Introduction to CAT4: Part 1. Axioms

Andrew Holster
Jan 2021

Abstract

CAT4 is proposed as a general method for representing information, enabling a powerful programming method for large-scale information systems. It enables generalised machine learning, software automation and novel AI capabilities. It is based on a special type of *relation* called CAT4, which is interpreted to provide a semantic representation. This is Part 1 of a five-part introduction. The focus here is on defining the key mathematical structures first, and presenting the semantic-database application in subsequent Parts. We focus in Part 1 on general axioms for the structures, and introduce key concepts. Part 2 analyses the CAT2 sub-relation of CAT4 in more detail. The interpretation of *fact networks* is introduced in Part 3, where we turn to interpreting *semantics.* We start with examples of relational and graph databases, with methods to translate them into CAT3 networks, with the aim of retaining the *meaning* of information. The full application to semantic theory comes in Part 4, where we introduce general functions, including the language interpretation or *linguistic functions*. The representation of *linear symbolic languages,* including natural languages and formal symbolic languages, is a function that CAT4 is uniquely suited to. In Part 5, we turn to software design considerations, to show how files, indexes, functions and screens can be defined to implement a CAT4 system efficiently.

Contents

Introduction.

CAT4 is proposed as a general method for representing information, enabling a powerful programming method for large-scale information systems. It enables generalised machine learning, software automation and novel AI capabilities. In the theoretical sphere, it unifies formal theories of semantic logic and information programming through a mathematical model. CAT4 is an extension of the earlier CAT3 method, which was awarded a US Patent [Holster, 2011]. This patent lapsed in 2014. The intention of this publication is to put the theory in the public domain, making it freely available to researchers or developers. We state the general aim of the CAT4 model as:

- The aim of CAT4 is to provide a *complete semantic representation of information.*

We do this by defining a special type of network structure, which we interpret it as a *fact database,* with points of the network representing facts. This system of facts and their relations provides a *semantic model of information.* We see the semantic relations are fully represented by the network join structure, which is primarily what CAT4 introduces. The compactness and simplicity of this is what enables its strong properties. CAT4 is proposed to have major advantages over current relational and graph databases, object oriented programming methods, etc.

The properties can be motivated by examples, but they come back to general mathematical properties, and the focus here is on defining the core mathematical structure of the C2 relation first, and presenting the semantic-database application subsequently. We focus first on mathematical properties of the network structures, with variations called *C2, CA2, CAT2, CAT3, CAT4,* and *FCAT2.* Part 1 provides general axioms for these, and introduces key concepts. Part 2 analyses the CAT2 sub-relation of CAT4 in more detail. The interpretation of *fact networks* is introduced in Part 3, where we turn to interpreting *semantics.* We start with examples of relational and graph databases, with methods to translate them into CAT3 networks, with the aim of retaining the *meaning* of information. The full application to semantic theory comes in Part 4, where we introduce general functions, including the language interpretation or *linguistic functions*. The representation of *linear symbolic languages,* including natural languages and formal symbolic languages, is a function that CAT4 is uniquely suited to. It provides a natural method for representing meaning and grammar together with their facts. Translation between languages is a natural function of this method, and it is continuous with general data integration, from external sources into CAT4. The CAT4 representation of languages is very different to conventional logical-grammar theories. In Part 5, we turn to software design considerations, to show how files, indexes, functions and screens can be defined to implement a CAT4 system efficiently.

The treatment in the first parts is formal and intended for mathematicians or programmers. It provides important detail for programming a system. An axiomatisation of the CAT2-CAT3-CAT4 *network structures* is given first. The structures are defined with the subsequent *interpretation* in mind, and we can adapt them to this purpose to some extent. But the underlying mathematical structure is simple and natural, and we must maintain these uniform properties. The uniformity enables universal recursive functions over the network domain.

We describe implementations as *tables* to help visualisation, and this becomes the important software design theme later*.* The *fact interpretation* is mentioned in earlier parts, but is not discussed in detail until Parts 3 and 4. Note the *semantic interpretation* is not formally axiomatized, rather, it is presented as fulfilling certain general principles. A more systematic treatment might formalise this interpretation as well, but this would amount to a full semantic meta-theory, which is beyond the scope here.

The aim of Parts 1 and 2 is to provide a formal basis for proof of key properties. A number of basic propositions are proved. Proofs are mainly given from first principles, so we need few special theorems from other areas. Models are defined over finite integer domains, and this matches the programming method, because we use integer indexes for table rows in the same way. Once essential properties are established, methods from several areas of mathematics can be used to analyse the structure, including abstract algebra, graph theory, topology, model theory.

While the mathematics of the network is quite straightforward, the semantic model may be more debateable, as there is no standard theory. We introduce this in Parts 3 and 4, motivating its main points with examples. In judging it, we can use our own intuition first: does the representation and interpretation of information make sense? But to understand system properties like completeness, consistency, logical adequacy, etc, we have to look at the theoretical level. There is no single accepted theory of *semantics* in philosophy or linguistics or computer science to compare with. In fact there are many mutually contradictory theories, in mutually incompatible traditions. Semantic theorists are often not even talking about the same things.

Nonetheless, most of the significant logical-semantic theories that developed in the C19th - C20th have some core logical-linguistic phenomenon or intuition at heart. We can compare CAT4 semantics with these core phenomena. We want to explain more general phenomenon of meaning, and we can consider CAT4 as a broadly explanatory theory. This is part of the fascination when we apply CAT4 to the larger fields of linguistics, philosophy, and information systems.

However, our concern here is primarily to define and analyse CAT4 formally, and in terms of semantic theory, its closest comparison is with the tradition of formal semantic logics, developed by the likes of Frege, Russell, Goedel, Tarski, Church, Turing, Kleene, Carnap, Kripke, Montague, Tichý, and others, and continuing in different programs in modern semantic logic. E.g. see [van Benthem *et al* 1997] and [Duží *et al* 2010], for two distinct theoretical approaches emerging from this tradition. These logic programs generally study modern *intensional or hyper-intensional logics,* which are much more powerful and complete than predicate logic (which is the basis for SQL and relational and graph database semantics).

CAT4 provides a natural embodiment of high-level intensional logics, including modal, counterfactual and temporal logics. For a key theoretical comparison, we compare it to a leading semantic logic, *Transparent Intensional Logic* (TIL), originating with [Tichý, 1987], see [Duží *et al* 2010], [TIL Website]. This has a rigorous development, and has good claims to be the most complete formal semantic logic yet developed. TIL aims to provide a *complete basis for the semantic analysis of factual linguistic expressions,* while CAT4 aims to provide a *complete basis for a semantic representation of information*, so this comparison is an important verification*.* They both take *compositionality* as a key semantic property. Despite appearing very different on the surface, CAT4 matches TIL closely on key conceptual points. But CAT4 goes outside the traditional method of symbolic logic, providing a distinct representational method, which gives a ready-made programming method. Whereas semantic logics seek to show how to translate natural language (NL) expressions into logical language expressions (like TIL), there is no programmable method for doing this. CAT4 starts with a more fundamental representation than *linguistic expressions*, and can provide a programable method.

Much effort in computer science and logic has been spent searching for higher-order algorithms for AI ("generalized AI") and natural language interpretation ("Universal Grammar"). Leading researchers have believed for fifty years or more that general algorithms must exist, but they have not been found. CAT4 confirms that such algorithms do exist, but they require *a primary representation of information in a suitable form* to be able to be defined. By providing a better primary representation, CAT4 enables more powerful algorithms to be defined.

If it seems unlikely that a *mere switch of representational method* can achieve such dramatic results, an analogy may be drawn with algorithms for arithmetic operations. In the Roman numeral system, algorithms for multiplication and division are very difficult, whereas in the Indo-Arabic decimal system they are simple. Moreover, the Roman system is generally inadequate as a *representation of the natural numbers,* since it is incomplete, and requires *ad hoc* introduction of new symbols for larger and larger numbers, etc. Yet accountants used it for a thousand years, developing specialised algorithms for calculating, without noticing that a simplification of the symbolic representation makes calculations far easier. CAT4 has a similar message: we must reform *the representation of information* before we can take the next step into mastering the algorithms for processing information.

Preliminaries.

The mathematical subject is a special type of point-relation called C2, and extensions of this, culminating in what is called the CAT4 relation. The point-relations associate points with other points, as in a *bi-graph.* C2 captures an intuitive graph structure, in which every *point (vertex* in graph theory*)* has precisely two *joins (out-edges* in graph theory*),* forming a special topology. C2 is more general, and is progressively *restricted* to stronger relations (subsets of C2), called CA2, CAT2, and FCAT2, which impose a more ordered structure. It is *extended* to C3 and C4 by adding two more parent joins. The restriction of C2 to CAT2 gives corresponding restrictions of C3 and C4 to CAT3 and CAT4.

The motivating insight is that CAT2 represents a *general finite hierarchical relation,* representing collections of finite relations (projected into a single first order representation). Collections of tables, functions, meta-data, SQL joins, etc, defining conventional RDBs, can be represented in a single CAT4 table. CAT2 provides the core *intensional relation*, but is not a sufficiently rich structure. The CAT3 extension adds an additional join called the *reference relation*, and CAT4 adds a further join called the *function join.* These are required to fully equip the structure for the proposed semantic interpretation.

We briefly summarise some general concepts of the model. The C4 relation will be seen as a construction from four trees, T_1, T_2, T_3, T_4, over a single shared domain, ***N***. These trees combine in a sequence of constructions to give relations: *C2 < C3 < C4.*

$$C2 = T_1 \otimes T_2 \quad \mathrm{C3} = C_2/T_3 \qquad C4 = C_3/T_4$$

The first relation C2 is the fundamental construction. The two subsequent relations, C3 and C4, are like filters over C2. These are defined as general types of relation. They are instantiated by specific relations, which we will first define over finite integer domains. We use variables or constants: c, c_1, c_2, c_3, c_4, *etc,* to refer to specific relations. Formally:

- Relations of type *C2, C3, C4* are isomorphic to functions from *finite domains* $\boldsymbol{N}$ *of points,* represented by integers: *{0,1,…,N},* back onto *2-, 3-, or 4-tuplets of points of* $\boldsymbol{N}$, respectively.

