Formal Representation of the Logic Embedded in Legal Language

Michael Heather
School of Law
University of Northumbria at Newcastle, NE1 8ST, UK

B.Nick Rossiter
Computing Science
Newcastle University, NE1 7RU
B.N.Rossiter@ncl.ac.uk

February 17, 1997

Abstract

The processing of normative statements depends heavily on the embedding of higher order logic within natural language. This is a prime component of legal computer science that needs a theoretical basis which can be realized in practice both as good science and as good engineering. Scientific rigour expects the use of mathematics, and the engineering element needs that mathematics to be constructive. Theoretical computer science has recently seen developments in constructive mathematics satisfying these principles with the use of category theory. An example of this formalism is used to represent the normative statement:

John gives Mary the ring and title passes on delivery

and to show by the use of the adjoint functor theorem the integration of law and logic embedded in natural language as needed in legal computer science[1].

About the author

Michael Heather is senior lecturer in law where he has been engaged on research into computers and law since 1979.

Nick Rossiter is lecturer in the Department of Computing Science with particular interests in databases and systems analysis.

Suggested Keywords

Category theory, formal objects, higher–order logic, legal language, adjoint functors.

[1]This report describes work on law, language and geometric logic presented to the Legal Computer Science Working Group 66 and included in the Proceedings of the 17th IVR World Congress of the International Association for Philosophy of Law and Social Philosophy, on *Challenges to Law at the End of the 20th Century* held at the University of Bologna, Italy, in June 1995.

Chapter 1

Language and Logic in Legal Computer Science

On the continent of Europe legal computer science has been elevated to a subject in its own right. For example the Italian phrase for legal computer science *informatica giuridica* has been in use since the 1960s [Lupoi 1970]. In Italy it is a discipline that embraces a wide span of topics connected by the concept of law even extending well beyond the concept of computer science as used in the UK or North America. *Informatica giuridica* is taken to include [Biagioli et al 1995] studies carried out in the legal field applying cybernetics and systems theory, telematics, expert systems, legal office automation, automatic analysis of legislation, hypertext (hyperlaw), artificial intelligence and law, philosophy of law, electronic democracy, integrated intelligent systems and even the law of cyberspace.

It is arguable that hiving off parts of computer science and grafting it on to areas of substantive law leads to the fragmentation of computer science and the legal area loses the benefit of cross-fertilization with other applications of computing. The disadvantage in creating a separate kind of subdiscipline, however, does have the advantage of focussing attention on special attributes or requirements to be found in the application of the computer to law.

Legal computer science (in so far as we are concerned here with the technology and not the law) shares fundamentals found in the main body of computer science. It is a science and needs a rigorous basis. It is also a branch of engineering and requires proper engineering practices. However along with most computer science, legal computer science cannot rely on the direct involvement of a human engineer in applying theory to practice. This is in contrast to the normal methods traditionally available in the older branches of civil, mechanical and electrical engineering where a direct application of human experience can bring theory into line with

practice. Results obtained computationally from a machine must depend on the use of realizable theory as produced in the real–world if they are to have any validity there.

Legal computer science is concerned with computational processing of normative statements in the real world. At first sight this might appear to be restricted to a very narrow branch of computing. However it is quite extensive as it embraces both language and logic as well as law. Language and logic are fundamental to the representation of legal knowledge. Moreover the three subjects law, language and logic have to be treated in an integrated fashion. This suggests a requirement for more than first order logic. A statement of a lawyer like

John gives Mary the ring and title passes on delivery

is a natural language expression containing a normative statement of substantive law as well as normative inferences relying on logic. This prime characteristic of the internal cohesion within these strands of law, logic and language is explored in this technical report. A formal representation for this statement is attempted as a presentation of the basic tools needed for legal computer science.

At the heart of the practice of law is the way that normative logic can be embedded in natural language. Any theory therefore has to cope with the integration of operations across different levels. To satisfy these requirements we have made use of the methods in constructive mathematics that are based on the concept of process and found in category theory. A number of texts on category theory for computing science has been appearing over the last few years [Barr & Wells 1990; Pierce 1991; Pitt et al 1986; Rhydeheard & Burstall 1988]. Category theory has recently been applied to law [Heather & Rossiter 1994a, 1994b, 1995; Karpf 1989; 1991; 1995]. Constructive mathematics is concerned only with knowable truth. This means that it is not possible to use indirect arguments like *reductio ad absurdum* because the rule of *tertium datur* need not apply. Likewise the axiom of choice cannot be relied on.

Legal computer science also has to satisfy the same constructive principles as computer science in general. In terms of formal algebraic language, constructivism is to be found in the case of the Heyting algebra. In terms of logic, constructivism requires conformity with the intuitionistic predicate calculus [Boileau & Joyal 1981; Lambek & Scott 1986] which provides the basis of geometric logic. In the language of categorists we are in the realm of the topos [Barr & Wells 1990; Bell 1988] but we keep to basic categorial concepts. The outcome is that we are restricted to data and information in a closed Cartesian category. We should now outline the components of formal categories.

Chapter 2

Formal Categories and Objects

The form of constructive mathematics to be found in category theory is based not on the set as a fundamental but on the concept of process. This is generally thought of in terms of the arrow and represented by $\longrightarrow$ [Manes & Arbib 1986]. The arrow represents any dynamic operation or static condition and can cope therefore with descriptive/ prescriptive equivalent views. For $A \longrightarrow B$ may be a descriptive action or a prescriptive one. That is a norm. Alternatively it may be a probabilistic relationship. There may be any number of different arrows between the same objects. For the arrow may be thought of as a generalization of verbs.

