CS 332/780: Object-Oriented Databases   Spring 2018
Course Notes #2
Keitaro Yukawa
Department of Computer Science
Queens College, CUNY

Relations

The concept of *relation* plays an important role in any types of database modeling methods and database systems. We will review terminology and formalism of relations that are useful for object-oriented data modeling and databases. Their understanding only requires a knowledge of introductory discrete mathematics.

For each finite set S, $|S|$ denotes the number of elements in S.

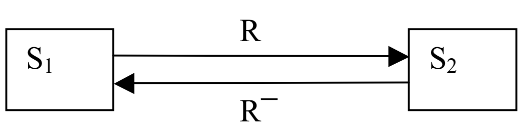
The Cartesian product of n sets $S_1, \ldots, S_n$, denoted $S_1 \times \cdots \times S_n$, is defined to be $\{ \langle x_1, \ldots, x_n \rangle \mid x_i \in S_i, 1 \le i \le n \}$, the set of all n-tuples whose i-th component is a member of S_i. An *n-ary relation* over sets $S_1, \ldots, S_n$ is a subset of $S_1 \times \cdots \times S_n$, that is, any set of n-tuples whose i-th component is a member of S_i. We use the notation $R(S_1, \ldots, S_n)$ to indicate that the relation R is defined over $S_1, \ldots, S_n$, and $R(a_1, \ldots, a_n)$ to indicate that $\langle a_1, \ldots, a_n \rangle \in R$. The relations over $S_1, \ldots, S_n$ form a lattice with respect to set containment $\subseteq$ with the Cartesian product $S_1 \times \cdots \times S_n$ at the top and the empty relation $\varnothing$, the empty set of tuples, at the bottom. If R_1, R_2 are relations over $S_1, \ldots, S_n$ and $R_1 \subseteq R_2$, R_1 is said to be a sub-relation of R_2 and R_2 is said to be a super-relation of R_1. A relation is finite if it has a finite number of tuples. If the S_i are all finite, $S_1 \times \cdots \times S_n$ has $|S_1| \times \cdots \times |S_n|$ elements, hence there exist $2^{|S_1| \times \cdots \times |S_n|}$ relations over $S_1, \ldots, S_n$.

Often 2-ary and 3-ary relations are called, respectively, *binary* and *ternary* relations. A binary relation $R(S_1, S_2)$ is sometimes denoted by the arrow notation $R: S_1 \to S_2$ when we wish to connote a directional view of R from S_1 to S_2. Directional view of binary relations has significance in OO data modeling and databases as representation of binary relations spanned by object references. In database applications, a set S_i may contain entities in application domains or elements of basic types like integers, floating-point numbers, characters, and strings. We will refer to these entities and elements as *objects*.

Consider any 1-ary relation $R(S) \subseteq \{ \langle x \rangle \mid x \in S \}$. This is essentially the same as the set $R' = \{ x \mid \langle x \rangle \in R \}$, obtained from R by turning each 1-tuple $\langle x \rangle$ to its component x. Conversely, any set can be converted to the corresponding 1-ary relation by turning each object x to $\langle x \rangle$. Since the only difference between R′ and R is whether each object x appears as itself or $\langle x \rangle$, we identify them when no confusion should arise.

For each binary relation $R(S_1, S_2)$, its *inverse relation* $R^{-}(S_2, S_1)$ is defined by:

$$\forall x \in S_1 \forall y \in S_2 \, (R(x, y) \Leftrightarrow R^{-}(y, x)).$$



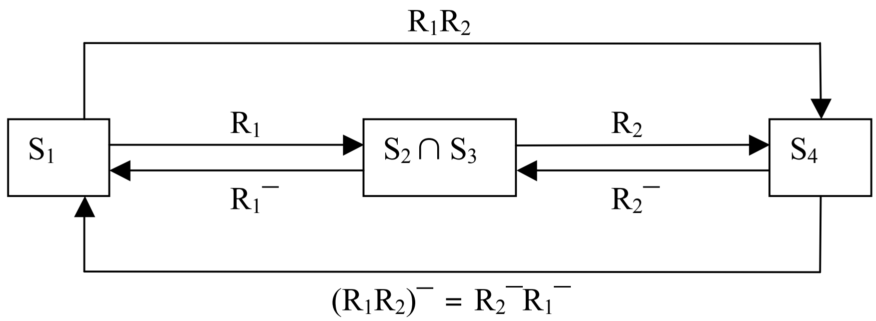
It is clear from the definition that $(R^{-})^{-} = R$ for any R. The inverse operation does not alter the information content of relation $R(S_1, S_2)$ as a whole; rather, it changes a viewpoint by switching the order of S_1 and S_2.

