
vd_max	$\max\{v_1, v_2\}$	The maximum of v_1 and v_2
vd_min	$\min\{v_1, v_2\}$	The minimum of v_1 and v_2
vd_left	v_1	I_1.value
vd_right	v_2	I_2.value

Table 2: Nine types of value-derivation for intersectjoin, under each model

value-derivation	$v_r =$	remark
vd_sum	$v_{f1} + v_{f2}$	The sum of v_{f1} and v_{f2}
vd_avg	$(v_{f1} + v_{f2})/2$	The average of v_{f1} and v_{f2}
vd_diff	$v_{f1} - v_{f2}$	The difference of v_{f1} and v_{f2}
vd_product	$v_{f1} \times v_{f2}$	The product of v_{f1} and v_{f2}
vd_quotient	$v_{f1} \div v_{f2}$	The quotient of v_{f1} and v_{f2}
vd_max	$\max\{v_{f1}, v_{f2}\}$	The maximum of v_{f1} and v_{f2}
vd_min	$\min\{v_{f1}, v_{f2}\}$	The minimum of v_{f1} and v_{f2}
vd_left	v_{f1}	The fractional share of I_1.value
vd_right	v_{f2}	The fractional share of I_2.value

Table 3: Nine types of value-derivation for intersectjoin, under total model

value-derivation	I_r.value =	remark
vd_sum	$\frac{\sum_{p=I_r.\text{chrstart}}^{I_r.\text{chrend}} \sum_{\substack{\hat{I}\in\hat{S} \\ \hat{S}=\{I' \mid I'\in T_1 \wedge I' \text{ covers } p\}}} \hat{I}.\text{value}}{\text{length}(I_r)}$	For each p within I_r, compute the sum of values of the intervals in T_1 that covers it. Compute the average value across I_r.
vd_avg	$\frac{\sum_{p=I_r.\text{chrstart}}^{I_r.\text{chrend}} \left(\frac{\sum_{\substack{\hat{I}\in\hat{S} \\ \hat{S}=\{I' \mid I'\in T_1 \wedge I' \text{ covers } p\}}} \hat{I}.\text{value}}{\lvert\hat{S}\rvert} \right)}{\text{length}(I_r)}$	For each p within I_r, compute the average of values of the intervals in T_1 that covers it. Compute the average value across I_r.
vd_product	$\frac{\sum_{p=I_r.\text{chrstart}}^{I_r.\text{chrend}} \prod_{\substack{\hat{I}\in\hat{S} \\ \hat{S}=\{I' \mid I'\in T_1 \wedge I' \text{ covers } p\}}} \hat{I}.\text{value}}{\text{length}(I_r)}$	For each p within I_r, compute the product of values of the intervals in T_1 that covers it. Compute the average value across I_r.
vd_max	$\frac{\sum_{p=I_r.\text{chrstart}}^{I_r.\text{chrend}} \max_{\substack{\hat{I}\in\hat{S} \\ \hat{S}=\{I' \mid I'\in T_1 \wedge I' \text{ covers } p\}}} \hat{I}.\text{value}}{\text{length}(I_r)}$	For each p within I_r, compute the maximum of values of the intervals in T_1 that covers it. Compute the average value across I_r.
vd_min	$\frac{\sum_{p=I_r.\text{chrstart}}^{I_r.\text{chrend}} \min_{\substack{\hat{I}\in\hat{S} \\ \hat{S}=\{I' \mid I'\in T_1 \wedge I' \text{ covers } p\}}} \hat{I}.\text{value}}{\text{length}(I_r)}$	For each p within I_r, compute the minimum of values of the intervals in T_1 that covers it. Compute the average value across I_r.