The arrow can never be free–standing: it must have some source and target, often named domain and codomain respectively. A category is a collection of arrows. The concept of a dual category arises from the view of arrows in the reverse direction.

The arrow is a more effective representation of real–world phenomena. $A \longrightarrow B$ can represent an action from a state A to a state B, an interaction of A with B, for example a product of A with B, or a type change from type A to type B.

The arrow can represent a more general relationship than the set-theoretic function. Language is concerned with representing more general relationships which exist between real world data. For example in considering verbs as functions, an object is not necessarily a strict mathematical function of the subject.

Domains may have various levels of complexity, that is the domains may themselves consist of arrows at lower levels. Domains and codomains need not have the same level of complexity. Their arrows will then enhance the structure or will simplify, that is a higher level type conversion.

Analogical reasoning using natural logic is an important example of inference as a higher–level type conversion where there is invariance of intension but possibly unrecognizable changes in the extension.

The simplest arrow has null domain and null codomain. This is the identity arrow **1** which, after the category with no arrows at all, forms the next simplest category. In more concrete terms this arrow can be thought of as identifying an object. When there is only one object, it is indistinguishable or what is technically known as *unique up to natural isomorphism*. It is often labelled $\{*\}$. A category with two such objects is sometimes written as **2** but with two or more objects it is possible to distinguish them by arrows between the objects and the identifying arrows may be labelled $1_A, 1_B, 1_C, 1_D$, etc or more simply $A, B, C, D, \ldots$ from the viewpoint of the object as a concrete entity.

This amounts to an object–oriented approach. For historical reasons mainly from the influence of set theory, the emphasis is more often on the objects rather than the arrow. It is important to bear in mind that objects can always be abstractly defined in terms of the arrow.

Conventionally then a category in this context consists of the collection of arrows between objects as shown in Figure 2.1:

$$A \xrightarrow{\quad f \quad} B$$

$$C \xrightarrow{\quad g \quad} D$$

Figure 2.1: Simple Categories

If arrows are like verbs or prepositions, objects are nouns and adjectives represented by categories themselves. Adverbs are natural transformations (below). As with language there is a whole range of possibilities on offer. Arrows may be treated as objects in the same way as verbs can be employed as nouns or in the way noun–phrases can be constructed with the use of prepositions.

2.1 Initial and Final Objects Representation of Truth

An object in a category $\mathbf{C}$ where there is one and only one arrow from every other object to it is known as the final or terminal object of $\mathbf{C}$. This may be denoted by $\mathbf{1}$ for the whole category, more precisely with the subscript $\mathbf{1_C}$ where $\mathbf{C}$ now represents the whole category $\mathbf{C}$. In the logic context the symbol top $\top$ carries over from elementary theories of logic. The statement that A is true may be represented by $A \longrightarrow \top$.

Dually to the final object there may exist a corresponding initial object where there is one and only one arrow from it to every other object in the category. In the category of sets, for example, the empty set $\emptyset$ performs this role. The initial object in the context of logic is the symbol bottom $\perp$. That A is false is therefore representable as $A \longrightarrow \perp$.

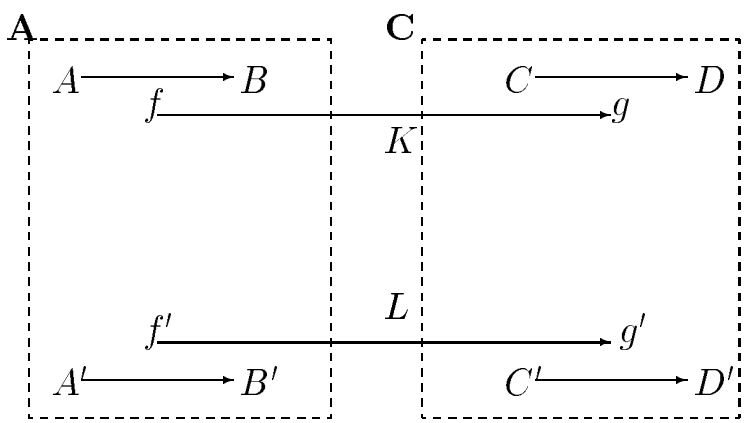

Figure 2.2: Functors compare Categories

2.2 Functors

An arrow between categories is termed a functor. Figure 2.2 shows functor arrows K, L between categories $\mathbf{A}$ and $\mathbf{C}$ containing objects $A, B, C, \ldots$ interrelated by arrows $f, g, \ldots$. In Figure 2.2, K assigns from the source object A the target object K(A) to C and from a source arrow f the target arrow K(f) to g. These are covariant arrows. The direction of K and L may be reversed to give the dual contravariant arrow.

An arrow between any two categories or subcategories will be a functor. So the inclusion of a subcategory within its category is a functorial concept. An important functor for language is the free functor on an

alphabet:

$$F(X) : \mathbf{X} \longrightarrow \mathbf{X}^*$$

The free functor F generates language by its arrangements and organization of words, that is finite strings over the alphabet. A double powerset functor is therefore needed to carry a character x to a string X, $\quad x \mapsto < X >$.

2.3 Typing

The classical limitation to set theory is that it is by nature typeless. From this arises most of the paradoxes with sets [Russell 1900]. Russell himself showed that the paradoxes could be resolved by introducing above sets a higher–level concept of *class*. He and Whitehead developed a theory of logical types [Whitehead & Russell 1910] which has proved unwieldy for everyday mathematical use. Category theory starts afresh at a higher abstract level and has a naturally inherent concept of the type.

