J.M. Dunn[1]

TERNARY RELATIONAL SEMANTICS AND BEYOND: Programs as arguments (data) and programs as functions (programs)

1. Introduction

The purpose of this paper is to review some ideas from Dunn and Meyer (1997) and Dunn (2001) and "write them large". The first paper showed how to represent combinatory algebras using ternary frames, and the second paper showed how to do the same for relation algebras. It should be pointed out that these representations both fall under the general heading of "gaggle theory", as developed in a series of papers beginning with Dunn (1991), in which an *n*-ary operator is represented using an *n+1*-placed relation. These results were presented in a somewhat mathematical fashion, although philosophical motivations were introduced as well. The present paper will focus on these philosophical motivations and clarify and extend them. In addition it will point to some future research directions in connection with Pratt's dynamic logic and Hoare's logic of programming.

2. Frege and Boole

Frege's famous dictum, expressed as a conundrum, is that "the concept horse is not a horse". This is our text for this occasion.

Frege has three theses that are relevant. These theses are to be found scattered about his three papers "Concept and Function", "Concept and Object", and "Sense and Reference"[2].

1. A concept is a special kind of function, taking objects as arguments and operating upon them to produce a rather unfamiliar object (a truthvalue)[3]. Thesis 1.

[1] School of Informatics, Indiana University, Bloomington IN 47405 (USA), dunn-@indiana.edu.

[2] Originally published in German in 1892-93 and translated into English in Geach and Black (1960).

[3] Frege also had a distinction between two kinds of functions, which we might call, using terminology that has become current, intensional and extensional (Frege talked rather of functions and their course of values (identifying the latter with sets)). This distinction is not important for this paper and we shall ignore it.

2. A function is a very different thing than its argument, and in fact a function cannot itself ever be an argument (functions are "incomplete" whereas their arguments are "complete"). Thesis 2.

3. A definite description ('the so-and-so') refers to an object. Thesis 3.

From 3 we conclude:

4. The definite description `the concept horse' refers to an object.

From1and 2 we conclude

5. A concept cannot be an object.

And so we get something very close to Frege's dictum (but not yet a conundrum):

6. The reference of the definite description `the concept horse' is not a concept.

But the following seemingly obvious principle gives the conundrum:

7. The reference of a definite description `the so-and-so' is the so-and-so.

For now we can conclude:

8. The reference of a definite description `the concept horse' is the concept horse.

And using substitution of identicals, from 6 and 8, we finally obtain the conundrum:

9. The concept horse is not a concept.

Thus, the concept horse, given the argument Man of War, gives the value True, and given the argument Francis (the talking mule) it gives the value False. Frege's point is that there is a very distinct difference between referring to a function, and using it. Frege liked to talk of concepts/functions as complete (saturated) items, and of objects/arguments as incomplete (unsaturated) items.

Perhaps a more contemporary way of making the distinction is to say that when we refer to the concept horse by using the phrase "the concept horse" we are referring to a static object, and not to the function in its full dynamic nature.

Modern logicians can make this distinction using the lambda notation of Church (1941). Thus $x + 1$ is incomplete, waiting for an argument to be input to the variable x, say 2, before producing the output value $2 + 1 = 3$. Whereas $\lambda x(x + 1)$ is complete and denotes the function that applied to an argument x returns the result $x +1$: It can have functions applied to it as an argument. For example, we can apply iteration to it so as to obtain $\lambda x(x + 2)$. Frege goes on at length to make this by now familiar distinction, using Greek lower-case letters (he did not have the lambda notation, which importantly denotes the scope of the binding), writing things such as $\xi + 1$ to denote the successor function. Can the same item be both an object/argument and a concept (function)? Frege thought not, but thought that when we want to refer to a concept, we end up referring to a close cousin of it which he termed a "concept correlate". In this paper we show how to make sense of this idea.