The relations thus correspond to these mappings:

C2: $\boldsymbol{N} \rightarrow \boldsymbol{N}^2$,   C3: $\boldsymbol{N} \rightarrow \boldsymbol{N}^3$,   C4: $\boldsymbol{N} \rightarrow \boldsymbol{N}^4$

In a relational database setting, we may represent them by three relations (tables):

C2: *ID* $\rightarrow$ *(ID1, ID2)*   C3: *ID* $\rightarrow$ *(ID1, ID2, ID3)*   C4: *ID* $\rightarrow$ *(ID1, ID2, ID3, ID4)*

C2 is the primary relation and we define it first. Note for domain ***N*** the cardinality of points is *N+1,* and the integer variables: *i, j, k, …* range over: *{0,1,…,N}.* We can extend any *C2* relation with domain ***N*** to a *C2* relation with domain ***N+1***, by adding a new point, *i=N+1*.

After defining the structures we need to develop a functional algebra for the system. This ultimately revolves around topological properties of C2, but we can start from first principles, with primitive operators, to illustrate. The simplest example of functional operators are the *parent n-tuple functions* defined:

- Definition. Parent n-tuple functions.
- $P_2(i) = (j,k)$ in a C2 relation c_2 *if and only if (i,j,k)* is in the relation c_2.
- Ditto: $P_3(i) = (j,k,l)$ in c_3, $P_4(i) = (j,k,l,m)$ in c_4.

These functions map points of the domain of *c* to the *n-tuples of c*. The condition that these *functions* exist is simply equivalent to the condition that c_2, c_3, c_4 are *relations*. To get the individual parent points of a points we can define primitive left and right parent operators like:

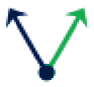
- Definition. Left and Right Parent operators for a C2 relation *c*.
- *L(i) = j in c* if and only if there is some *k* such that *(i,j,k)* is in *c*.
- *R(i) = k in c* if and only if there is some *j* such that *(i,j,k) is in c.*
- (These may be duplicated for the additional joins of C3, C4.)

These are comparable to the *predecessor relation* among natural numbers. However a more general functional algebra is required (just as in arithmetic). We need to represent general relationships between points in the CAT4 relations. Pairs of points in C2 are related through the *chains of points* connecting them, and the algebra reflects a topology induced by chains. We will see there is a general way of representing recursive functions over the network, using general *interior* and *exterior* functions, with their unions and intersections.

We should emphasise the distinction between the *relational view* and the *graph view.* We take the definitions of the structures as *relations* to be fundamental in the axioms, but it is useful to visualise these as *bi-graphs or networks,* and we move freely between the language of *relations or functions,* and the language of *graph theory*. *C2* may be taken as a special class of bi-graphs, formed from a finite set of points (vertices), having exactly two out-edge-types. A C2 graph defines two trees, T_1 and T_2, as mentioned above, and we will later introduce *C3* and *C4* as two further trees mapped onto *C2*. We can speak of a C2 *graph.* However the primary definition of C2 is as a *relation*, which may be visualised as a single table representing the domain and the (2,3, or 4-place) relation, all together.

Defining this as a *relation* ensures the uniqueness of the mapping: $i \rightarrow (j,k)$. By contrast, graphs are represented by two classes representing *vertices* (here *points)* and *edges* (here *joins*). There are no logical constraints on the *number of joins* in a bi-graph. And if we wish to identify two distinct *join types* we have to further categorise the edges. But the *relational* representation of C2 means the double-join structure and two join types are logically ensured.

The table representation is also useful from a database perspective. Relational database management systems (RDBMS) provide generic tools for tables, and may be used as a platform to build practical CAT4 database system. A CAT4 system can be made *scalable* in this form, using generic RDBMS tools and special indexes enabled by the CAT4 topology.

In the graph context, we visualise C2 relations as graphical mosaics, built by adding or deleting discrete elements. C2 or CAT2 graphs are built with only one type of repeated 'mosaic element', viz a point with two joins, pictured like this.  The arrows (out-joins) have to go to other points, which have their own joins. C2 has rules that allow only certain types of mosaics. These are *local rules,* that tell us where we can insert new points in a graph by checking local relations, without referring over the whole graph. This is an underlying motivation for the formulation of the axioms.

We start in Part 1 with an elementary statement of axioms and elementary proofs. We model the structures with classes of finite integer domains. This also useful to visualise the database model, because every point becomes a row in a table, identified by a unique integer, viz. the primary key, here called the *ID*. This treatment provides a first-principle set-theoretic method for modelling the system mathematically, and after we establish the relations in this way we can relate them to graphs, trees, lattices, etc, which can provide more powerful methods of visualisation and analysis. However the foundational treatment from first principles using basic set theory and basic concepts of trees and lattices is effective without needing knowledge of specialised areas of mathematics. The first sections may be taken as exercises establishing the system. As we go on in future parts, we can increasingly dispense with such elementary proofs, but they are needed to begin with.

C2 Platonic Axioms.

We start with Platonic axioms (defining classes), and then propose equivalent recursive axioms (defining constructions). We begin by treating *C2* as a relation on finite integer domains, which are finite classes: ***N** = {0,1,2,…,N}*. These *domains* are defined as sets. They correspond to the naturally ordered sequences, denoted: *(0,1,2,…N).* For the following definitions we assume a *C2* relation is represented by a class of ordered triples: *{(i,j,k)},* with variables *i,j,k* ranging over a finite integer domain: ***N** = {0,1,…,N).* The relation is unique and complete in *i.* We refer to individual triples as *point-relations,* with separate variables: *f,g,h* for these, e.g. *f = (i,j,k).* The constants*: 0 and z = (0,0,0)* are used for the zero-point and its point-relation.

Axioms for C2 over ***N***. A relation: c_2 *= {(i,j,k)}* is a *C2 relation over the finite integer domain: **N** = {0,1,2,…N},* if and only if it is a relation over ***N*** that satisfies these properties:

(Axiom 1) *(0,0,0)* is in c_2

(Axiom 2) For every i≠0 in ***N*** there ***N*** are unique j and k where: *(i,j,k)* is in c_2, and *i≠j* and *i≠k*, and for *j* and *k* there are unique *l,m* and *n,r* where: *(j,l,m)* and *(k,n,r)* are in c_2, and either:

A.	*j=k*	or:	[Case A: identical L and R parents for i]
B.	*l=m=n=r*	or:	[Case B: j,k have identical parents]
C.	*m=n and l≠m and n≠r*		[Case C: LR(i) = RL(i) but j, k are distinct]

(Axiom 3) Each point in c_2 is connected in a chain to *0*.

The first axiom is trivial. The second axiom represents a 'local condition' on points, being determined from the point-relations in the immediate neighbourhood of *f = (i,j,k),* viz. by the two parent point-relations: *g = (j,l,m)* and *h = (k,n,p).* The third is a 'global condition', which requires that all points in the domain of the *C2* relation are connected. We will see that this is also ensured by adding points into the local structure by Axiom 2, in the recursive axioms below.

To complete Axiom (3), a *chain* is defined as an ordered sequence of points.[2]

- Definition. Chains.
    - (i) *<i>* is a chain in a relation *c* for any point *i* in *c*.
    - (ii) *<i,j>* is a chain just in case point *j* is the left or right parent of *i.*
    - (iii) If *<i,j>* and *<j,k>* then *<i,j,k>.*
    - (iv) Generally: if *<i,…,j>* and *<j,…,k>* then *<i,…j,…k>.*

If there is a chain: *<i,…,j> i* is said to be connected in a chain (down) to *j*. C2 axiom (3) means there is at least one chain: *<i,…,0>* from every point to the zero point.

We define the *interior of a point* as:

- Definition. Interior. A point *j* is in the *interior* of *i* just in case there is a chain: *<i,…,j>.*
- The *interior set of i* is the set of all points in the interior of *i.*

We later define analogous *C3* and *C4 chains* and *interiors*. In Parts 1 and 2 we deal mainly with C2, and generally just refer to *chains* and *interiors.* In Part 3 we distinguish between *C2, C3* and *C4 chains* and *interiors*. The *C2 interior* is a fundamental concept and we will deal with it at length.

Note the zero point is defined as the *bottom* (root) of the structure, to conform to the conventional directions in trees, and to the numerical models and the coordinate system we later use. The graph diagrams picture the zero point at the *top of the diagram,* but we should consistently refer to directions between points as the tree directions to avoid confusion.[3]

Axioms 1-3 only define relations on the number domains, ***N***. But the C2 property may extended to all isomorphic relations as follows.

> (Axiom 4) A relation over an arbitrary domain is C2 just in case it is isomorphic to a C2 relation over a finite integer domain: ***N*** = *{0,1,2,…N},* as defined by (1)-(3) above.

Note this is a second-order axiom extending the definition of C2. Axioms 1-3 are definitive of the structure. The following proposition means that *permutations of **N** generate isomorphic C2 relations*.

---

[2] Note *chains* are directed down, while *paths* are more general, and may zig-zag up or down. *Paths* are defined in graph theory as sequences of *edges,* but we use sequences of points.
[3] It may be advisable in future to present the graphs with *0* at the bottom, but it is too late to change this here.

- Proposition 1. If c_2 is C2 with domain ***N***, then any permutation π of elements of ***N***>0 with $\pi(0)=0$ generates an isomorphic relation: $\pi(c_2)$ that is also C2 with domain ***N***.

This is shown later.

CA2 and CAT2 and FCAT2 Axioms.

We now define a stronger relation, called the *CA2 relation.*

Axiom for CA2. A C2 relation is CA2 just in case every interior set is a lattice.

We subsequently introduce the stronger CAT2 relation.

Axiom for CAT2. A CA2 relation is a CAT2 relation just in case the order of the points in rows is consistent for all interior lattices.

We subsequently introduce the stronger FCAT2.

Axiom for FCAT2. An *FCAT2* relation is a CAT2 relation where all *CP's* it contains are *flat*.

These axioms are stated here for completeness, but require the further definitions of a *CP* (*collection point*) and a *flat CP*. These are treated in Part 2. In Part 1 we focus on C2.

C3 and C4 Platonic Axioms.