Discrete items are identified by the single category $\{*\}$ or $\mathbf{1}$. Therefore elements in a set $a \in A$ is represented categorically by $a : \mathbf{1} \longrightarrow A$. Typing is added by indicating the category (i.e. some pool of values in set theory extensions) from where the item is taken. For example $a : 1_{\mathbf{C}} \longrightarrow A$ or more simply $\mathbf{C} \xrightarrow{a} A$ makes the element a in set A of type $\mathbf{C}$. Furthermore A need not be an object in the category of sets but may belong to a more general category.

In each of these examples, the arrow is relating categories and is strictly a functor. This emphasizes the need for multi–level capabilities for typing. A fundamental type of arrow is the isomorphism which can be rigorously defined in category theory. An arrow $f : A \longrightarrow B$ is an isomorphism if there is an arrow $g : B \longrightarrow A$ such that

$$gf = 1_A, fg = 1_B$$

Isomorphism is a simple example of an idempotent (e) where the composition of an arrow with itself is itself: $e \circ e = e$. The category of idempotents may be split with the effect that an isomorphism is a composition of a section followed by a retraction: $s : A \longrightarrow B$ and $r : B \longrightarrow A$. That is a section can be created out of an arrow; the retraction of this section collapses it back onto the identical arrow. This is important for language because the verb *to be* is idempotent (a thing is itself is itself). A section is a grammatical *complement* of the verb *to be*.

Legal concepts are instantiated in this way. For instance a transaction that can be reciprocated between persons is an isomorphism

$$\begin{tikzcd} \mathbf{1_P} \arrow[r, shift left, "g"] & \mathbf{1_P} \arrow[l, shift left, "f"] \end{tikzcd}$$

where $gf = fg = \mathbf{1_P}$. If this arrow defines delivery, it creates the class (a section) of deliverables in law. In an isomorphism the property can be returned as it was. This can define a class of personal property, namely personalty, represented by the diagram:

$$\begin{tikzcd} & \mathbf{A} \arrow[dl, "\text{section}"'] \arrow[dr, "\text{retraction}"] & \\ \mathbf{1_P} \arrow[rr, shift left, "g"] & & \mathbf{1_P} \arrow[ll, shift left, "f"] \end{tikzcd}$$

Figure 2.3: Personal Property as a *section* of Persons

2.4 Preorders

The concept of direction in the arrow also gives rise to ordering. It is a weak sense of ordering only defined locally in the context of the domain or codomain of the arrow and those composed with it. From the nature of the arrow, a fundamental ordered structure is generated and known as the preorder where any two objects are related by at the most one arrow $A \longrightarrow A'$. For the preorder the conventional symbol for this arrow is $\leq$: $A \longrightarrow A'$. From the axiomatic construct of composition, there is transitivity. $A \longrightarrow A' \longrightarrow A''$ means that $A \longrightarrow A''$. This does not mean that A'' cannot precede A globally. For a preorder where A precedes A' and A' precedes A, A is equivalent to A'. The preorder arrow written $\leq$ therefore has the meaning 'less than or equivalent'.

2.4.1 Examples of preorders

Information systems are preordered, that is there is a potential connection between any two items of information. The fundamental ordering

of parallel processing and distributed computing is the preorder. Hypermedia is preordered [Heather & Rossiter 1995]. Again there is the potential connection between any two items and until there is a specific information query, every item is equivalent to every other.

Because of the importance of preorders in information systems, various models are being advanced to handle the preorder relationship, such as Petri nets and databases. The universal relation is a universal preorder. The underlying organization in a neural net is a preorder. Some operating systems provide facilities for representing preorders. The Unix file system is an example. It is possible to have cycles and files with the same name. They are equivalent for the purpose of name, not equal. The fundamental nature of the law before it is applied to any particular situation is a preorder.

2.4.2 Partial Orders

A stronger form of ordering is the partial order. In this instance the partial order arrow $\leq: A \longrightarrow A'$ has the meaning that A precedes or is the same as A'. This is in effect to add an anti-symmetric condition that if A precedes A' and A' precedes A then A is the same as A'. Examples of partial orders are lattices, trees, acyclic graphs. The finite strings, which compose the free functor $\mathbf{X}^*$, form a partial order represented as a lattice ordered by inclusion.

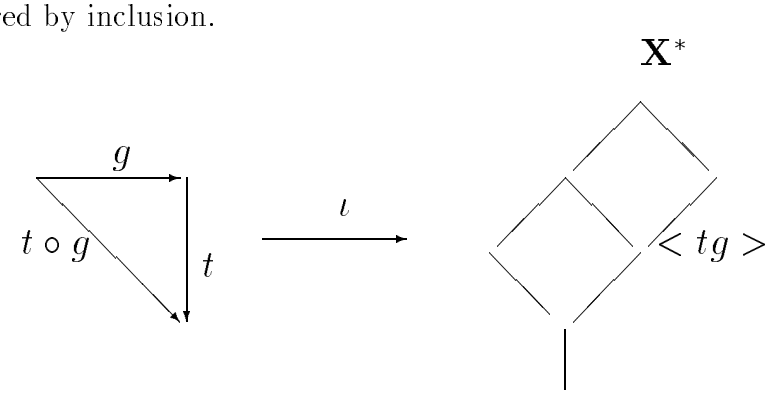

Figure 2.4: Diagram mapped onto a Language String $\mathbf{X}^*$

It should be stressed that as the diagrams are formal, the labels applied to them are formal algebraic representations. This is the nature of the formal language of mathematics. However, it is also the essence of natural language: labelling is equivalent to applying a language. Sequences of characters forming words, sentences, paragraphs, etc, are not arbitrary but are formally constrained. As explained earlier, a language is given by $\mathbf{X}^*$. A statement describing the state or an action is an insertion into the language. Figure 2.4 shows a diagram (geometry) expressed by mapping it onto a string (algebra).