In further motivating this paper, we turn to another one of the founders of modern logic, George Boole. Boole (1847) had two interpretations of what has now come to be known as Boolean algebra. The primary interpretation was that the elements of a Boolean algebra are to be thought of as propositions. The secondary interpretation was that the elements of a Boolean algebra are to be thought of as sets. Boole observed that propositions and sets obeyed the same laws, mapping conjunction into intersection, disjunction into union, and negation into relative complement. He explained this by observing that a proposition can be regarded as the set of cases in which it is true. This is the fundamental origin of the idea that a proposition can be understood as a set of possible worlds. This latter notion has been generalized to the idea that a proposition can be understood as a set of information states. These can be partial, and even inconsistent. Total, consistent information states are surrogates for possible worlds. Let us call this Insight 1. It is the basis of the Carnap-Kripke-Montague-Stalnaker-et al program of semantics. This might be called a static conception of a proposition, since propositions just "sit there" as sets of states.

More recent authors, e.g., Gärdenfors and Makinson (1988) have observed that there is also a more dynamic conception of proposition. When we add a proposition to our existing set of beliefs, it transforms these beliefs into another belief set. Let us call this Insight 2. A belief set can be conceptualized to be one gigantic proposition (the conjunction of someone's beliefs). So a proposition can be viewed as an operator on propositions. This is the dynamic conception of proposition.

Insight 1 and Insight 2 both seem to be correct. How can we reconcile them? The quick answer is that a proposition must be simultaneously an object (potential argument) and a function. But what does this mean? This is the topic which we shall explore throughout this paper.

3. Von Neumann's concept of a "stored program"

There is a certain irony in the development of the study of logic and computation which has been a hallmark of the twentieth century. Although this development has taken place in at least the first half of the century in largely the same venues and by largely the same pioneers, largely the study of logic as a foundation for mathematics has been driven by a desire to respect types, whereas formal systems developed for computation are by their very nature type defying. The first is clear from the fixation of most of mathematical logic on first-order logic, despite this a small flirtation with second-order logic, and the standard set-theories based on notions of hierarchies (Zermelo-Frankel, von Neumann-Bernays-Gödel, Morse, Kelly). And the last is clear from the fundamental notion of a "stored program" in the "von Neumann architecture". John von Neumann, in 1946 wrote a paper with Arthur W. Burks and Hermann H. Goldstine titled "Preliminary Discussion of the Logical Design of an Electronic Computing Instrument" (Burks, et al., 1963). This paper begins by discussing the four main components of a computer: the arithmetic logic unit, memory, control, and input-output human interface. In other words, the arithmetic logic unit, the control unit, and input-output. We quote from Riley (1997):

> To von Neumann, the key to building a general purpose device was in its ability to store not only its data and the intermediate results of computation, but also to store the instructions, or orders, that brought about the computation. In a special purpose machine the computational procedure could be part of the hardware. In a general purpose one the instructions must be as changeable as the numbers they acted upon. Therefore, why not encode the instructions into numeric form and store instructions and data in the same memory? This frequently is viewed as the principal contribution provided by von Neumann's insight into the nature of what a computer should be.
>
> He then defined the control organ as that which would automatically execute the coded instructions stored in memory. Interestingly he says that the orders and data can reside in the same memory "if the machine can in some fashion distinguish a number from an order" [Burks, et al., p. 35].
>
> And yet, there is no distinction between the two in memory. The control counter (what we now usually call the program counter) contains the address of the next instruction, and that word is fetched to be executed. Whatever the control unit "believes" to be an order or to be data is treated as such. One ramification of this is that the instructions can operate upon other instructions, producing a self-modifying program. This has not been considered good form for many years, because of the implications for program debugging and the desire for reentrant code in some situations. It

is possible that new developments in artificial intelligence may bring fresh attention to the possibilities afforded by this characteristic [Bishop 1986].

While von Neumann and his co-workers deserve the credit for specifying the broad outlines of the contemporary electronic computer, there is an important presaging in the work of Turing in his proof that there could be a universal computing machine, where he used the idea of coding up the machine tables of other Turing machines in terms of numbers that could then be used as input to the universal machine.