For C3 we introduce a third join-type, referred to as the *reference join.* This extends the C2 relation. The additional axiom for this is:

(Axiom 5) If: $c_2=\{(i,j,k)\}$ is a C2 relation, an extended C2 relation with point-relations: $c_3=\{(i,j,k,l)\}$ is C3 just in case: (a) The relation: $T_3 = \{(i,l): (i,j,k,l) \,\varepsilon\, c_2\}$ is a tree* (directed rooted in-tree with $0 \equiv (0,0,0,0)$ as the root), and (b) c_3 has no cycles and (c) the point l is not in the (C2) interior of i.

The tree* concept is defined below (Proposition 3). A *cycle* in C3 means a chain through any of the three joins that has the same start-point and end-point. The concept of a *C2 chain* can be extended in the obvious way to chains though any of the three joins, as all joins taken alone define rooted trees with *0* as the common root. We can call these *3-chains* defined on C3*,* to distinguish them from the *2-chains* defined on C2. The absence of cycles in C3 then means that *3-chains have no repeated points.* Note this entails the converse of (c): *the point i is not in the interior of l.*

For C4 we similarly introduce a fourth join-type, referred to as the *function join.* This extends the C3 relation. The additional axiom for this is almost identical to the C3 axiom:

(Axiom 5) If: $c_3=\{(i,j,k,l)\}$ is a C3 relation, an extended C3 relation with point-relations: $c_4=\{(i,j,k,l,m)\}$ is C4 just in case: (a) The relation: $T_4 = \{(i,m): (i,j,k,l,m) \varepsilon c_3\}$ is a tree* (directed rooted in-tree with $0 \equiv (0,0,0,0,0)$ as the root), and (b) c_4 has no cycles and (c) the point *m* is not in the (C3) interior of *i.*

Cycles now refer to chains in all four joins. We subsequently define the extended *CAT3* and *CAT4 relations* in a similar way, from the CAT2 relation. In the remainder of Part 1 we discuss properties of C2.

Graph illustration. Insertions.

We now illustrate the system with graphs. C2 graphs are comprised of *points (vertices) each with two joins (out-edges).* Axioms 1-3 mean there are just three types of parenting structures permitted for a point. If we have a C2 relation in which the *j,k,l,m,n,r-* structures exist as shown below, we may add a point *i* in these three cases (to preserve C2).

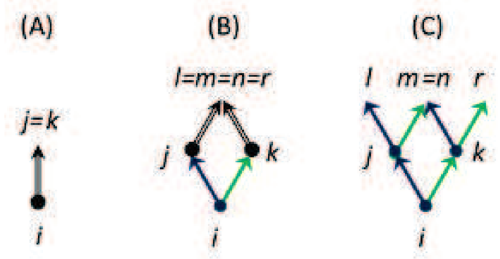


Figure 1.1. Detail showing the three ways a point *i* can be inserted into an existing C2 graph. Double-joins (identical left and right parents) are shown with black double arrows (A). Single left joins are blue and single right joins in green. Note in case C we may have: *l=r* or *l≠r*.

Note that (C) above is not a complete C2 relation as there is no zero point, and there must be further points to complete the C2 relation. But (A) and (B) are complete C2 relations, if we identify *j=k=0* as the zero point in (A), and *l=m=n=r=0* as the zero point in (B).

We can show from the axioms that if a point *i* is inserted in any of these three positions in a C2 relation, the extended relation is C2.

- Proof. Assume c_2 is C2. Show that adding a point *i* in the three cases preserves C2.
- Method. Check each axiom in turn.

- Axiom 1. Given c_2 satisfies Axiom 1, then c_2 with any point added satisfies Axiom 1, since (0,0,0) is in c_2.
- Axiom 2. This has three cases.
    - Case (A). All points in c_2 satisfy Axiom 2, independently of the point *i*. The additional point *i* satisfies Axiom 2A, and thus satisfies Axiom 2.
    - Case (B). All points in c_2 satisfy Axiom 2, independently of the point *i*. The additional point *i* satisfies Axiom 2B and thus satisfies Axiom 2.
    - Case (C). All points in c_2 satisfy Axiom 2, independently of the point *i*. The additional point *i* satisfies Axiom 2C and thus satisfies Axiom 2.
    - Hence in all cases all points satisfy Axiom 2.
- Axiom (3). All points in c_2 satisfy Axiom 3, independently of the point *i*. In each case, *i* parents to a point *j* with a chain *<j,…,0>.* Hence there is a chain *<i,j,…,0>.* Hence all points satisfy Axiom 3.

We can also show that these are *the only ways to add a point*. The following are ruled out.

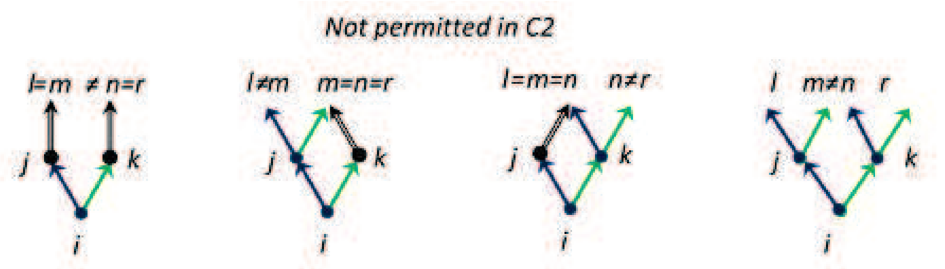


Figure 1.2. Four ways a point *i* cannot be inserted in C2.

Note these four cases and the three above are all the possible cases. This is useful when we need to prove propositions by enumeration of the cases of Axiom 2. Proof is by enumeration of cases.

- Proof. Given any *(i,j,k),* either *j = k* or *j ≠ k.* The first is case (A) and is always permissible. In the second case, *j ≠ k,* there are four possibilities for *j* and *k,* viz (i) both have double joins, (ii) *j* has a double join, *k* has single joins, or vice-versa: (iii) *k* has a double join, *j* has single joins, or (iv) both have single joins. In (i), there are points: *(j,m,m)* and *(k,n,n),* and two possibilities: either *m=n* or *m≠n.* If *m=n* this satisfies (2A), if *m≠n* this fails to satisfy Axiom (2). Similarly, in case (iv), there are two possibilities: either *m=n* or *m≠n.* If *m=n* this satisfies (2A), if *m≠n* this fails to satisfy Axiom (2). Neither (ii) nor (iii) satisfies axiom (2).

Proposition 1. Permutations of C2 are C2.

Proposition 1 means that permutations* of ***N*** generate isomorphic C2 relations.

- Proposition 1. If c_2 is C2 with domain ***N***, then any permutation π of elements of ***N*** with $\pi(0)=0$ generates an isomorphic relation: $\pi(c_2)$ that is also C2 with domain ***N***.

Note these permutations* over the full set ***N*** are defined with: $\pi(0)=0$, i.e. the zero point is always numbered *0.* We prove this proposition directly from Axioms (1) to (3). This shows those axioms include all C2 relations over all finite domains ***N***. It does not matter how we number the points in a C2 relation, except the zero point is always *0.* The C2 properties that interest us most are invariant under these permutations.

- Proof. Assume c_2 is C2 with domain ***N***, show that a permutation $\pi(c_2)$ is a relation with domain ***N*** that is also *C2*, i.e. satisfying Axioms 1-3.
    - Axiom (1). $\pi(0) = 0$, by definition. By Axiom (1), c_2 contains $(0,0,0)$. Hence $\pi(c_2)$ contains: $\pi(0,0,0)=(\pi(0),\pi(0),\pi(0))=(0,0,0)$. Hence Axiom (1) holds for $\pi(c_2)$.
    - *Axiom (2).* Consider a permutation, π, on elements of ***N*** excluding 0. By definition this is a 1-1 invertible function. Note the following equivalences, which follow from the definition of the permutation:[4]
        - c_2 contains (i,j,k) just in case $\pi(c_2)$ contains $\pi(i,j,k) = (\pi(i),\pi(j),\pi(k))$.
        - $i=j$ just in case: $\pi(i)=\pi(j)$.
        - $i\neq j$ just in case $\pi(i)\neq\pi(j)$.
        - For a given *i*, there is a unique *j,k* in ***N*** with (i,j,k) in c_2 just in case there is a unique $\pi(j),\pi(k)$ in ***N*** with $(\pi(i),\pi(i),\pi(i))$ in $\pi(c_2)$.
    - Suppose Axiom 2 holds for a point *i* in c_2. We now show it also holds for $\pi(i)$ in $\pi(c_2)$. Substituting the equivalences above into Axiom 2 entails:
    - Use Axiom 2 for $\pi(c_2)$. For every $\pi(i)\neq 0$ in ***N*** there are unique $\pi(j)$ and $\pi(k)$ where: $(\pi(i),\pi(j),\pi(k))$ is in $\pi(c_2)$ and $\pi(i)\neq \pi(j)$ and $\pi(i)\neq \pi(k)$, and for $\pi(j)$ and $\pi(k)$ there are unique $\pi(l)$, $\pi(m)$ and $\pi(n)$, $\pi(r)$ where: $(\pi(j), \pi(l), \pi(m))$ and $(\pi(k), \pi(n), \pi(r))$ are in $\pi(c_2)$, and either: $\pi(j) = \pi(k)$ or: $\pi(l)=\pi(m)=\pi(n)=\pi(r)$ or: $\pi(m)=\pi(n)$ and $\pi(l)\neq\pi(m)$ and $\pi(n)\neq \pi(r)$

[4] E.g. [Durbin, 1992], Ch 7,8.

- Since π is 1-1 invertible on *{1, 2, …, N},* the quantification range of $\pi(i)\neq 0$ in ***N*** is identical to quantification range of *i≠0 in **N**,* and similarly for all the other variables, *j,k,l,m,n,r.* We could define new variables: $\pi(i) = i'$, $\pi(j) = j'$, etc, and rewrite the previous condition in terms of these. Then because the range of each variable *i,i',j,j',k,k', etc,* is identical, *and there are no conditions on relative order of i,j,k,l,m,n,r,* in Axiom 2, the previous condition is identical to Axiom 2 for the relation $\pi(c_2)$.
- Since Axiom (2) holds for simple permutations, it holds for general permutations, because every permutation is identical to a sequence of simple permutations.
- Axiom (3). For any i, there is a chain: *(i,j,K,…,0)* in c_2, by Axiom (3). Hence there is a chain: $\pi(i,j,k,…,0) = (\pi(i),\pi(j),\pi(k),…,0))$ in $\pi(c_2)$. Hence $\pi(i)$ satisfies Axiom(3) in $\pi(c_2)$. Because $\pi(i)$ is an injection, every point in $\pi(c_2)$ is given by some: $\pi(i)$ where *i* is a point in c_2. Hence every point in $\pi(c_2)$ satisfies Axiom (3).
- The domain of $\pi(c_2)$ is a permutation of ***N***, hence it is identical to ***N***.