Different languages $\mathbf{X}^*, \mathbf{Y}^*, \mathbf{Z}^*, \ldots$ give different lattices varying the choice of finite string at any point in the respective lattice. When the alphabet is a set, the string representing the diagram is a subset of the possible sequences in the language, that is the double powerset over the alphabet. European languages use alphabets which are sets but even in their written form tend to go beyond linear orderings. Further dimensions are often layered onto the sequence of characters by the use of different fonts, etc. For instance the printed page of an Act of Parliament is not just a linear sequence of characters but a partial ordering with information carried in the format of the printed text [Heather & Rossiter 1987].

$\mathbf{X}^*$ need not be over an alphabet but may be a free functor on a phonetic category for speech or over a category of syllabaries for languages with graphic writing. Set theory is not easily extendible to deal with partial orders of subobjects beyond subsets and this shows the need for the use of category theory or a formalism of equivalent power to cope with natural language. The law applying to a particular case is a partial order and this is the basis of legal hypermedia [Heather & Rossiter 1995].

2.5 Natural Transformations

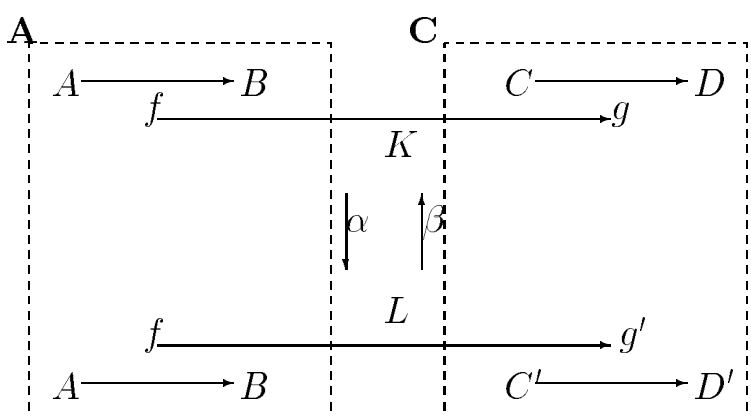

Figure 2.5: Natural Transformations compare Functors

An arrow between functors is termed a natural morphism (or transformation) as shown in Figure 2.5 where there is a natural transformation α from K to L, written:

$$\alpha : K \longrightarrow L$$

This natural transformation assigns to each source object A a target arrow

$$\alpha_A : K(A) \longrightarrow L(A)$$

There are tight inter-relationships between the levels in category theory: morphisms and objects of categories at the lowest level are part of the expressions at the highest level of natural morphisms. A special case of natural transformation is the concept of natural isomorphism where, in the example given, the composites $\alpha \circ \beta$ and $\beta \circ \alpha$ are the identity natural transformations of L and K respectively. This links to another mathematical approach where α is regarded as an isomorphism of a model of categories giving connections to model theory.

Natural transformations operate at the level of the message. This will include philosophy, policy, discretion and meaning. For language, which is derived from the free functor as described above, there is a natural transformation between the characters and what the sequence of characters means:

$$\eta : X \longrightarrow F(X)$$

The Greek alphabet is conventionally used to represent natural transformations. In Figure 2.4, it should be noted that the arrow mapping the insertion of the left–hand diagram onto the partial order is labelled with the greek character iota ι (not i) because it is a natural transformation.

For applications the essential effect of natural transformations is that they relate one order to another and control reordering. The effect is that any form of interpretation, cognition or of perception by the senses in general requires a natural transformation.

European languages that use Latin characters form subcategories of $\mathbf{X}^*$. For two such languages $\mathbf{U}^*, \mathbf{V}^*$ we have $\iota : \mathbf{U}^* \longrightarrow \mathbf{X}^*$ and $\iota : \mathbf{V}^* \longrightarrow \mathbf{X}^*$. This insertion is monic but the two languages do not partition $\mathbf{X}^*$ because (even taking all languages together), there are some strings which do not exist in any language and some strings that occur in more than one language (often with different meanings). Translation is a natural transformation between strings in the respective languages $\alpha_u :< u > \mapsto < v >$. α is a natural transformation indicating that it operates at the pragmatic level. In the language of category theory α is a generalized string.

2.6 Products and Pullbacks

Two operations common in relational algebra, product and projection, are represented directly in category theory within the concept of the *limit* and diagrammatically represented through the construction of a cone. A cone consists of an open triangle comprising three objects, for example, P, A and $P \times A$ where the product $P \times A$ is the vertex of the cone as shown in Figure 2.6 below. The projection (natural transformation) arrow π operates in either a left (π_l) or a right (π_r) context, depending on which part of the product is being selected.

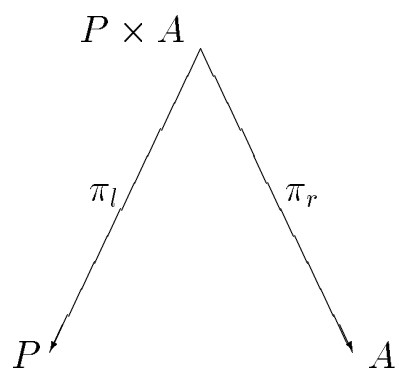

Figure 2.6: Product Cone for Objects P and A

In strict category terms, the cone as presented above does not appear to commute but it may alternatively be presented as in Figure 2.7 where for any object V and arrows $q_1 : V \longrightarrow P$ and $q_2 : V \longrightarrow A$, there is a product U with projections P and A such that the diagram commutes, that is the two equations hold:

$$\pi_l \circ q = q_1$$

$$\pi_r \circ q = q_2$$

U is the universal product of $P \times A$. It is the limit (*meet*) of any conjunction of P and A from multiplication in arithmetic, AND in logic to real world phenomena like chemical compounds and in legal language [Heather & Rossiter 1994b] like joint liability and ownership as joint tenants and marriage. It is an abstraction of the concept of combined

togetherness. The dual concept of separate togetherness is the coproduct which involves the colimit (*join*) corresponding to the arithmetical sum, the logical OR (the exclusive XOR), a chemical mixture or in legal language several liability, tenancy in common ownership, divorce, etc. Projected onto sets these would be examples of disjoint unions. The features of real–world colimits need to be represented in more general categories.