The feature that we wish to draw attention to here is the fact that in a general purpose computer, a program can be either function or argument, depending on context. There is a similar feature in the λ-calculus of Church (1941), or the combinatory logic of Curry and Feys (1972), where an expression such as **KI**, or $(\lambda x \lambda yx)(\lambda xx)$ treats the first term as a function and the second term as an argument. Indeed one can have a term of the form *MM*, say **II** or 4.$(\lambda xx)(\lambda xx)$, which treats the very same term *M* as simultaneously standing for both a function and an argument. So much for types!

The contrast during the first fifty year or so of the twentieth century between the type conscious mainstream in mathematical foundations, and the type insensitive undercurrent in the logic of computation is quite striking. It reminds one of the (likely apocryphal) stories of early aviation theorists proving the impossibility of powered flight while the Wright brothers and others were working first on models, and then on real airplanes.

4. The Routley-Meyer Semantics for Relevance Logic

We make what might seem a side trip, to discuss the so-called "Routley-Meyer semantics" for relevance logic[4], but we shall see in the next section that it is closely related to our main journey. It is well-known that Routley and Meyer (1972, 1973) provide a "Kripke-style" semantics for relevance logic using a ternary accessibility relation R[5]. Ignoring negation, a Routley-Meyer frame is a structure $(U, R, 0)$ with $R \subseteq U^3$ and $0 \in U$. A Routley-Meyer model adds an atomic "forcing relation" $\Vdash_0$ between states and atomic sentences $(\alpha.\Vdash_0 p)$. This can be

[4] Anderson, Belnap, and Dunn (1992) contains a good discussion of this semantics, including variants due to Urquhart and Fine discussed below.

[5] There is an alternative "operational semantics" that we could be using. The ternary relation $R\alpha\beta\gamma$ is replaced using a binary operation on states $\alpha + \beta = \gamma$ (Urquhart) or else $\alpha + \beta$ š γ (Fine). The first has also been used by Girard (1987) in the "phase space" semantics for linear logic. The last is more general, extending as it does to allow an underlying distributive lattice. Cf. Anderson, Belnap, and Dunn (1992) for details. Cf. also Dofišen (1992).

extended inductively to provide a "forcing relation" between states and compound sentences. Thus for relevant implication they have the following (χ, α, β range over U)[6]:

$$\chi \models \varphi \rightarrow \psi \quad \textit{iff}\ (\forall\alpha, \beta : R\chi\alpha\beta\ \&\ \alpha \Vdash \varphi\ \textit{imply}\ \beta \Vdash \psi). \qquad (1)$$

They also gave a truth condition for "fusion" (intensional conjunction):

$$\chi \models \varphi \circ \psi \quad \textit{iff}\ \exists\alpha, \beta(R\alpha\beta\chi\ \&\ \alpha \Vdash \varphi\ \&\ \beta \Vdash \psi). \qquad (2)$$

Routley and Meyer thought of *U* as "set ups". A "set up" is like a Carnapian state description except it can be inconsistent and/or incomplete. Routley and Meyer also thought of *U* as something like a set of possible worlds, but with the thought that these can be partial/and or incomplete. We shall continue to use the more neutral and trendy word "state", anticipating applications to computers.

Given the Carnap-Montague-Kripke idea of a "proposition" as a set of possible worlds, a sentence φ can then be taken to be interpreted as the "proposition" $\| \varphi \| = \{\alpha : \alpha \Vdash \varphi\}$.