All that is essential for this result that there are no *order conditions* in the axioms for the C2 integer models over ***N***, except for the uniqueness of the *0* point at the top, so the order in which we number points *i>0* is immaterial to the C2 property. To illustrate, suppose we add a further condition, restricting the *C2 integer models over **N*** to only those with a numeric ordering where *parent points always have smaller numbers than child points*, viz: *i>j and i>k whenever: (i,j,k).* (This ordering is always possible: see recursive axioms below.)

Then Proposition 1 would be false, because not all permutations of C2's satisfy this order condition. E.g. *{(0,0,0),(1,0,0),(2,1,1)}* satisfies it, but its permutation: *{(0,0,0),(2,0,0),(1,2,2)}* does not. This extra order condition defines a smaller class than C2 over ***N***. However, Axiom (4) ensures all permutations of C2's over ***N*** are included in the class of C2 relations. The class defined by Axiom 4 is larger than that defined by Axioms 1-3. Axioms 1-3 are first-order and define a class of specific C2 relations over the numeric domains ***N***, while Axiom 4 is second order, and defines a more general class. We call them both C2 for simplicity, and disambiguate when necessary.

Proposition 2. Duplicate points preserve C2.

An important proposition is that any existing point except *0* can be duplicated in a *C2* relation, meaning a new point can be added with the same parents as any existing point.

- Proposition 2. Duplicate points. If c_2 is C2 with domain ***N***, and: *(i,j,k)* is in c_2, with: *i>0*, the extended relation: c_2 *U {(N+1,j,k)}* is C2.

The point *N+1* is said to be in the identical *position* as the point *i,* meaning it has the same parents.

- Proof: Since c_2 is C2, all the points in c_2 satisfy the axioms (1)-(3), and it only remains to prove the additional point-relation: *(N+1,j,k)* satisfies (2) and (3). It satisfies (2) in the same way as *(i,j,k),* viz. *j=k* or *m=n≠l≠r.* It satisfies (3) in the same way as *(i,j,k),* since they have identical parents, and hence have the same chains upwards, and hence both connect to 0.

We think of points *j,k* graphically as the *parents* or defining the *position* of point *i* in the graph. This property means we can duplicate as many additional points as we wish in the same as an existing point in a C2 relation. (Note if we duplicate *0* we get: *(N+1,0,0),* a new point, but not a zero point.)

Proposition 3. C2s are two trees.

Axiom 3 ensures that the entire C2 graph is connected and there are no cycles. Without it we could have a structure like this.

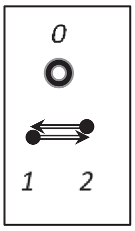


Figure 1.3. This relation is: *{(0,0,0),(1,2,2),(2,1,1)}.* This satisfies Axioms 1 and 2 but not 3. Points 1 and 2 both satisfy Axiom 2A. There is a cyclic chain: *<1,2,1,2>.* There are no chains: *<1,0>* or *<2,0>.* This represents two disjoint sets of points. By Axiom 3 this is not C2.

Axiom 3 can be put in several equivalent forms, including:

- Every point *i* has at least one chain to 0.
- Every maximal chain from every point ends at 0.
- Every maximal chain of left parents and of right parents from every point ends at 0.
- There are no cycles (except *<0,0>* which is explicitly permitted*).*

Or we can show an equivalent condition in terms of trees. The key proposition is:

- Proposition 3. Every C2 is a combination of two trees. For any c_2 that is C2:
  T_1 *= {(i,j): (i,j,k)* ε c_2*}* is a tree*   Left-tree relation

$T_2 = \{(i,k): (i,j,k) \varepsilon c_2\}$ is a tree*   Right-tree relation

Here *tree** means a *tree relation* for a *directed rooted in-tree,* with all its joins (edges) pointed towards the root, *0*, except* that the root *0* is self-parenting.[5] Note a tree in graph theory has a broader condition, viz.

- Definition. "In graph theory, a tree is an undirected graph in which any two vertices are connected by exactly one path, or equivalently a connected acyclic undirected graph". [Wikipedia; *Tree (graph theory).*]

However the common use in applied sciences is a rooted tree.

- Definition. "The various kinds of data structures referred to as trees in computer science have underlying graphs that are trees in graph theory, although such data structures are generally rooted trees. A rooted tree may be directed, called a directed rooted tree, either making all its edges point away from the root—in which case it is called an arborescence or out-tree —or making all its edges point towards the root—in which case it is called an anti-arborescence or in-tree." Wikipedia, https://en.wikipedia.org/wiki/Tree_(graph_theory).

We use tree in the latter sense, also restricted to finite trees. We define it in our relational terms, and by *tree* we mean a *finite directed rooted tree relation,* which has a unique point, *0,* as the root.

- Definition: *tree* relation*. A relation: $T = \{(i,j)\}$ over a finite domain $\mathbf{N} = (0,1,2,..,N)$ is a *tree* relation* just in case:
  - (i) Every point has a unique parent. For each *i* there is a unique *j* such that: *(i,j)* is in *T*.
  - (ii) There is a single maximal point *0*, with a (unique) chain: *<j,…,0> up* from every other point. (Making it connected, rooted, anti-arborescent: directed *into* the root *0*.)
  - (iii) There are no cycles, except* in the root point: *(0,0,0)*. Equivalently: any two points are connected by exactly one path, excluding* adjacent repetitions of the root point.

We now prove that the left-join and right-join structures are separately trees*.

- Proof. Because the axioms are symmetric between T_1 and T_2, we need only prove it for T_1. For an inductive proof, we show: (a) the zero point alone is C2 and the T_1 it generates is a

[5] This allows us to take the relation: *{(i,j,k)}* to be over the single domain *{0,1,..,N}* of all points, and to be a complete function. We might alternatively choose an improper point-relation for *0,* using null values in the 3-tuple: *(0, ,).* But using *(0,0,0)* is simpler.

tree*, (b) Suppose *T_1 = {(i,j): (i,j,k) ε c_2}* is a tree* for some *c_2* that is a C2 relation over ***N***. Then any extension, call it *c_2'*, that is a C2 relation over ***N+1*** also generates a tree: *T_1' = {(i,j): (i,j,k) ε c_2'}*.

A further important lemma is required to complete this inductive proof. This is Proposition 4, below. It says that all C2 relations over ***N*** can be formed recursively by starting with the zero point, and sequentially adding *N* points (according to Axiom 2). We show this next when we introduce equivalent recursive C2 axioms.

- Proof of Proposition 3 for the *T_1* tree.
- (a) The zero point alone trivially satisfies the C2 axioms, by inspection.
- (b) There are exactly three positions in which a new point *N+1* can be added to *c_2,* viz. cases A, B, C above. In each case, the extension preserves the *T_1* tree structure. In each case:
    i. The additional point *N+1* has unique point relation: *(N+1, j, k*), for some *j,k* in ***N***. Hence the relation *T_1'* is extended by: (*N+1,j*), and *j* is the unique parent, as there are no other point-relations containing *N+1*.
    ii. The new point *N+1* is in a chain: *<N+1,j>,* and *j* has a chain: *<j,0>,* so there is a chain: *<N+1,j,0>.* All other points have chains to 0, as no other parents in *c_2* are changed. Hence all points parent to 0, and the zero point remains maximal.
    iii. A cycle containing *N+1* requires a chain like: *<N+1, j, k, …,r, N+1>.* This requires a relation: *(r,N+1,k),* but there are no point-relations involving *N+1* except the *(N+1,j,k)* relation. The extension cannot introduce a cycle containing the *N+1* point as there are no points: *(i,N+1,k).* No other cycles are introduced as no other point relations are changed.
- The same proof holds for the *T2* tree, as the axioms are symmetric w.r.t. the two parents.

This shows Proposition 3 holds as long as all C2 relations can be formed recursively by extending the zero relation *{(0,0,0)}* by adding a discrete series of additional points, shown next.

Hence we can separate the C2 structure into two distinct trees, *T_1,* and *T_2,* with the same domain. These trees are determined uniquely. Conversely, we may regard the C2 structure as a combination of two trees, symbolised: $T_1 \otimes T_2$. The operator ”$\otimes$” combining two trees into a C2 relation is not deterministic, and represents a class: $T_1 \otimes T_2 = \{c_2\}$ of all C2’s that have these trees. I.e. we may generally form two or more non-isomorphic C2 structures from two given trees. Equivalently, two distinct C2 relations can have the same trees.

A key point of interest is the condition for two trees to be able to form a C2 relation. What range of C2's is possible by combining two given trees? In a database, because trees are simpler than C2s and can be represented as linear strings, the two trees of a C2 can be analysed, recorded, transmitted, and used to reconstruct a C2. If this was deterministic it would be a very compact way of recording C2 structures, but it is not. However this indeterminism can be used as a pattern-filtering device. There is a deterministic algorithm generating the two trees from any C2 relation: $c_2 \rightarrow T1\&T2$. But there are multiple reconstruction algorithms possible to determine a C2 back from the trees: $T1\&T2 \rightarrow$*(reconstruction algorithm)* c_2'. A class of C2's that all map to a single c_2' under a given reconstruction algorithm: $c_2 \rightarrow T1 \ \& \ T2 \rightarrow$ *(reconstruction algorithm)* c_2' is an equivalence class w.r.t that algorithm, representing a kind of structure. The space of such algorithms provides a means of *normalising relational structures*, and represents special functions mapping C2's to a smaller space of C2s. This is a point of interest for data systems that we discuss later.

C2 Recursive axioms.

(1)-(3) are proposed as the primary axioms for C2. This is a Platonic definition, stating an abstract universal condition for C2. But there is an effective finite procedure for constructing any C2, starting from the zero point, without any deletions or insertions. This is the natural recursive definition:

Recursive Axioms for C2.