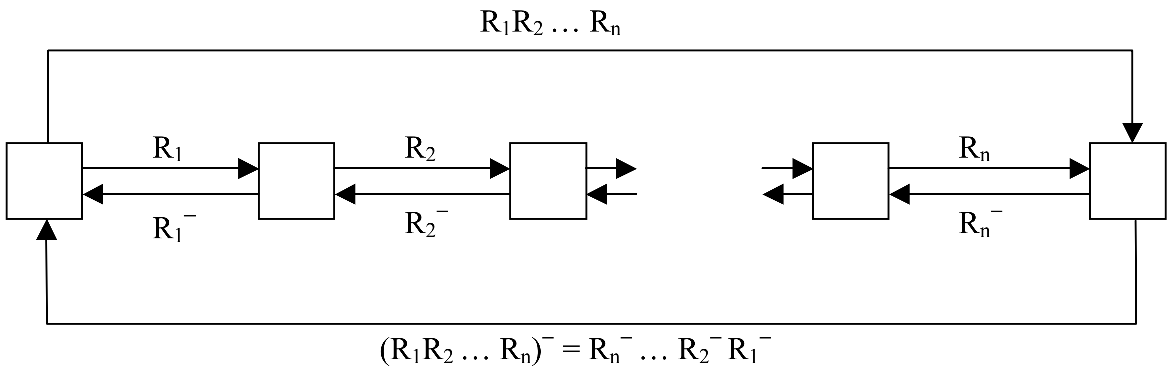
For any two binary relations $R_1(S_1, S_2)$ and $R_2(S_3, S_4)$, the *composition* of R_1 and R_2, denoted $R_1R_2(S_1, S_4)$, is defined by:

$$\forall x \in S_1 \forall y \in S_4 \ (R_1R_2(x, y) \Leftrightarrow \exists z \in (S_2 \cap S_3)(R_1(x, z) \wedge R_2(z, y))).$$

If $S_2 \cap S_3 = \varnothing$, the composition R_1R_2 is the empty relation, namely the empty set of tuples. Composition is *associative*: for any three relations R_1, R_2, R_3, $(R_1R_2)R_3 = R_1(R_2R_3)$. Therefore the composition of *n* relations may be written as $R_1R_2 \ldots R_n$ without parentheses. For any *n* binary relations $R_1, R_2, \ldots, R_n$, it holds that

$$(R_1R_2 \ldots R_{n-1}R_n)^{-} = R_n^{-}R_{n-1}^{-} \ldots R_2^{-}R_1^{-}.$$



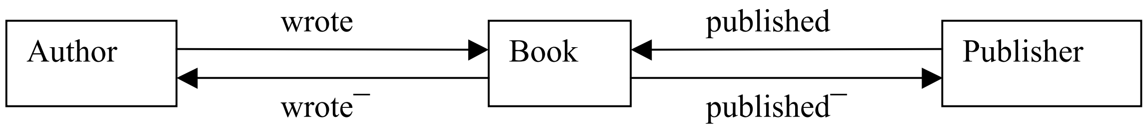


Let $R_i(S_{2i-1}, S_{2i})$, $1 \le i \le n$, and let G be the directed graph where the node set is $S_1 \cup \ldots \cup S_{2n}$ and there is a directed link labeled with R_i from x to y if and only if $R_i(x, y)$ holds. Then $R_1R_2 \ldots R_n(x, y)$ if and only if there is a directed path from x to y consisting of labeled links $R_1, R_2, \ldots, R_n$ in G — it does not matter which path is chosen.

For any R(S, S), the *i-fold composition* of R, denoted R^i, is defined recursively: $R^0 = I$, $R^{i+1} = RR^i$, where I is the identity relation $\{\langle x, x\rangle : x \in S\}$. It is R composed with itself *i* times. The *transitive closure* of R, R^+, is defined to be $\bigcup_{i \ge 1} R^i$ and the *reflexive-transitive closure* of R, R^*, is defined to be $\bigcup_{i \ge 0} R^i$. It holds that $(R^i)^{-} = (R^{-})^i$, $(R^+)^{-} = (R^{-})^+$, and $(R^*)^{-} = (R^{-})^*$.