Table 4: Five types of value-derivation for project-on, under each model

value-derivation	I_r.value =	remark
vd_sum	$\sum_{\substack{I' \in T_1 \wedge \\ I' \text{ ovelaps with } I_r}} \left(I'.\text{value} \times \frac{\text{length}(\text{intersect}(I_r, I'))}{\text{length}(I')} \right)$	The sum of the values
vd_avg	$\frac{\sum_{\substack{I' \in T_1 \wedge \\ I' \text{ ovelaps with } I_r}} \left(I'.\text{value} \times \frac{\text{length}(\text{intersect}(I_r, I'))}{\text{length}(I')} \right)}{\lvert \{ I' \mid I' \in T_1 \wedge I' \text{ overlaps with } I_r \} \rvert}$	The average of the values
vd_product	$\prod_{\substack{I' \in T_1 \wedge \\ I' \text{ ovelaps with } I_r}} \left(I'.\text{value} \times \frac{\text{length}(\text{intersect}(I_r, I'))}{\text{length}(I')} \right)$	The product of the values
vd_max	$\max_{\substack{I' \in T_1 \wedge \\ I' \text{ ovelaps with } I_r}} \left(I'.\text{value} \times \frac{\text{length}(\text{intersect}(I_r, I'))}{\text{length}(I')} \right)$	The maximum of the values
vd_min	$\min_{\substack{I' \in T_1 \wedge \\ I' \text{ ovelaps with } I_r}} \left(I'.\text{value} \times \frac{\text{length}(\text{intersect}(I_r, I'))}{\text{length}(I')} \right)$	The minimum of the values

Table 5: Five types of value-derivation for project-on, under total model

value-derivation	I_r.value =	remark
vd_sum	$\sum_{I_i \in S} I_i.\text{value}$	The sum of the values of the intervals in S
vd_avg	$\left(\sum_{I_i \in S} I_i.\text{value}\right) / \lvert S \rvert$	The average of the values of the intervals in S
vd_product	$\prod_{I_i \in S} I_i.\text{value}$	The product of the values of the intervals in S
vd_max	$\max_{I_i \in S} I_i.\text{value}$	The maximum of the values of the intervals in S
vd_min	$\min_{I_i \in S} I_i.\text{value}$	The minimum of the values of the intervals in S

Table 6: Five types of value-derivation for coalesce, under total model

value-derivation	I_r.value =	remark
vd_sum	$\sum_{\hat{I} \in \hat{S}} \hat{I}.\text{value}$	The sum of the values the intervals in $\hat{S}$
vd_avg	$\left(\sum_{\hat{I} \in \hat{S}} \hat{I}.\text{value}\right) / \lvert \hat{S} \rvert$	The average of the values the intervals in $\hat{S}$
vd_product	$\prod_{\hat{I} \in \hat{S}} \hat{I}.\text{value}$	The product of the values the intervals in $\hat{S}$
vd_max	$\max_{\hat{I} \in \hat{S}} \hat{I}.\text{value}$	The maximum of the values the intervals in $\hat{S}$
vd_min	$\min_{\hat{I} \in \hat{S}} \hat{I}.\text{value}$	The minimum of the values the intervals in $\hat{S}$

Table 7: Five types of value-derivation for discretize, under each model

value-derivation	I_r.value =	remark
vd_sum	$\sum_{\hat{I} \in \hat{S}} \left(\hat{I}.\text{value} \times \frac{\text{length}(I_r)}{\text{length}(\hat{I})} \right)$	The sum of the values
vd_avg	$\left(\sum_{\hat{I} \in \hat{S}} \left(\hat{I}.\text{value} \times \frac{\text{length}(I_r)}{\text{length}(\hat{I})} \right) \right) / \lvert \hat{S} \rvert$	The average of the values
vd_product	$\prod_{\hat{I} \in \hat{S}} \left(\hat{I}.\text{value} \times \frac{\text{length}(I_r)}{\text{length}(\hat{I})} \right)$	The product of the values
vd_max	$\max_{\hat{I} \in \hat{S}} \left(\hat{I}.\text{value} \times \frac{\text{length}(I_r)}{\text{length}(\hat{I})} \right)$	The maximum of the values
vd_min	$\min_{\hat{I} \in \hat{S}} \left(\hat{I}.\text{value} \times \frac{\text{length}(I_r)}{\text{length}(\hat{I})} \right)$	The minimum of the values

Table 8: Five types of value-derivation for discretize, under total model