$$\begin{tikzcd}
 & V \arrow[dl, "q_1"'] \arrow[d, "q"] \arrow[dr, "q_2"] & \\
P & U \arrow[l, "\pi_l"] \arrow[r, "\pi_r"'] & A
\end{tikzcd}$$

Figure 2.7: Commuting Product Cone for Objects P and A

An important product in practice is the pullback or fibred product where a product is restricted over some object or category. If P and A both have arrows to some common C as $P \xrightarrow{f} C$ and $A \xrightarrow{g} C$, then the subproduct of P and A over C written as $P \times_C A$ may be represented by the diagram shown in Figure 2.8.

$$\begin{tikzcd}
 & P \arrow[dr, "g"] & \\
P \times_C A \arrow[ur, "g^*(t)"] \arrow[dr, "\epsilon_g(t)"'] & & C \\
 & A \arrow[ur, "t"'] &
\end{tikzcd}$$

Figure 2.8: Diagram of Pullback of P and A over C

where $g(P) \sim t(A)$ and $g(P), t(A)$ are both objects of C. This diagram commutes in that

$$g \circ g^*(t) = t \circ \epsilon_g(t)$$

$g^*(t)$ can be described as the pullback of t along g. In terms of Figure 2.7, for any pair of arrows $q_1 : V \longrightarrow P$ and $q_2 : V \longrightarrow A$ with $g \circ q_1 = t \circ q_2$, there is a unique morphism $q : V \longrightarrow P \times_C A$ satisfying $g^*(t) \circ q = q_1$ and $\epsilon_g(t) \circ q = q_2$.

If $C = \{*\}, P \times_C A$ is the whole product $P \times A$ shown in Figure 2.9.

$$\begin{tikzcd}
 & P \arrow[dr] & \\
P \times A \arrow[ur] \arrow[dr] & & \{*\} \\
 & A \arrow[ur] &
\end{tikzcd}$$

Figure 2.9: Pullback of P and A over $\{*\}$

The pushout is the dual of the pullback and corresponding diagrams deal with coproducts.

2.7 Adjointness

Adjointness between two categories

$$F \dashv G : \mathbf{Y} \longrightarrow \mathbf{Z}$$

has left and right components which specify how an arrow in category $\mathbf{Y}$ is related to an arrow in category $\mathbf{Z}$. This is the fundamental concept of implication to be found in geometric logic. The left component is the free functor $F : \mathbf{Y} \longrightarrow \mathbf{Z}$ and the right component the underlying functor $G : \mathbf{Z} \longrightarrow \mathbf{Y}$. F is left adjoint and G is right adjoint to F. This is a natural bijection between arrows which holds subject to the condition for all objects Y belonging to $\mathbf{Y}$ and all Z belonging to $\mathbf{Z}$ such that:

$$F(Y) \longrightarrow Z \text{ implies and is implied by } Y \longrightarrow G(Z)$$

F is a generalization of intransitive verbs and G of transitive ones.

With this condition there are two natural transformations or unit of adjunction:

$$\eta : 1_{\mathbf{Y}} \longrightarrow GF,\ \epsilon : FG \longrightarrow 1_{\mathbf{Z}}$$

Adjointness is therefore an important concept whose universal properties have only really been appreciated since the advent of category theory [Freyd & Scedrov 1990]. The importance of equivalent classes has been long recognized but adjointness provides a formal specification for equivalent structures which include dynamic systems.

This discovery of adjoints by Kan [1958] is of far–reaching importance in that it describes the behaviour at the centre of all systems. It is fundamental to information systems. The law itself is always right adjoint to the free functor that describes the society in which it operates [Heather & Rossiter 1994a]. By the adjoint functor theorem [Freyd & Scedrov 1990], left adjoints preserve colimits and right adjoints preserve limits.

Translation is an example of adjointness. A text in the language category **U** is mapped onto the language category **V**. This is a free functor mapping from a string in the partial order $\mathbf{U}^*$ onto a particular string in the partial order $\mathbf{V}^*$. The translator has a freedom in the selection of the string (for example, in style as determined by some natural transformation). The underlying functor α^* relates the meaning of the string $< v >$ to the string $< u >$. Figure 2.10 shows the relationship.

$$\begin{tikzcd} U^* \arrow[r, shift left, "\alpha"] & V^* \arrow[l, shift left, "\alpha^*"] \end{tikzcd}$$

Figure 2.10: Adjointness between U^* and V^*

The counit of adjunction $\epsilon : \alpha\alpha^* \longrightarrow \mathbf{1_V}$ is a measure of the completeness of the translation. If the translation is ideal, ϵ equals 1 i.e. $\alpha\alpha^* = \mathbf{1_V}$. In practice it is likely that $\alpha\alpha^*$ is less than $\mathbf{1_V}$. In effect this is performing a translation back from the category **V** to the category **U**, retranslating the result, and then comparing the result with the first translation. That is for a given translated string $< v >$

$$\epsilon_{<v>} = \alpha(\alpha^* < v >) - < v >$$

this is an example of overloading in an object–oriented context of the minus sign to indicate the difference in translation. The pure categorial representation would be

$$\epsilon_{<v>} : \alpha(\alpha^* < v >) \longrightarrow < v >$$

It may be noted that this is a comparison from the point of view of the target language of the translation. The alternative measure of comparing translations from the point of view of the initial language can be given in terms of the unit of adjunction η

$$\eta_{<u>} = \alpha^*(\alpha < u >) - < u >$$

If a retranslation produces an original the translations are true and both are isomorphic functors.