(1) $\{(0,0,0)\}$ is C2

(2) For any c_2 in C2 with domain: $\boldsymbol{N} = \{0,\dots,N\}$, an extension defined by: $c_2 \cup \{(N+1,j,k)\}$ is also in C2 with domain: $\boldsymbol{N}^+ = \{0,\dots,N,N+1\}$, just in case j,k are in $\boldsymbol{N}$ and either:

   a. $j=k$ or: (Case a: identical parents)

   b. (j,l,m) and (k,n,r) are in c_2 and: $l \neq m = n \neq r$ (Case b: identical LR and RL GPs)

(3) The class generated by repeated applications of (2), starting with (1), is the class of recursive C2 relations.

As before, we generalise to isomorphic relations, with identical Axiom 4 to that above. Note that C3 and C4 recursive axioms are defined in a similar way, and we need not be concerned with them until later. But equivalence of the *Platonic* and *Recursive* formulations for C2 is a key lemma. First we briefly illustrate with examples.

Graph illustrations.

Below are graphs of the first few types of C2's that can be generated inductively.

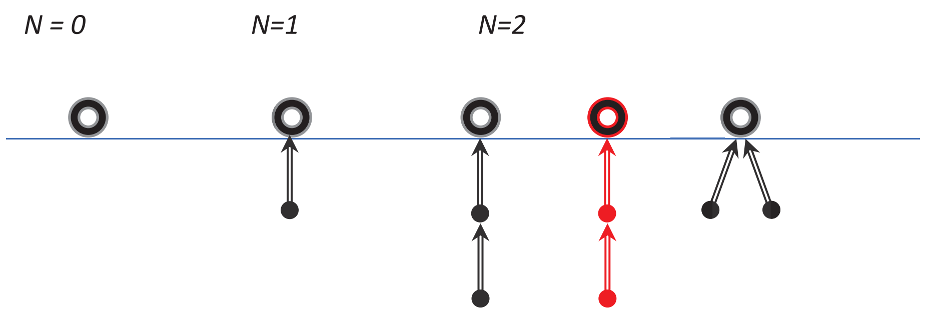


Figure 1.4. Graph illustrations. C2 relations over ***N*** for *N = 0, 1, 2*. Note there are two isomorphic relations for *N=2,* with one indicated in red. (We cannot see these are distinct without the numbers).

- The zero point-relation: *{(0,0,0)}* is the only C2 for *N=0*.
- *{(0,0,0),(1,0,0)}* is the only C2 for *N=1.*
- Note there are two distinct C2's generated recursively for *N=2,* viz: *{(0,0,0),(1,0,0),(2,1,1)},* and: *{(0,0,0),(1,0,0),(2,0,0)}.*

Note there is a third distinct C2 for *N=2,* viz: *{(0,0,0),(2,0,0),(1,2,2)},* in red in the diagram, which is *not* generated by (1)-(3) above. But it is isomorphic with the first graph, under the permutation: *1* ⇔*2,* and belongs to the general C2 class by Axiom 4. It is evident it is not in the recursive C2 class, because there is a point *(i=1,j=2,k=2)* where *i<j* and *i<k.* There is a fairly obvious proposition that all point-relations generated recursively by those rules (allowing only additions, not renumbering of existing point relations) must produce point-relations: *(i,j,k)* where *i>j* and *i>k.*

We would normally say there are only two types of C2's for N=2, because the C2 properties are independent of the numbering of points produced by our recursion. It is the permutation-group properties that we are interested in. This indifference to specific numbering allows us to assume special numbering conventions for some purposes, e.g. the hierarchical order just noted.

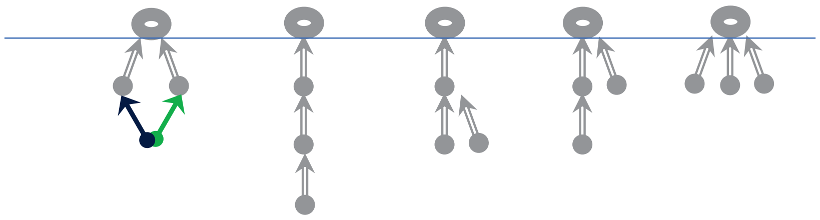


Figure 1.5. Graph illustrations. *N = 3.* By induction from *N=2*, there are exactly five distinct types of *C2's* for *N=3.* In the diagram above, we represent points as indistinguishable from each other. However, the points can be labelled *1,2,3* in several permutations in each structure; and each represents a distinct *C2* relation over *{0,1,2,3},* as we saw with the *N=2* example.

- Definition. Two (recursively defined) *C2* relations are the *same type* just in case they are isomorphic.

Proposition 4. Equivalence of Recursive and Platonic axioms.

We now demonstrate Proposition 4, the lemma required to complete Proposition 3.

- Proposition 4. All C2 relations over ***N*** can be formed recursively by starting with the zero point and sequentially adding *N* points according to the insertion Axiom (2), and then including all their permutations.
- Proof: Equivalence of the Recursive and Platonic axioms.

Start with any C2 that obey the Platonic axioms. First we permutate ***N*** so *i > j* and *i > k* for every *(i,j,k)* in the relation. We can always do this because:

- Proposition. Recursive Ordering. Every C2 can be permutated so *i > j* and *i > k* for every *(i,j,k)* in the relation.
- Proof: Axiom 2 means all recursive C2's are ordered in this manner, because every *(N+1,j,k)* added has *j,k* in ***N,*** so *i,j <N+1.* Axiom 4 means each C2 over ***N*** is isomorphic to some recursive C2.

This means we can define a process of *deconstructing a C2* by removing points sequentially, until we return to the zero point.

- Proposition. With a recursive ordering, we can always remove the highest numbered point, *N,* from a C2 over ***N,*** and retain a C2 over ***N-1***.

- Proof: ***N*** can be generated recursively from a C2 over ***N-1*** by adding the point *N.* Therefore removing *N* returns us to a C2.
- Proposition. Since we can always remove at least one point from any C2, and *N* is finite, there is always a finite process that produces a finite series: $(c_2^N, c_2^{N-1} \ldots c_2^0)$ of C2's, that removes all points, except the zero point.

Note any possible deconstruction process is the reversal of a possible construction process.

- Proposition. There is always at least one point, *N,* that is not a parent, except in the Zero relation: *{(0,0,0)}.* The last point inserted, *N,* does not appear in the relation anywhere as a parent: *(i,N,j) or (i,j,N),* only in its own point-relation: *(N,i,j).*
- Definition. We call the class of all such points the *tips (of the relation).*
- Proposition. Tips of a current C2 can be removed in any order, preserving the C2 property.
- Proposition: Every order in which it is possible to deconstruct a C2 corresponds to an order in which it is possible to construct it.
- Proof: renumber the points in the descending order of removal, from *N* downwards, until you get to 0. This is possible because each renumbering is a permutation. Reconstruct the tree by reinserting point-relations in the new order, *1, 2, …, N,* with the permutated point relations. Every *n* inserted as *(n,i,j)* must preserve C2 since it is identical to the C2 from which *n* was removed in the deconstruction.

Proposition 5. Duplicate C2s.

Previously we were only inserting single points, we now consider combining by inserting an entire C2 relation into another. This is always possible because:

- Proposition 5. If c_2 and c_2' are two distinct C2's, over ***N*** and ***M*** respectively, and *j,k* are any points in c_2 in which it is legitimate to insert a new point (by Axiom 2), then a new C2 relation we may symbolise as: $c_2'' = +(c_2,c_2';j,k)$ can be constructed as follows.
    - Renumber the points *1, … M* in c_2' to: *N+2, … N+M+2.*
    - Renumber the 0 point in c_2' to: *N+1* and alter its point-relation in c_2' to: *(N+1,j,k).*
    - Take the union of this relation with c_2.

This inserts a whole C2 in a position *j,k.*

- Proof. Show the new relation: $c_2'' = +(c_2,c_2';j,k)$ satisfies the first axiom set, given the assumptions that c_2 and c_2' are C2 and *(x,j,k)* is a valid position in c_2. The points-relation in the renumbered c_2' results from a 1-1 mapping (injection) of the integer domain: *(0,…M)* onto the integer domain: (*N+1,…N+M+2*), viz by: *π(j) = j+N+1,* and all point-relations are isomorphic, except for the original zero point-relation, which is changed after the domain mapping from: *π(0,0,0) = (N+1,N+1,N+1)* to the insertion point: *(N+1,j,k).*

- We first need to show that the altered point-relation for point *N+1* obeys the axioms, (since this is all that has really changed). By assumption, a point can be inserted in the position *(j,k),* so *N+1* satisfies Axiom 2.
- Remaining points must also satisfy Axiom 2.
  - All points now joining directly to *N+1* have double joins, since it was originally the zero point, and double joins are always valid, independent of the *N+1* parents.
  - All points below these depend for their parent conditions (Axiom 2) only on point-relations up to points in (*N+1,…,N+M+2)*, and remain valid.
  - All points in the original C2 are unchanged.
- Hence all points obey Axiom 2.
- All points in *(N+2,…N+M+2)* now have chains to *N+1,* as it was originally the zero point, and *N+1* has a chain *<N+1,…,i>* to *i* which has a chain to *0* so every point has a chain to *0*. Hence Axiom 3 is satisfied.

This means we can readily insert or remove entire C2 sub-networks from a C2. Note there are only two types of insertion: by a double join or by two separate joins. Here we illustrate the classic form of a *C2 interior lattice, or C2 leaf,* representing a simple type of C2, being inserted into a copy of itself, in two different ways.

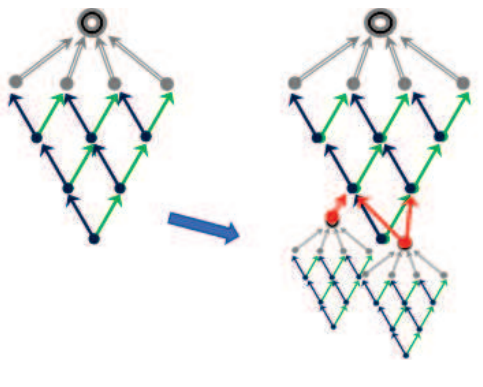


Figure 1.6. Inserting a C2 into another C2.