Using propositions, we can rephrase the truth conditions above as:

$$\| \varphi \rightarrow \psi \| = \| \varphi \| \Rightarrow \| \psi \| =_{def} \{\chi : (\forall\alpha, \beta : R\chi\alpha\beta\ \&\ \alpha \in \| \varphi \|\ \textit{imply}\ \beta \in \| \psi \|)\} \qquad (3)$$

$$\| \varphi \circ \psi \| = \| \varphi \| / \| \psi \| =_{def} \{\chi : \exists\alpha, \beta(R\alpha\beta\chi\ \&\ \alpha \in \| \varphi \|\ \&\ \beta \in \| \psi \|)\} \qquad (4)$$

1. There are several things about this semantics that must be mentioned:
2. Implicit in this semantics is a relation α š β (Routley and Meyer would write "a<b") which can be defined as $R0\alpha\beta$: The "Hereditary Condition" (adopted from intuitionistic logic) imposes the requirement that if $\alpha \in \| \varphi \|$ and α š β, then $\beta \in \| \varphi \|$. This in effect says that a proposition is not just any set of states, but rather one that is closed upwards under š. It is natural to regard this as "the information order".
3. Validity is defined not by reference to arbitrary states (as in the Kripke semantics for modal logic and intuitionistic logic), but rather by reference to the "zero states". Also it is a bit of an

[6] In the definition below Routley and Meyer make the first position the "pivot", but there is no reason why the second point cannot be the pivot. Then we get $\chi \models \psi \leftarrow \varphi$ iff $(\forall\alpha, \beta : R\alpha\chi\beta\ \&\ \alpha \Vdash \varphi\ \textit{imply}\ \beta \Vdash \psi)$. In previous papers I have often reversed these arrows for reasons of wanting to match a convention of residuation theory due to Pratt.

accident that Routley and Meyer pick out one such. In general terms it does not hurt to add a set of them, $Z \subseteq U$.

4. The above observations motivate our defining a (ternary) frame as a structure $F = (U, R, š, Z)$, with š a partial-order on U subject to the condition that $\exists\zeta{\in}Z(R\zeta\alpha\beta)$ iff α š β iff $\exists\zeta{\in}Z(R\alpha\zeta\beta)$ (Z-condition).
5. Validity in a frame ($|=_F \varphi$) is then defined as: $\forall\alpha{\in}U, \forall\zeta{\in}Z : \zeta \,||\!- \varphi$. And validity in general ($|= \varphi$) is defined as validity in all frames ($\forall F, |=_F \varphi$).
6. Every frame gives rise to an algebra of propositions $A(F) = (\wp^{\uparrow}(F), \wedge, \vee, \Rightarrow, /, Z)$. The carrier set $\wp^{\uparrow}(F)$ is just the set of propositions $A \subseteq U$, i.e., subsets of U closed upwards under š. $\wedge$ and $\vee$ are respectively intersection and union, and $\Rightarrow$ and $/$ are defined as above. Z is of course simply the set of zero states Z of the frame.

Remark 1. *Urquhart (1972) had a simpler idea about how to model relevant implication. His idea was to have a set U of "pieces of information"* $\alpha, \beta, \chi, \ldots$ *Urquhart included the least (empty) piece of information* $\varnothing$ *used to define validity. Urquhart postulates that the pieces of information can be combined by a semi-lattice operation* $\cup$*. For this reason it is common to refer to Urquhart's approach as the "operational semantics", and to call the contrasting Routley-Meyer approach the "relational semantics". While there are problems in trying to extend Urquhart's idea to provide a semantics for the whole of the system* **R** *of relevant implication (conjunction is ok, but both disjunction and negation cause problems), Urquhart showed that we can model the implicational fragment* $\mathbf{R}_{\rightarrow}$ *with the following:*

$$\chi \mid= \varphi \rightarrow \psi \;\; iff \;\; (\forall\alpha : \alpha \,||\!- \varphi \;\; imply \; \chi \,\tilde{}\; \alpha \,||\!- \psi). \qquad (5)$$

Remark 2. *The essential difference between the operational and relational semantics can be put in terms of the former assuming determinism: Given the state* χ *and the input* α *there is a unique outcome* $\chi(\alpha) = \chi \,\tilde{}\; \alpha$*. The relational semantics rather assumes that there can be many accessible outcomes* $\chi(\alpha) = \{\beta : R\chi\alpha\beta\}$.