Chapter 3

Application

We are now in a position to explore this integrated formalism of category theory for law, language and logic as it would apply to the sentence quoted in the introduction:

John gives Mary the ring and title passes on delivery

to see how category theory deals with language, logic and law.

3.1 Language

A starting point in the language might be the statement: *John gives the ring to Mary.* This is represented by the formal diagram in Figure 3.1:.

$$\begin{tikzcd} J \arrow[r, "g"] \arrow[dr, "t \circ g"'] & R \arrow[d, "t"] \\ & M \end{tikzcd}$$

Figure 3.1: Formal Diagram for *John gives the Ring to Mary*

The composition arrow (compare Figure 2.4) would be labelled in conventional mathematics as $t \circ g$ but the full version, remembering that the objects are identity arrows, would be $MtRgJ$ as shown in Figure 3.2.

$$\begin{tikzcd} J \arrow[r, "g"] \arrow[dr, "MtRgJ"'] & R \arrow[d, "t"] \\ & M \end{tikzcd}$$

Figure 3.2: *John gives the Ring to Mary*
with the full label for the composite arrow

This composite label is only one of a number of possible strings and other strings are possible to represent the diagram as a whole. These are analagous to the alternative ways in which the basic statement can be represented in language. The English language is fairly flexible and can deal with a number of (but not all) the possible representable strings for the diagram. It is possible to start on any arrow and go in either direction. Possibilities are:

composition arrows	literal meaning	potential English
$MtRgJ$	*Mary to, the ring, gives, John*	the mathematical convention, improbable in English, but might just be acceptable in poetic form
$JgRtM$	*John gives the ring to Mary*	natural English order but opposite to the mathematical convention
$RtMJg$	*the ring to Mary (by) John is given*	the passive voice with a contravariant composition arrow
$MJgRt$	*Mary by John is given the ring to*	another passive with contravariant g and contravariant composition
$RgJMt$	*The ring is given by John, Mary to*	passive with contravariant g but covariant composition
$JgMR$	*John gives Mary the ring*	an alternative representation of the composite arrow but reordered

The reordering in the last example with the admission of the t indicates a natural transformation at work, which is a matter of style and at the level of the senses, allowing this reordering in English. Because it is a natural transformation, it can be bound up with the meaning and indeed it is the semantics that determines there is no ambiguity in this order.

Despite the flexibility of English, some of these possible strings are

not well–founded forms but may sometimes be used by small children or non–native speakers of English. However, these alternatives might be acceptable in other languages and might be quite appropriate forms in an inflected language like Latin or German. A Latin version could be

Mariae aulum dat Johannes

where the inflected dative ending *ae* picks up the arrow t; also the accusative ending of *aulum* indicates the codomain and the nominative *Johannes* the domain of the arrow g. With an inflected language the order of words is not critical and combinations might make good Latin which would not be acceptable in English. This demonstrates the categorial significance of the endings and at the same time indicates there is no difference in principle in the mapping onto the strings of an inflected language. It is for this reason that there no particular problems in translating between inflected and non–inflected languages.

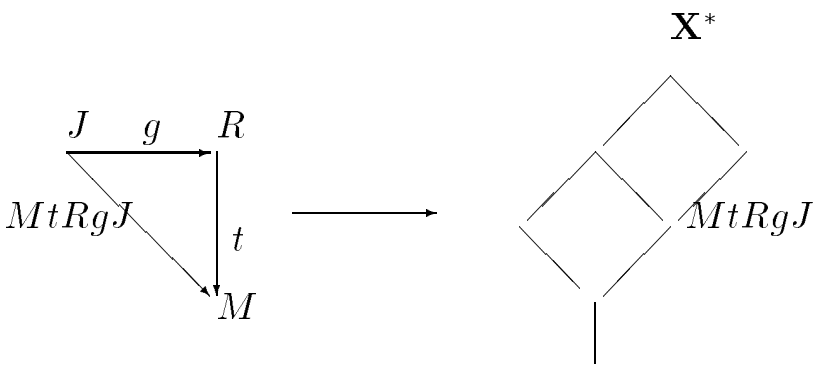

Figure 3.3: The Expression $MtRgJ$ as a string in the free word functor $\mathbf{X}^*$

A more general form of *John gives the Ring to Mary* could be given with the identity functor $\mathbf{1_P}$ on the category of persons $\mathbf{P}$ replacing John and Mary and the identity functor $\mathbf{1_A}$ on the category of articles replacing the ring, as shown in Figure 3.4:

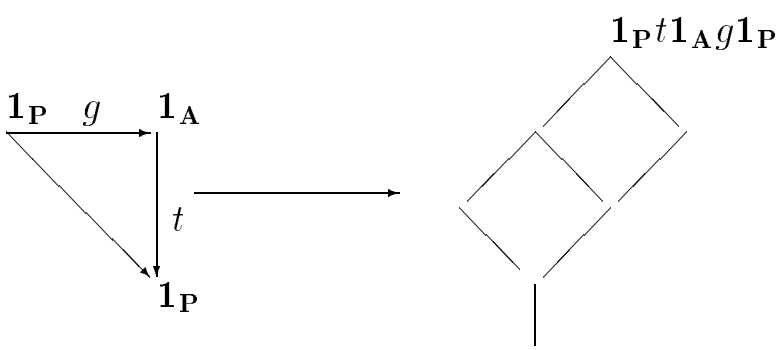

Figure 3.4: Diagram for a Person gives an article to a Person

This diagram should be compared with diagram 2.3 to see that the abstract concept of *giving* is related to the legal concept of *personalty*.