Once a C2 is inserted, its original zero point becomes the *collection point of its sub-graph* (red) in the larger C2 graph. Note that none of its points can be joined a chain to any points in the original

C2, and all chains go through the *collection point* (original zero point). Consequently, iterative construction of C2s by inserting C2s into each other can only make *trees of C2s*.

Interior and Exterior Relations and Sets.

We now introduce the concepts of the *Interior* and *Exterior* of points.

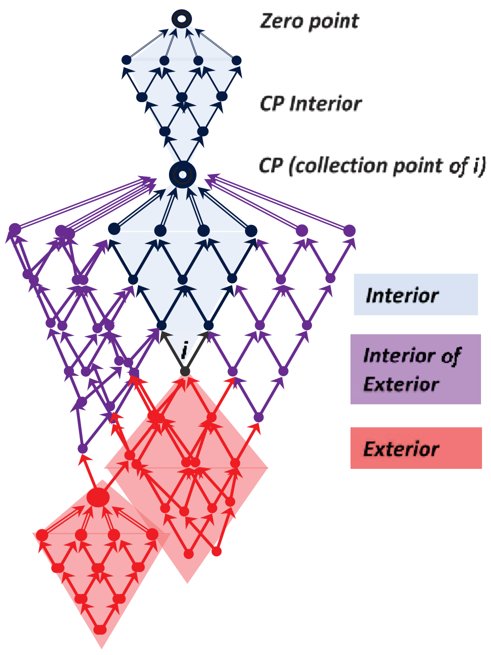


Figure 1.7. Illustration of the global structure around a point *i.* This illustrates a CAT2 or FCAT2 graph (flat lattices), but the same topology applies to C2.

The Interior of *i* is the blue part of the graph at the top of the diagram (which is *below i* in the trees), which includes the *CP* or *collection point* of *i,* and the interior of the *CP,* finally stopping at the zero point. There may be any number of CPs before we reach the Zero point. The Exterior is the red part of the graph at the bottom (which is *above i* in the trees). The Interior of the Exterior is the interior of all the Exterior points (including the interior of *i*).

This illustration is for a typical *CAT2* graph, introduced in Part 2, with ‘flat’ lattice structures. There may be ‘circles’ (horizontal repetitions of points) in C2, which will be banned in CAT2, but the topological concept of interior-exterior applies to C2. This topology is why we see C2 as the general structure before narrowing down to CAT2.

The Interior-Exterior duality is the central concept. Note the Interior has a single point in every position, while the Exterior is branching. The Interior of any point or any collection of points is C2.

Interior and Exterior intersections determine relations between points, and generating these will be the key to an efficient database method.

- Definition. The *Interior Relation,* or just *interior of a point i* is the class of all point-relations for points below and including *i* in the graph.
- Definition. A point *k* is below point *j* if there is a chain: <*j…,k*>. With recursive numbering, *if k is below j then j>k.* The lowest point is 0.
- Alternative Definition. The interior set of a point *i* is the set of all the end-points *j* of all its chains: <*i,…,j*>.

INT(i) = {j: <i,…,j>}

- Recursive Definition. If point i has the point-relation: *(i,j,k) in c_2,* the *interior of i in c_2* is the union of *{(i,j,k)}* and the interiors of *j* and *k*.
- The interior of *0* is the zero point.
- Definition. The *Interior Set* is the set of points with point-relations in the *Interior Relation*.

(We can normally refer interchangeably to the *set of interior points* and *set of interior point-relations,* but they are technically distinct.)

- Definition. The *Exterior Relation,* or just exterior of a point i is the inverse of the Interior Relation. *$i \in EXT(j) \Leftrightarrow j \in INT(i)$.*
- The exterior set of *i* is the set of all points above and including *i.*
- Proposition. The exterior of a point is the class of all points containing that point in their interior.
- Proposition. If a chain <*i,…,j*> exists then: *$i \in EXT(j)$* and *$j \in INT(i)$.*

The *INT* and *EXT* functions are inverses of each other, but they are very asymmetric, reflecting the hierarchical asymmetry of the C2 relation. The *interior* is much simpler than the *exterior.*

C.f. Analogy for trees: the *interior of a point i in a tree* are the points in the *chain: <i,…,0>* from the point to the root (a simple 1-dimensional chain), while the *exterior of a point in a tree* is the *tree* above the point, which is normally branching.

We prove two key properties of the interior next.

- Proposition 6. The interior of any point in C2 is C2.
- Proposition 7. The interior of a point in C2 is a lattice just in case there is no circle in its atomic sequence. Equivalently: just in case all points in its interior positions are distinct.

Graph illustrations. The interior position lattices.

There are a large number of possible C2 graphs that can join *below* any given point – indeed any C2 graph – and the number of possible graphs increases rapidly with *N*. But there are few types of graphs that join above a point. Their structures conform to this sequence of graphs.

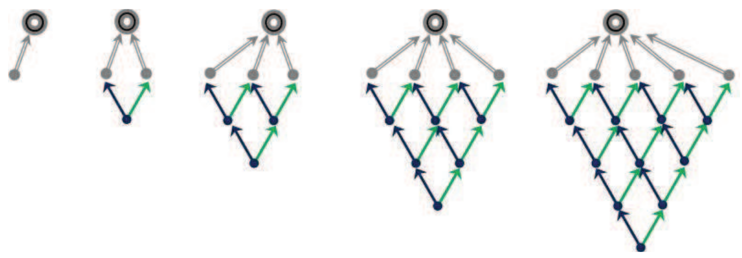


Figure 1.8. Lattices of Interior Positions. The point at the bottom of each graph is the point *i,* and the graph is the *lattice of interior positions of i, up to the first collection point (CP).* The CP is the first point that acts like a zero point for *INT(i).* The *Atomic points* are in the first row below 0.

Table: counts of atoms and points.

A	*1*	*2*	*3*	*4*	*5*	*… A*	*Atomic points*
Point Count	*2*	*4*	*7*	*11*	*16*	*… A(A+1)/2 +1 → A²/2*	
N	*1*	*3*	*6*	*10*	*15*	*… A(A+1)/2*	

These graphs show the *lattices of interior positions* that are possible, determined by the binary branching structure of the point-relations upwards. The reason these are the only *parenting structures* for interior relations is because:

- Proposition. A double-joined point *j* and a single-joined point *k* cannot be parents of a third point, *i*. (No mixed parents).
- Proof. The two cases of Axiom 2 require j, k to be either both double-joined (to a common grandparent, the collection point), or both single-joined, with the shared internal grandparent: *LR(i) = RL(i).*
- If we go up the interior from a given point *i*, putting in the parents at each level, either all the parents in a level (a row) are single-joined upwards to the next level, or all the parents in the row are double-joined upwards. Otherwise there must be a position in the row where points must transition from double-joined to single-joined, and a point must have mixed parents, which is ruled out by the proposition above.

- The grand-parenting rule means that instead of a full *binary tree graph* upwards, having $N = 2^{(A-1)}$ points, the parent position structures are the *flat binary lattices,* with $N = A^2/2 + A/2$.

Graph illustrations. Point graphs.

We must emphasise that points in these positions, for a specific relation, can be repeated, so these are not all the C2 interiors. Repetitions give what we call *circular relations*. E.g.

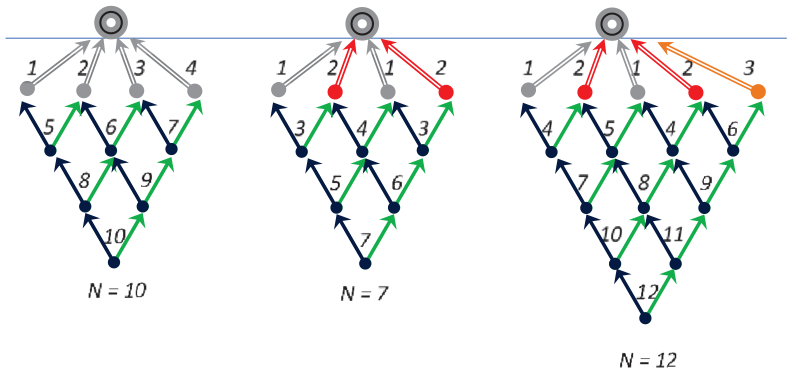


Figure 1.9. Points *1* to *N* in positions in the interior lattice for a specific C2 interior relation gives the *point graph.* Points in different positions are not necessarily distinct. The left graph is the *flat C2 interior lattice* for *A = 4* over domain *N=10*. There are no repetitions of the *atoms,* the points in the atomic row: *(1,2,3,4).* The middle graph has a repetition or *circle* in the atomic row: *(1,2,1,2).* It is the *A=4* interior, but the number of *distinct atoms, D,* is only 2. The right has *A=5*, but *D=3*.

Note the circular interior graphs can be illustrated more compactly using only unique *points*.

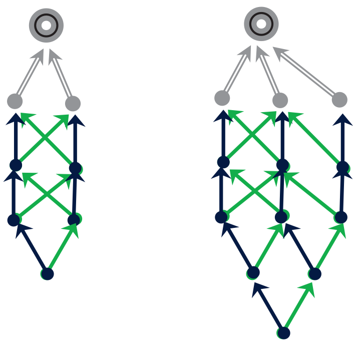


Figure 1.10. These are compact diagrams of the circular interior graphs above. Note the left and right joins are green and blue respectively, but the order in the diagram is reflected for some points. E.g. points *(3,1,2)* and *(4,2,1)* are reflections in the first graph.

Note these *circular C2s* are not lattices. E.g. in the first graph, there are pairs (e.g. *3,*4) having multiple LUB (1 and 2) and HLB (5 and 6). These must be unique for a lattice. A lattice has a unique *meet* and *join* for each pair (or subset) of points. We state some essential propositions.

- Proposition. No circular interiors are lattices.

The opposite of the circular interior C2 is called the *flat interior*, with no repetitions of points.

- Proposition: No repetition in the atomic row entails no repetition in the lattice.

This is the simplest condition for the *flat lattice.*

- Proposition: All flat interiors are lattices.

The compacted diagrams also do not show the paths or structure as clearly as the lattice graphs. We will look at the interior relation graphs as projected onto the position lattice graph, providing a kind of background matrix or *coordinate system*. To reiterate: the *interior position lattice* uniquely determines a *point* in each position (up to the CP), but points may belong to more than one position, so the general function is many to one:

General Interior Points: Positions (many)→ (one) Points

But for the lattice, it is 1-1:

Lattice Interior Points: Positions (one) ↔ (one) Points

The "coordinate system" we develop next helps define an algebra and operations, and it is subsequently related to search methods in a data context.