Remark 3. *Fine (1974), independently from Routley, Meyer, and Urquhart, developed a semantics for relevant implication that in effect combined their two ideas. In place of* $R\alpha\beta\gamma$ *Fine writes:* $\alpha{\circ}\beta \leq \gamma$*. We will call the Fine approach the "refined semantics" because it makes explicit the binary operation implicit in the Routley-Meyer semantics. If you look at the Routley-Meyer completeness proofs and at the* *canonical model of theories, one sees that they have an operation combining theories and also the relation of inclusion between theories.*

There are many different ways to interpret these semantical ideas. Thus for example think of χ as a "proof" of $\phi \rightarrow \psi$. In terms of the operational semantics, χ determines a function taking proofs of ϕ to proofs of ψ. In the relational semantics, χ determines a relation from proofs of ϕ to proofs of ψ.

I suggested verbally (and it is referenced in the first Routley-Meyer paper) that one somehow think of $R\rho\alpha\beta$ as akin to Kripke's relative accessibility relation, but relativized one step further. Rather than saying that β is possible relative to α, we save that given ρ that β is possible relative to α. In the early 1970's, Peter Woodruff (verbally to me) suggested thinking of the ternary relation R as an indexed set of *binary* relations $\{R_\rho\}_{\rho\in U}$, where each $R_\rho = \{<\alpha, \beta>: R\rho\alpha\beta\}$. It is a short metaphorical step from this to the observation that one can think of each state ρ as having a dual nature, first as a state and second as determining a binary relation $R_\rho\alpha\beta$ between states α and β. This can be given the "philosophical" reading: "the pair $<\alpha, \beta>$ *exemplifies* the relation (determined by) ρ". Also it can be directly related to Frege's distinction between "concept" and "object". We think of ρ as the "object correlate" of the relation R_ρ. We can take the further metaphorical step of thinking of the state ρ as "static", and the relation R_ρ as "active". We are within a hair's breadth of von Neumann's concept of a stored program[7]!

5. Interlude on Indeterminism

A *theory* is simply a set of sentences closed under adjunction and (relevant) entailment. It is easy to see that theories behave well with respect to conjunction. A theory contains a conjunction iff it contains each conjunct. But once we add disjunction, theories lose respect. While a disjunction is in a theory when either disjunct is, the converse does not hold. It is standard to call a theory where the converse holds prime. Given two theories T and T, it is natural to form their "fusion" $T \circ T$. This is done by considering the set of sentences of the form $\varphi \circ \varphi'$ where $\varphi \in T$ and $\varphi' \in T$, and closing it under adjunction and

[7] We have therefore several reasons to regard Frege as a proto computer scientist. Skipping this anticipation of the idea of a "stored program", and passing quickly over the issues of formal logic and its relations to computer science, we get to the important fact that Frege received funding from Zeiss (told to me by Werner Stelzner). Corporate support is one of the most important marks of a computer scientist.

entailment. Even when both T and T' are prime, their fusion is unlikely to be prime. This is because of the **R**-theorem:

$$(\varphi \rightarrow \psi) \circ \varphi \rightarrow \psi.$$

In particular $\varphi \rightarrow \psi_1 \vee \psi_2$ might be in T and φ might be in T', and so $\psi_1 \vee \psi_2$ would end up in their fusion $T \cdot T'$. And yet there is no reason that either one of ψ_1 or ψ_2 should end up there[8].

This is why when we look at the canonical model we focus on the ternary relation $R\alpha\beta\gamma$ rather than the binary operation $\alpha \bullet \beta$. In Fine's terms this is $\alpha \bullet \beta \leq \gamma$, i.e., $T \bullet T' \subseteq T''$, where T, T', and T'' can all be prime. In the proof of the completeness theorem we build prime extensions of $T \bullet T'$ in various possible ways, and in some of these 1 is true and in others of these ψ_2 is true.