A person could be a corporate body like a company and this would probably affect the meaning of the arrow g so it has the meaning of a general presentation. Alternatively the functor $\mathbf{1_P}$ could be a pullback of more than one person in a joint presentation.

3.2 Logic

We have seen that limits and colimits are generalizations of logical operators AND, OR and the intersection and unions of set theory. These are wrapped up in the form of language expressions we have just discussed. These lead to logical inferences. Any composition is a logical inference in geometric logic. It is this equivalence for composition in language that we have been considering that makes language logical [Heather & Rossiter 1994b]:

$$\frac{g : J \longrightarrow R, t : R \longrightarrow M}{tRg : J \longrightarrow M}$$

This is the general (higher–order) predicate logic expression which subsumes the various possible forms of language discussed above.

The possible initial legal and physical states for the example of *John giving the ring* are:

$$\begin{tikzcd}
 & J \times R \arrow[dl] \arrow[dr] & & R \times M \arrow[dl] \arrow[dr] & \\
J \arrow[dr] & & R \arrow[dl] \arrow[dr] & & M \arrow[dl] \\
 & J + R & & R + M &
\end{tikzcd}$$

Figure 3.5: Preorder of possible initial states

This preorder of possible states combines both pullbacks and pushouts. Each pullback limit $\mathbf{P} \times \mathbf{A}$ or $\mathbf{A} \times \mathbf{P}$ indicates potential ownership (*dominium*) and the pushout colimit $\mathbf{P} + \mathbf{A}$ or $\mathbf{A} + \mathbf{P}$ provides the possibility that a person has the article (*possessio*). The natural state is that both ownership and possession continue in time if no action affects

them. This is the arrow of time which is right adjoint to possessing and left adjoint to owning, that is the adjointness:

$$\text{possess} \dashv \text{time} \dashv \text{own}$$

holds in the following diagram:

$$\begin{tikzcd}
 & J \times R \arrow[dl] \arrow[dr] & & R \times M \arrow[dl] \arrow[dr] & \\
J \arrow[dr] & & R \arrow[dl] \arrow[dr] & & M \arrow[dl] \\
 & J + R & & R + M &
\end{tikzcd}
\quad \longrightarrow \quad
\begin{tikzcd}
 & J \times R \arrow[dl] \arrow[dr] & & R \times M \arrow[dl] \arrow[dr] & \\
J \arrow[dr] & & R \arrow[dl] \arrow[dr] & & M \arrow[dl] \\
 & J + R & & R + M &
\end{tikzcd}$$

Figure 3.6: Preorder of the time arrow between various states

This is a consequence of the adjoint functor theorem referred to in the discussion of adjoints above, namely right adjoints preserve limits and left adjoints preserve colimits. Time is both left and right adjoint to the laws of physics. The continuation of possession is an effect of the laws of physics particularly Newton's first law that things continue where they are put if nothing is done to them. However, time also preserves ownership. *To own* is right adjoint to *time*. These are examples where ownership (a limit) is preserved by right adjoints and ownership (a colimit) by left adjoints.

The action of giving is a composition of the time arrow with a change in the physical and/or legal states of the article.

These preorders describe states that are possible in the real world. The existence of left and right adjoints determines respectively whether possession and/or ownership passes. The preorder provides possible situations but not all are mutually possible. The quotient equivalent partial orders give the possibilities that can exist together. The legal action is a functor onto one of the following logical states depending on the legal position in the Figure 3.6.

That diagram is a preset giving the possible orderings available represented by the quotient posets. Various poset possibilities are given in the following figures:

$$\begin{tikzcd}
 & J \times R \arrow[dl] \arrow[dr] & & & & R \times M \arrow[dl] \arrow[dr] & \\
J \arrow[dr] & & R \arrow[dl] & M \arrow[r] & J & R \arrow[dr] & & M \arrow[dl] \\
 & R + M & & & & & R + M &
\end{tikzcd}$$

Figure 3.7: *John gives Mary the ring she already possesses*
Mary already has possession, John passes title.
Left adjoint but no right adjoint.

$$\begin{tikzcd}
 & J \times R \arrow[dl] \arrow[dr] & & & & J \times R \arrow[dl] \arrow[dr] & \\
J \arrow[dr] & & R \arrow[dl] & M \arrow[r] & J & R \arrow[dr] & & M \arrow[dl] \\
 & J + R & & & & & R + M &
\end{tikzcd}$$

Figure 3.8: *John lends Mary the ring*
John has both title and possession and parts with possession but not title.
Right adjoint but no left adjoint.

$$
\begin{tikzcd}
 & J \times R \arrow[dl] \arrow[dr] & & & & R \times M \arrow[dl] \arrow[dr] & \\
J \arrow[dr] & & R \arrow[dl] & M \arrow[r] & J & R \arrow[dr] & & M \arrow[dl] \\
 & J + R & & & & & R + M &
\end{tikzcd}
$$

Figure 3.9: *John gives Mary the ring*
John having both title and possession passes both to Mary.
No right or left adjoint.

$$
\begin{tikzcd}
 & & R \times M \arrow[dl] \arrow[dr] & & & & R \times M \arrow[dl] \arrow[dr] & \\
J \arrow[dr] & & R \arrow[dl] & M \arrow[r] & J & R \arrow[dr] & & M \arrow[dl] \\
 & J + R & & & & & R + M &
\end{tikzcd}
$$

Figure 3.10: *John gives Mary her ring (back)*
John has possession. Mary has title and John returns possession.
Right adjoint, no left adjoint.