Coordinate systems for interior lattices

The following coordinate systems are systematic methods of assigning labels to positions in the interior lattices. They are *interior coordinate systems.* They provide references to points relative to an origin. An *exterior* coordinate system for C2, providing paths looking *up from a point,* is more complex because of the branching structure of the *exterior.* The interior coordinate system is much simpler to start with because of the deterministic structure of *interior position lattice*. Here are four initial possible labelling methods for positions.

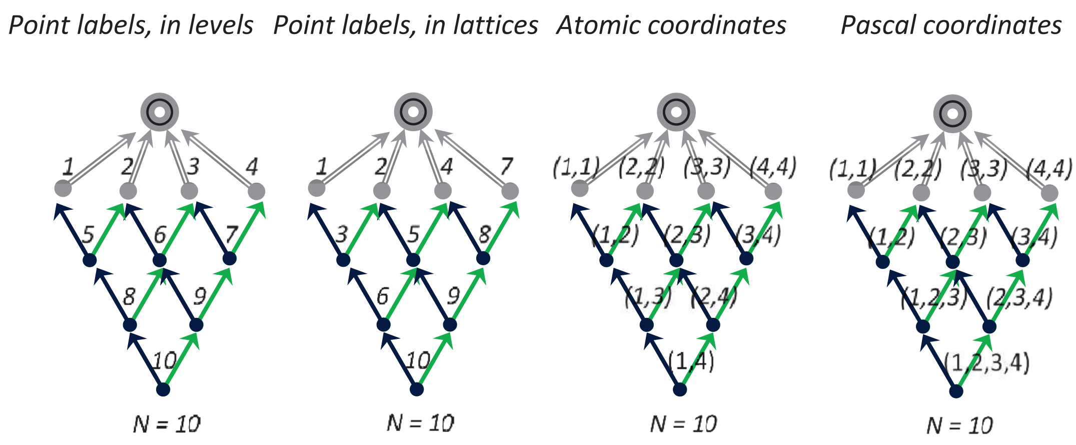


Figure 1.11. Four initial methods of assigning coordinate labels.

From left to right:

- Number positions *1 to N,* by levels. Number each *row* of the lattice sequentially in turn. The *atomic row* is numbered *1* to *N.*
- Number positions by lattice pattern. Number each *interior lattice* in turn, so the series of sub-lattices is contained with the same numbering.
- Atomic coordinates: (a,b). Use end-points of intervals: a≤b in (1,…,A).
- *Pascal coordinates: (a,b,…n).* Record the full ordered string of atomic points. This represents permutations of atomic points in the C2 interior graph.

These all have their uses, but the most useful coordinate system uses the lattice structure and enables *relative position vectors,* with some properties of a vector system.

Interior Coordinates.

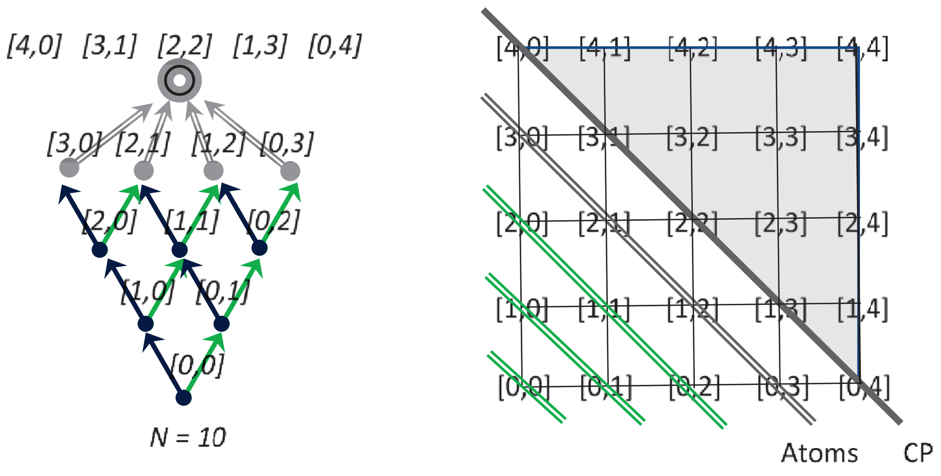


Figure 1.12. Relative vector coordinates for *A=4*. We write these in square brackets.

These coordinates are equivalent to 2-D discrete orthogonal coordinates for an *4x4* triangular grid, with *x* vertical and *y* horizontal, extending up to the diagonal through *(0,4) – (4,0)*. Above this, where: *x+y > 4,* is outside the coordinate space. The origin *[0,0]* is the tip of the lattice, and coordinates are relative to this.

This is a special simple type of lattice, with a strict hierarchical structure of (vertical) *levels.* We might define *Levels* going up simply by: *L = i+j for a point with coordinates: [I,j].* However levels then depend on the origin point (at the bottom in the diagram). It is better to define levels down.

- Definition. Levels in the interior lattice. The CP is Level 0. The atomic points are Level 1. Positions immediately below are level 2, and so on.
- A point with coordinates: *[i,j]* and *A* atomic points is at Level: *L = A-(i+j).*
- E.g. the CP point has Level *0* since its coordinates are: *[0,A],* or more generally: *[i,j]* where *i+j = A.* The origin point is Level *A* since it has coordinates: *[0,0].*

Note this coordinate system stops at the CP level, where it ‘collapses’ back to a single point. This can be relabelled: *[0,0]’* for the new coordinate system for its interior. Hence: *[0,0]’ = [4,0] = [3,1] = [2,3] = [1,3] = [0,4] = [4].* We use *[4]* or *[A]* for the CP point. This is a discontinuity in the coordinate system, representing the topological discontinuity in the relation. Coordinates referring above the CP level have no meaning. They do not correspond to the interior graph above the CP.

Vectors for the interior coordinate system.

These coordinates give a kind of vector system. The *vector* from *[0,0]* to *[x,y]* is written: ***[x,y]*** (bold brackets). These are like vectors from points: *j = [x,y]* to *k = [x',y']* because they have Cartesian coordinate component addition: $\boldsymbol{v}_{j,k} = \boldsymbol{v}_k - \boldsymbol{v}_j$. Thus we define this coordinate system by the position lattice. Since the latter represents paths to all points this is adequate.

- Definition. Relative interior coordinates.
- *[x,y]* for positive *x,y* is a coordinate for the position of a point *j* in the interior position lattice of a point *i* just in case there is a path from *i* up to *j* having *x* left joins and *y* right joins.
- Proposition: *[x,y]* for positive *x,y* is a coordinate in *i* just in case has *INT(i)* has *A atoms* and: $0 \leq x+y \leq A$.

And for vectors:

- Definition. Relative interior coordinate vectors.
- If *j ≡ [x,y]* and *k ≡ [x',y']* are interior position coordinates, the *vector from j to k* is:
    - $\boldsymbol{v}_{j,k}$ = ***[x'-x,y'-y]***
- Proposition. Vectors are independent of the coordinate system for: *j ≡ [x,y]* and *k ≡ [x'y'].*
- Proposition: Vectors ***[x,y]*** may have positive or negative values of *x,y.*

E.g. two points: *[2,0]* and *[0,1]* are related by vectors: $\boldsymbol{v}_{j,k}$ = ***[-2,1]*** and conversely: $\boldsymbol{v}_{k,j}$ = *[j]-[k]* = **[2,-1]**.

This introduction of vectors extends us beyond the original coordinate system. We may allow negative coordinates, which define positions outside the interior. E.g. two points *[j]=[2,0]* and *[k]=[0,1]* are related by: ***[-2,1]***. So In the (dashed) coordinate system with *j* at the origin: *j = [0,0]'*, the point *k* may be assigned coordinates: *[-2,1]'.* This extends the first definition.

- Definition. Relative coordinate vectors.
- If there is a *vector from point j to point k*: $\boldsymbol{v}_{j,k}$ = ***[x'',y'']*** in any coordinate system, and: *j≡[x,y]* is an interior position of *j* in any coordinate system, then: *k ≡ [x''+x,y''+y]* is the coordinate of *k* in that system.
- Proposition. For any point *j,* if there is a vector: $\boldsymbol{v}_{j,k}$ = ***[x'',y'']*** to a point *k,* then the coordinates of *k* in system with *j* at the origin is: *[x'',y''].* Proof: by the definition: *j ≡ [0,0],* and: *k ≡ [x''+x,y''+y] = [x'',y''].*
- E.g. let $\boldsymbol{v}_{j,k}$ = ***[-2,1].*** Use the interior coordinate system for *j.* Then: *k ≡ [-2,1].*

This is the converse of the definition of vectors between points above.

In this extended system, the existence of coordinates outside the interior are not *deterministic,* because the existence of negative coordinates depends on the existence of points in the larger C2.

It is used to relate points across interiors and exteriors.

- Proposition. Two positions related by a vector having one positive and one negative coordinate are outside each other's *interior and exterior*.
- Proposition. Two positions related by a vector having both positive or both negative coordinates are inside each other's interior/exterior.

So far this still only defines vectors between positions in the interior of some common point. We extend this by allowing a transitivity rule:

- Definition. Vector Addition. If j = [x,y] in system with origin k and j = [x',y'] in system with origin k' (i.e. j is in the interior of two distinct points), then : $\mathbf{v}_{k,k'}$ = **[**x'-x,y'-y**]**.
- Proposition. All and only positions with at least one common *atomic point* in their interior have these vector relations under the CP.

Illustrations of vectors.