Example Let Book, Author, and Publisher be sets of books, authors, and publishers. Consider the two relations:

- wrote(Author, Book): wrote(*a*, *b*) means author *a* wrote book *b*, possibly with other authors.
- published(Publisher, Book): published(*p*, *b*) means publisher *p* published book *b*.



Then:

- wrote^{-}(Book, Author): wrote^{-}(*b*, *a*) means book *b* was written by author *a*.
- published^{-}(Book, Publisher): published^{-}(*b*, *p*) means book *b* was published by publisher *p*.
- (wrote published^{-})(Author, Publisher): (wrote published^{-})(*a*, *p*) means author *a* wrote a book published by publisher *p*.
- (published wrote^{-})(Publisher, Author): (published wrote^{-})(*p*, *a*) means publisher *p* published a book written by author *a*.
- wrote published^{-} and published wrote^{-} are inverses of each other because (wrote published^{-})$^{-}$ = published$^{-\,-}$ wrote^{-} = published wrote^{-}.

Consider any $R(S_1, S_2)$. The relation $RR^{-}(S_1, S_1)$ is defined by:

$$\forall x{\in}S_1 \forall y{\in}S_1 \ (\ RR^{-}(x, y) \Leftrightarrow \exists z{\in}S_2(R(x, z) \land R^{-}(z, y)) \ ),$$

or, equivalently:

$$\forall x{\in}S_1 \forall y{\in}S_1 \ (\ RR^{-}(x, y) \Leftrightarrow \exists z{\in}S_2(R(x, z) \land R(y, z)) \ ).$$

Note that x and y may be the identical object; in this case, the above reduces to:

$$\forall x{\in}S_1 \ (\ RR^{-}(x, x) \Leftrightarrow \exists z{\in}S_2 \ R(x, z) \ ).$$

If $R(x, z) \land R(y, z)$ holds, x and y are said to *share* z *via relation* R. $RR^{-}(x, y)$ holds if x and y share at least one z via R.

The relation RR^{-} models some interesting relations in reality.

Example Consider the following binary relations:

- parent(Person, Person): parent(*p*, *c*) means *p* is a biological parent of *c*.
- child = parent^{-}: child(*c*, *p*) means *c* is a biological child of *p*.
- memberOf(Person, Team): memberOf(*m*, *t*) means *m* is a member of team *t*.
- hasMember = memberOf^{-}: hasMember(*t*, *m*) means team *t* has member *m*.

Then:

- parent^{+}(x, y) if and only if x is a biological ancestor of y, and (parent^{+})$^{-}$ = (parent^{-})$^{+}$ = child^{+}, i.e. the descendant relation.
- (parent parent^{-})(x, y) = (child^{-} child)(x, y) means x and y are biological parents of at least one common child, including the special case where x = y and x has a child.
- (child child^{-})(x, y) = (parent^{-} parent)(x, y) means x and y are biological children of at least one common parent, i.e., common biological mother or father, including the case where x and y are half-siblings. This relation always includes the identity relation $\{ \langle x, x \rangle : x \in \text{Person} \}$ as everyone has a biological parent.

- (memberOf memberOf$^-$)(x, y) = (hasMember$^-$ hasMember)(x, y) means persons x and y are members of at least one common team, including the special case where x = y and x is a member of a team.
- (hasMember hasMember$^-$)(x, y) = (memberOf$^-$ memberOf)(x, y) means teams x and y have at least one common member.

Example Let S be a set of data objects created in a programming language. Define R(S, S) by: R(x, y) holds if and only if data object x contains, or has a field containing, a reference/pointer to data object y. Then R^i(x, y) holds if there is a chain of *i* references/pointers from x to y, and R^+(x, y) holds if there is a chain of references/pointers from x to y. RR^-(x, y) holds if there is a data object z to which both x and y reference/point. This is often called data/object sharing by references/pointers.

The concepts of *projection* and *multiplicity* (sometimes called *cardinality*) of relations are useful for database modeling and understanding queries. The projection operation reduces the arities of relations by fixing objects or by existential quantifications of some of the relation's arguments.