3.3 Law

The pullback diagram in Figure 2.8, reproduced here as Figure 3.11, can be applied to the situation for ownership. Category **C** is any context and here the context is the action of giving. The arrows g, t are insertions in that context and represent respectively the same arrows g and t as in all the figures above. The diagram is a pullback of g along t. $g^*(t)$ is then the projection of ownership onto the category of persons. g^* picks out the owner P_g of the particular article A_g that is to be the domain of the preposition arrow *to*.

$$\begin{tikzcd}
 & P \arrow[dr, "g"] & \\
P \times_C A \arrow[ur, "g^*(t)"] \arrow[dr, "\epsilon_g(t)"'] & & C \\
 & A \arrow[ur, "t"'] &
\end{tikzcd}$$

Figure 3.11: Legal ownership as a pullback of *giving* along *to*.

The natural transformation $\epsilon_{g(t)}$ is the projection of ownership onto the article. This epsilon $\epsilon_{g(t)}$, as a natural transformation at the message level, provides the legal position such as the statement that *title passes on delivery*. A corresponding diagram can be given for the pushout (the dual of the pullback) to deal with the law of co-ownership.

It is interesting to note that both g^* and ϵ represent legal norms. The contravariant functor g^* is a prescriptive expression while the natural transformation ϵ represents the law descriptively.

Possession is left adjoint and ownership right adjoint to the rule of law. This means that, by the adjoint functor theorem, right adjoints preserve limits. In this context this means that ownership goes with the article. The whole diagram gives the legal effect. This shows the integration of law, language and logic in constructive theory needed for legal computer science.

For diagram 3.11 taken with any of the figures 3.7 to 3.10 formally represents in geometric logic the statement:

John gives Mary the ring and title passes on delivery

Chapter 4

References

Barr & Wells 1990, M.Barr & C.Wells, *Category Theory for Computing Science*, Prentice–Hall.

Bell 1988, J.L.Bell, *Toposes and Local Set Theories*, Oxford University Press.

Biagioli et al 1995, Biagioli, C, Mercatali, P, Sartor, G, Taddei, G, & Tiscornia, D, L, Informatica Giuridica e la Filosofia del Diritto in Italia, 17th IVR World Congress Bologna, June 1995, **V** 330–361.

Boileau & Joyal 1981 A.Boileau & A.Joyal, La Logique des topos, *J Symbolic Logic* **46** 6–16.

Freyd & Scedrov 1990, P.J.Freyd & A.Scedrov, Categories, Allegories, *North–Holland Mathematical Library* **39**.

Heather & Rossiter 1987, M.A.Heather & B.N.Rossiter, *Database techniques for text modelling: the document architecture of British statutes,* Computing Laboratory Technical Report Series no. 227, University of Newcastle upon Tyne. (66pp)

Heather & Rossiter 1994a, M.A.Heather & B.N.Rossiter, *Models for Representing Legal Norms with the Pullback Functor* f^*, Computing Science Technical Report no.470, University of Newcastle upon Tyne, (20pp).

Heather & Rossiter 1994b, M.A.Heather & B.N.Rossiter, Applying Geometric Logic to Law, in: *4th National Conference on Law, Computers and Artificial Intelligence*, Exeter 1994, 80–95.

Heather & Rossiter 1995, M.A.Heather & B.N.Rossiter, Context Sensitive Awareness in Legal Hypertext, *Informatica e Diritto*, Special Issue on Hypertext and Hypermedia in the Law, Second Series, **IV**(1) 31–53.

Kan 1958, D.M.Kan, Adjoint functors, *Trans AMS*, **87** 294–329.

Karpf 1989, Karpf, J, The Category of Law Models, VI Nordic Confer-

ence of Legal Informatics, Oslo, Sept 1989.

Karpf 1991, Karpf, J, Categorical Modelling in Law, III Scandinavian Conference on Artificial Intelligence, SCAI90, Roskilde University, Denmark, May 1991.

Karpf 1995, J.Karpf, Italian Law as a Mathematical Category and Categorical Development of Computer Systems in Law, in: 17th IVR World Congress, CIRFID, Bologna, CLUEB **V** 257–265.

Lambek & Scott 1986, J.Lambek & P.J.Scott, Introduction to Higher Order Categorical Logic, *Cambridge Studies in Advanced Mathematics* **VII**, Cambridge University Press.

Lupoi 1970, Lupoi, M, Giuscibernetica, informatica giuridica. Problema per il guirista, Foro italiano, Roma 1970.

Manes & Arbib 1986, E.Manes & M.Arbib, *Algebraic Approaches to Program Semantics*, Springer Verlag.

Pierce 1991, B.C.Pierce, *Basic Category Theory for Computer Scientists*, MIT Press.

Pitt et al 1986, *Lecture Notes in Computer Science*, **242**, Springer Verlag, Berlin.

Rhydeheard & Burstall 1988, D.E.Rhydeheard & R.M.Burstall, *Computational Category Theory* Prentice–Hall.

Russell 1900, B.Russell, *Principles of Mathematics*, **I**, Cambridge.

Whitehead & Russell 1910, A.N.Whitehead & B.Russell, *Principia Mathematica*, Introduction Chapter 2, 1st edition, Cambridge (1910-13).