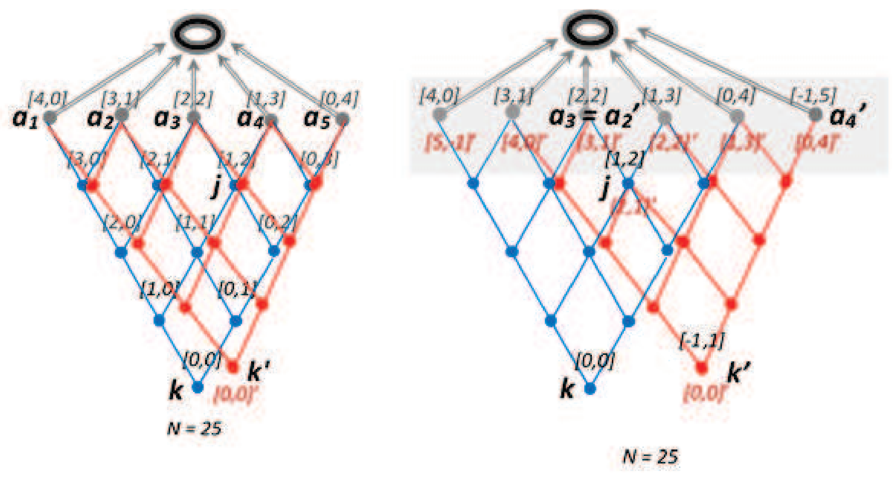


Figure 1.13. Left. Two identical interiors, *k* and *k'*, with the same atomic sequence. Points *[x,y]* in *k* get the same coordinates: *[x',y']'* in *k'*. Right, two partially overlapping interiors. *INT(k')* now has an additional atomic point, *a_4'*, not in *INT(k)*. The interiors join at *j = [1,2]* in *k*, which is *[2,1]'* in *k'*. Points: *[x,y]* in *k* are renumbered *[x-1,y+1]'* in *k'*.

On the left, the interiors of *k, k'* join each other (lcb) at the five points of the shared atoms: *{a_1,a_2,a_3,a_4,a_5}*. On the right, the interiors of *k, k'* join (lcb) at three points: *{a_2, j, a_5}*. The points *a_3,a_4* are shared in the interiors of *k* and *k'*, but are not *joins of the pair: {k,k'}*.

- Definition. We say that two points *k,k' join at point j* just in case *j* is in the shared interior: *INT(k)* ∩ *INT(k')*, and *j* is not in the interior of any other point in the shared interior.
- We write this as: *k^k' = {a_2, j, a_5}*.
- Proposition. If *k^k'* is singular (unique join) then *k, k'* are both in the interior of a single point.
- Proposition. If *k^k'* is singular (unique join) then *INT(k)* ∩ *INT(k')* has no circles.

We may sometimes be ambiguous between referring to *vector relations between positions* or *between points in those positions,* but we should remember that just as *a point* may have more than one position in the interior of another point, it may have more than one vector relation with another point. This occurs only if there are *circular interiors.* When all interiors are flat, the position lattice is isomorphic to the point lattice.

Circular interiors and atomic domains.

The same *points* can occur in different *positions* of the interior lattice for a specific C2 relation. E.g. as we saw above, the point lattice fills in the position lattice, but may have repetitions (circles).

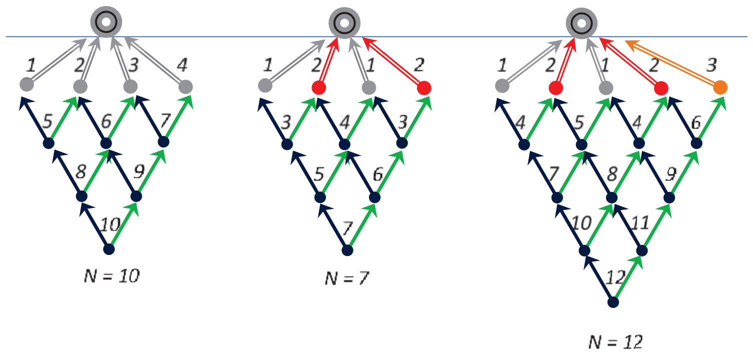


Figure 1.14. The left graph is the flat C2 interior lattice for *A = 4* over domain *N=10.* There are no repetitions in the atomic row, *A.* In this case there can be no repetitions in subsequent rows. The second and third graphs have repetitions in their atomic rows. This allows repetitions in subsequent rows. Here there are repetitions in the second rows.

As the set of atomic points is generally larger than the ordered sequence of the atomic row, we need a reference to the Atomic Domain, called *D*.

- We use *D* as a variable for the number of *distinct points* in the atomic row sequence: $(a_1,a_2,\ldots,a_A)$ of the interior of point *i*.

We can define an indexed set of these points as *the Atomic Domain of i.*

- $D(i) = (d_1,d_2,\ldots,d_D)$ is the ordered class of atomic points in the interior of point *i.*
- The order is taken as the order of the integers representing the points.
- There is an integer function: $\delta(n) = m$ from the index for *D*: $n \varepsilon (1,\ldots,D)$ into the index for *A*: $m \varepsilon \{1,\ldots,A\}$, so that: $d_n = a_{\delta(n)}$.
- Proposition. Each C2 interior for a point *i* uniquely determines an ordered sequence of *A* atomic points: $A(i) \equiv (a_1, a_2, \ldots, a_A)$ in the atomic row, from the class of *D* points: $(d_1,d_2,\ldots d_D)$, with repetitions of points allowed, but no adjacent repetitions.

Note that the class of atomic sequences possible in C2 includes all finite ordered sequences except those with adjacent repetitions. E.g. if *1,2,3,4,…* are points, then: *(1,2,3,4), (1,2,1,2), (1,2,3,1)* are possible, but *(1,1,2,3…)* or *(1,2,2,3,…)* are not.

- Definition. An *atomic sequence* is any sequence from domain *D(i)* with no adjacent repetitions of points.

This allows us to represent any permutation of the atomic domain, by constructing a point with a given atomic sequence in its interior. This is a class with: $D(D\text{-}1)^{A\text{-}1}$ atomic sequences of length *A* from a class of *D* points. E.g. for *D = 3* and *A = 4,* we can have: *(1,2,1,2), (1,2,1,3), (1,2,3,1), (1,2,3,2), (1,3,1,2), (1,3,1,3), (1,3,2,1), (1,3,2,3),* as sequences beginning with *1,* and similar for the two permutations: *1 ⇔2* and *1 ⇔3,* giving 24 seqs, which is: $3*2^3$. (Note the permutation: *2 ⇔ 3* is generated by the first two permutations.)

Acknowledgements.

Special thanks go to Jeremy Praat, for discussions and encouragement in the early development. Thanks go to Patricia Holster, Bev and Gary Thompson, Johan Kronholm, Peter Rudolf, Nic Conland, for support, discussions and encouragement.

References for Part 1.

Note references here include some references for later Parts.

Bridges, Jane. 1977. *Model Theory.* Clarendon Press.

Carnap, Rudolf, 1947, Meaning and Necessity. University of Chicago Press.

Chang, C.C. and H.J. Keisler. 1973. *Model Theory*. North-Holland.

Church, Alonzo. 1956. *An Introduction to Mathematical Logic I.* Princeton.

Codd, Edgar Frank (June 1970). "A Relational Model of Data for Large Shared Data Banks". *Communications of the ACM*. 13 (6): 377–387. doi:10.1145/362384.362685. S2CID 207549016.

Date, Chris. 2004. (8th Ed.) *An Introduction to Database Systems*. ISBN 0-321-19784-4

Durbin, John R., 1992. *Modern Algebra: An Introduction.* Wiley and Sons.

Duží, Marie. "Intensional Logic and the Irreducible Contrast between De dicto and De re", http://www.cs.vsb.cz/Duží/

Duží, M., Jespersen, B., Materna, P. 2010. *Procedural Semantics for Hyperintensional Logic: Foundations and Applications of Transparent Intensional Logic*. Springer Verlag.

Frege, Gottlob, 1879. Begriffsschrift, eine der arithmetischen nachgebildete Formelsprache des reinen Denkens. Halle a. S.: Louis Nebert. Translation: Concept Script, a formal language of pure thought modelled upon that of arithmetic, by S. Bauer-Mengelberg in Jean Van Heijenoort, ed., 1967. From Frege to Gödel: A Source Book in Mathematical Logic, 1879–1931. Harvard University Press.

Frege, Gottlob, 1884. Die Grundlagen der Arithmetik: Eine logisch-mathematische Untersuchung über den Begriff der Zahl. Breslau: W. Koebner. Translation: J. L. Austin, 1974. The Foundations of Arithmetic: A Logico-Mathematical Enquiry into the Concept of Number, 2nd ed. Blackwell

Frege, Gottlob, 1892. "On Concept and Object." (First published in the Vierteljahrsschrift fir wissenschaftliche Philosophie, 16 (1892).

Holster, Andrew T. 2008-2011. "System and method for representing, organizing, storing and retrieving information." US. Patent Number: 7,979,449. July 12, 2011.

Materna, P. 2004. Conceptual Systems. Berlin: Logos.

Materna, Pavel. 1998. *Concepts and Objects.* Acta Philosophica Fenica, vol. 63, 1998.

Montague, R. 1970. "Universal Grammar". *Theoria* **36,** pp. 373-398.

Montague, Richard. 1973. "The Proper Treatment of Quantification in Ordinary English". *Approaches to Natural Language: Proceedings of the 1970 Stanford Workshop on Grammar and Semantics.* D.Reidel.

Robinson, Ian, Webber, Jim and Eifrem, Emil. 2015. *Graph Databases (2nd Edition)*. NEO4J.

Russell, Bertrand. 1905. "On Denoting", Mind, Vol. 14. ISSN 0026-4423. Basil Blackwell.

Tichý, Pavel. 1971. "An Approach to Intensional analysis", *Nous* **5,** pp. 273-297.

Tichý, P. 1988. The Foundations of Frege's Logic. Walter de Gruyter.

Tichý, P. 2004. Pavel Tichý's Collected Papers in Logic and Philosophy. Svoboda, V., Jespersen, B., Cheyne, C. (eds.), Dunedin: University of Otago Publisher, Prague: Filosofia.

TIL (Transparent Intensional Logic) Website: http://www.phil.muni.cz/fil/logika/til/index.html

van Benthem, Johan and Alice ter Meulen. 1997. *Handbook of Logic and Language.* M.I.T. Press.

Wittgenstein, Ludwig. Philosophical Investigations , 1953, G.E.M. Anscombe and R. Rhees (eds.), G.E.M. Anscombe (trans.), Oxford: Blackwell.

Wittgenstein, Ludwig. Tractatus Logico-Philosophicus (TLP), 1922, C. K. Ogden (trans.), London: Routledge & Kegan Paul. Originally published as "Logisch-Philosophische Abhandlung", in Annalen der Naturphilosophische, XIV (3/4), 1921.