Let $R(S_1, ..., S_n)$ be an *n*-ary relation. In this definition, each S_i is identified by its positional index *i*. Let

$I = \{ i_1, …, i_p \}$ with $i_1 < … < i_p$
$J = \{ j_1, …, j_q \}$ with $j_1 < … < j_q$
$K = \{ k_1, …, k_r \}$ with $k_1 < … < k_r$

such that $I \cup J \cup K = \{ 1, …, n \}$ and I, J, K are mutually disjoint. I represents the S_i onto which R is projected. J represents the S_j from which specific objects a_j's are chosen. K represents the existentially quantified S_k. Let $a_m \in S_m$ for each $m \in J$. The *projection* of R onto I with respect to the a_m, $m \in J$, denoted $R{\downarrow}I(a_{j_1}, …, a_{j_q})$, is defined as the *p*-ary relation

$$\{ \langle x_{i_1}, …, x_{i_p}\rangle \mid (\exists x_{k_1}) … (\exists x_{k_r})\ R(y_1, …, y_n) \text{ where } \begin{array}{l} y_m = x_m \text{ for each } m \in I \cup K \text{ and} \\ y_m = a_m \text{ for each } m \in J \end{array} \}$$

If $J = \varnothing$, the projection is denoted by $R{\downarrow}I$, called the projection of R onto I. The *multiplicity* of R with respect to the a_m, $m \in J$, is $|R{\downarrow}I(a_{j_1}, …, a_{j_q})|$.

Example Let $R(S_1, ..., S_6)$ be a 6-ary relation. Let I = {3, 5}, J = {1, 4}, K = {2, 6}. Select arbitrary but fixed $a_m \in S_m$, $m \in J$. Then

$$R{\downarrow}I(a_1, a_4) = \{ \langle x_3, x_5\rangle \mid \exists x_2\ \exists x_6\ R(a_1, x_2, x_3, a_4, x_5, x_6) \}.$$

Let I = {3, 5}, J = {1, 2, 4, 6}, $K = \varnothing$, $a_m \in S_m$, $m \in J$. Then

$$R{\downarrow}I(a_1, a_2, a_4, a_6) = \{ \langle x_3, x_5\rangle \mid R(a_1, a_2, x_3, a_4, x_5, a_6) \}.$$

Let I = {3, 5}, $J = \varnothing$, K = {1, 2, 4, 6}. Then

$$R{\downarrow}I = \{ \langle x_3, x_5\rangle \mid \exists x_1\ \exists x_2\ \exists x_4\ \exists x_6\ R(x_1, x_2, x_3, x_4, x_5, x_6) \}.$$

It is helpful to think of an existential quantification $\exists x$ as a "wild card variable" that matches any one object within the constraint of relation R (in general, within the constraint of the logical formula following or containing $\exists x$).

In the following examples, we will use the sequence $\langle S_{i_1}, …, S_{i_p}\rangle$ of the sets S_i, $i \in I$, instead of the index set I, and will freely use the convention of identifying any 1-ary relation with the corresponding set.

Example Consider the *wrote* and *published* relations in the previous example. The set of books written by author a is

$$\text{wrote}{\downarrow}\text{Book}(a) = \{\, b \in \text{Book} \mid \text{wrote}(a, b) \,\}.$$

This can be equivalently expressed by $\text{wrote}^{\smile}$:

$$\text{wrote}^{\smile}{\downarrow}\text{Book}(a) = \{\, b \in \text{Book} \mid \text{wrote}^{\smile}(b, a) \,\} = \{\, b \in \text{Book} \mid \text{wrote}(a, b) \,\}.$$

The set of publishers that published a book written by author a is

$$(\text{wrote published}^{\smile}){\downarrow}\text{Publisher}(a) = \{\, p \in \text{Publisher} \mid (\text{wrote published}^{\smile})(a, p) \,\}.$$

The set of authors that wrote a book published by publisher p is

$$(\text{wrote published}^{\smile}){\downarrow}\text{Author}(p) = \{\, a \in \text{Author} \mid (\text{wrote published}^{\smile})(a, p) \,\}.$$

Example Consider a binary relation offer(Department, CourseSection) where Department and CourseSection are, respectively, the sets of academic departments and of course sections. The semantics of offer(d, c) is that department d offered course section c. Then

$$\text{offer}{\downarrow}\text{CourseSection}(d) = \{\, c \in \text{CourseSection} \mid \text{offer}(d, c) \,\}$$

is the set of course sections offered by department d, and

$$\text{offer}{\downarrow}\text{Department}(c) = \{\, d \in \text{Department} \mid \text{offer}(d, c) \,\}$$

is the set of departments that offered course section c.

Example Consider a 4-ary relation take(Student, CourseSection, Program, Grade) where Student, CourseSection, Program, and Grade are, respectively, the sets of students, course sections, programs, and letter grades in a certain college. The semantics of take(s, c, p, g) is that student s took course section c in program p receiving grade g.

- $\text{take}{\downarrow}\text{Grade}(s,c,p) = \{\, g \in \text{Grade} \mid \text{take}(s, c, p, g) \,\}$ is the set of letter grades student s received in course section c in program p; this set has only one object as every student gets a unique grade in each course section, and so its multiplicity is always one.
- $\text{take}{\downarrow}\text{CourseSection}(s,p,g)$ is the set of course sections in which student s received grade g in program p.
- $\text{take}{\downarrow}\text{Student}(c,p,g)$ is the set of students that took course section c in program p and received grade g in it.
- $\text{take}{\downarrow}\langle\text{CourseSection, Program, Grade}\rangle(s)$ is the 3-ary relation of all ⟨course section, program, grade⟩ triples that were taken by a particular student s.
- $\text{take}{\downarrow}\langle\text{Student, CourseSection, Program}\rangle(g)$ is the 3-ary relation of all ⟨student, course section, program⟩ triples that are related to a particular grade g.
- $\text{take}{\downarrow}\langle\text{Student, Grade}\rangle(c, p) = \{\, \langle s, g\rangle \mid \text{take}(s, c, p, g) \,\}$ is the binary relation of all student-grade pairs such that each student in the set received the paired grade in the given course section c in the given program p.
- $\text{take}{\downarrow}\text{CourseSection} = \{\, c \in \text{CourseSection} \mid \exists s\, \exists p\, \exists g\ \text{take}(s, c, p, g) \,\}$ is the set of all course sections that were each taken by at least one student.
- $\text{take}{\downarrow}\text{Student} = \{\, s \in \text{Student} \mid \exists c\, \exists p\, \exists g\ \text{take}(s, c, p, g) \,\}$ is the set of all students that took at least one course section.
- $\text{take}{\downarrow}\text{CourseSection}(s) = \{\, c \in \text{CourseSection} \mid \exists p\, \exists g\ \text{take}(s, c, p, g) \,\}$ is the set of all course sections taken by the given student s regardless of programs and grades.

- take↓Student(c) = { $s \in$ Student | $\exists p\ \exists g$ take(s, c, p, g) } is the set of all students who took the given course section c regardless of programs and grades.

If we use the names of sets S_i instead of their positional indexes i, as we have done in the above examples, ambiguities will arise if some of the sets S_i are identical. Consider, for example, a binary relation prerequisite(Course, Course) where the semantics of prerequisite(c_1, c_2) is that course c_1 is a prerequisite of course c_2. The meaning of prerequisite↓Course(c), where c is a course, is ambiguous since we do not know if Course refers to the first or second argument of the relation. This is easily resolved by attaching the sets' positional indexes to their names as subscripts. Then

prerequisite↓Course_1(c) = { c_1 | prerequisite(c_1, c) }

is the set of all prerequisite courses for c since Course_1 refers to the 1^{st} Course argument. On the other hand

prerequisite↓Course_2(c) = { c_2 | prerequisite(c, c_2) }

is the set of courses for which c is a prerequisite since Course_2 refers to the 2^{nd} Course argument.

For another example, let connect(Airport, Airport, Airport) be a 3-ary relation with the semantics of connect(a_1, a_2, a_3) meaning that there is a direct flight from a_1 to a_2 and also from a_2 to a_3 but none from a_1 to a_3. Then

connect↓Airport_2(a, a') = { a_2 | connect(a, a_2, a') }

is the set of all potential connecting airports that can be used for flying from a to a'. The binary relation

connect↓⟨Airport_1, Airport_3⟩(a) = { ⟨a_1, a_3⟩ | connect(a_1, a, a_3) }

is the set of all pairs of airports ⟨a_1, a_3⟩ that can use a as a connecting airport.

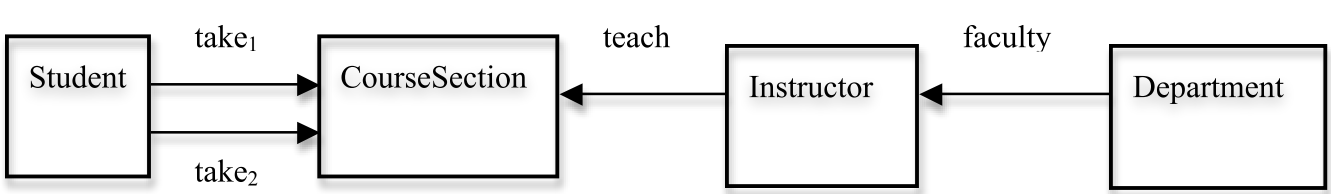
Recall that projection of any relation onto p sets $S_{i_1}, \ldots, S_{i_p}$, with respect to specific objects or by existential quantifiers, is a p-ary relation over $S_{i_1}, \ldots, S_{i_p}$. In particular, projection can be used to reduce an n-ary relation, $n \geq 3$, to binary relations, which in turn can be composed with other binary relations. Consider again the 4-ary relation take(Student, CourseSection, Program, Grade). This relation can be reduced, for example, to these binary relations:

take_1 = take↓⟨Student, CourseSection⟩(p, g),

take_2 = take↓⟨Student, CourseSection⟩

Note that take_1 is a sub-relation of take_2, i.e., $\text{take}_1 \subseteq \text{take}_2$.

Let teach(Instructor, CourseSection) and faculty(Department, Instructor) be the binary relations such that teach(i, c) means instructor i taught course section c and faculty(d, i) means department d has faculty member i.



The composition

$\text{take}_1\ \text{teach}^{\smile} = (\text{take}{\downarrow}\langle \text{Student, CourseSection}\rangle(p, g))\text{teach}^{\smile}$: Student → Instructor

is the binary relation of $\langle s, i\rangle$ such that student s took at least one course section taught by instructor i in program p and received grade g in it. The composition

$\text{take}_2\ \text{teach}^{\smile} = (\text{take}{\downarrow}\langle \text{Student, CourseSection}\rangle)\text{teach}^{\smile}$: Student → Instructor

is the binary relation of $\langle s, i\rangle$ such that student s took at least one course section taught by instructor i regardless of programs and grades. The composition

$\text{take}_2\ \text{teach}^{\smile}\ \text{faculty}^{\smile}$: Student → Department

is the binary relation of $\langle s, d\rangle$ such that student s took at least one course section taught by an instructor that is a faculty member of department d.

For last example we look at some of the basic geometric projections used in the area of spatial databases. Let each of X, Y, Z be the set of floating-point numbers, and consider a 3-ary relation V(X, Y, Z) representing a set of points in the 3-dimensional space with X, Y, Z axes. The following are the projections of V onto the XY-, YZ-, and XZ-planes respectively:

$V{\downarrow}\langle X, Y\rangle = \{ \langle x, y\rangle \mid \exists z\, V(x, y, z) \}$
$V{\downarrow}\langle Y, Z\rangle = \{ \langle y, z\rangle \mid \exists x\, V(x, y, z) \}$
$V{\downarrow}\langle X, Z\rangle = \{ \langle x, z\rangle \mid \exists y\, V(x, y, z) \}$

$V{\downarrow}\langle X, Y\rangle (z_0) = \{ \langle x, y\rangle \mid V(x, y, z_0) \}$ is the projection of V onto the XY-plane with respect to a particular value z_0 on the Z-axis. The following are the projections of V onto the X-, Y-, and Z-axis respectively:

$V{\downarrow}X = \{ x \mid \exists y \exists z\, V(x, y, z) \}$
$V{\downarrow}Y = \{ y \mid \exists x \exists z\, V(x, y, z) \}$
$V{\downarrow}Z = \{ z \mid \exists x \exists y\, V(x, y, z) \}$

$V{\downarrow}X(y_0, z_0) = \{ x \mid V(x, y_0, z_0) \}$ is the projection of V onto the X-axis with respect to particular values y_0, z_0 on the Y- and Z-axis.

Let T be the set of floating-point numbers and let VT(X, Y, Z, T) be a set of points in the (3+1)-dimensional spacetime with a time axis T.

$VT{\downarrow}\langle X, Y, Z\rangle = \{ \langle x, y, z\rangle \mid \exists t\, VT(x, y, z, t) \}$ is the projection of VT onto the 3-dimensional space, i.e., the set of all 3-dimensional space points existing inside VT at some arbitrary time instant t.

$VT{\downarrow}T = \{ t \mid \exists x \exists y \exists z\, VT(x, y, z, t) \}$ is the set of all time instants when at least one space point exists inside VT.

$VT{\downarrow}\langle X, T\rangle (y_0) = \{ \langle x, t\rangle \mid \exists z\, VT(x, y_0, z, t) \}$ is the projection of VT onto the X- and T-axis with respect to a particular value y_0 on the Y-axis.