6. Interpretation

We like to give $R_\rho\alpha\beta$ the "technological" reading: if the machine is currently in state ρ, then if it were to enter state α, then state β is a potential outcome (the pair $<\alpha, \beta>$ is a possible transition given ρ).

The reader may wonder why we need this ternary relation of accessibility, and not just a binary one, which is more usual. And indeed provides the framework for Pratt's dynamic logic. It might be worth investigating how to extend dynamic logic to involve the ternary accessibility relation.

It might be argued that if α is a state, then all the information that is relevant to possible transitions is contained in α alone, and so there is no reason to consider the "perspective" of ρ. Indeed, on a deterministic conception of computation, the initial state would uniquely determine all of the subsequent states. But this objection overlooks the fact that we are working with partial states. One might think of this partiality in terms of segregating parts of the machine so in considering $R\rho\alpha\beta$ we think of ρ as a state in that part of the machine that stores operations, and α and β as states in that part of the machine that stores data. Or one might think in terms of concurrent computing, with ρ being the state on one processor, and α and β being states in another processor (or complexes of processors including maybe the whole system) whose operation is affected by ρ. This kind of approach leads to the idea that states are of different types, and we are here wanting a type-free system. But we think it is worth exploring typed systems as well, particularly the division between operations and data. The information in α may well be partial and the information in ρ may further restrict

[8] Because the following is not an **R**-theorem:
$(\varphi \rightarrow \psi_1 \vee \psi_2) \rightarrow (\varphi \rightarrow \psi_1) \vee (\varphi \rightarrow \psi_2)$.

the states β that are accessible from α. Note that states may even be inconsistent. Think of a network where there are two machines that are supposed to be mirror sites but by some glitch they are not. Readers who are bothered by the thought of partial or inconsistent states should be reminded that states exist at the "logical", as opposed to the "physical" level. Two states of different machines can be identical even though there are hardware differences between the two machines, and of course this can also be true of the same machine at two different times.

One way to look at things is that we are somehow combining ρ and α and seeing whether this combination is contained in β: $\rho + \alpha$ š β. The simplest mode of combination would just be in effect to take their "union" $\rho\vee\alpha$. Thus thinking of a state as an assignment of the binary bits **0** or **1** to variables, a variable x_i will be assigned one of these values by $\rho\vee\alpha$ if either ρ or α assigns it that value. This is the minimal requirement on a "union". But oops! What happens if say ρ assigns x_i the value **1** and α assigns it **0**? One could imagine various answers here as to what value $\rho + \alpha$ assigns. E.g., it could assign no value, it could assign an error message, it could assign some principled choice (say, when in doubt take the maximum), or, most radically but perhaps most naturally for the concept of a "union", we give up on the idea that assignments are consistent and let it assign both values. But if we require that the assignments ρ, α and β must be consistent, one of the most interesting things to do is to take the maximum. This ensures to $R\rho\alpha\beta$ ρ š β and α š β, which as can be seen from Dunn (1993c) gives a ternary accessibility relation which provides the basis for a semantics for intuitionistic implication.

There is a related ternary relation in Barwise (1993). Barwise uses the notation $s_1 \mapsto^{c} s_2$ to indicate the 3-placed relation that exists when the two "sites" s_1 and s_2 are connected by the "channel" c. In the notation of Routley and Meyer this could be expressed as Rs_1cs_2. Channels hence determine binary relations between sites, but they need not be identified with those relations.

Barwise leaves explicitly open the idea that sometimes channels can be sites and *vice versa*. What we have in our representation of relation algebras is the totally "untyped" case where every site is a channel and *vice versa*.

7. The main idea

I wish to give here the main idea. Suppose we have a ternary frame with a set of states U, and a subset A of U. A can be thought of as a set of states, i.e., a proposition. So A is quite static. But, and this is than main idea, it can be turned in for the set of relations determined by those states, and those relations can of course be regarded as taking states to states. So *A* is *at the same time* quite dynamic. Here we model the duality implicit in von Neumann's concept of a stored program, or Frege's distinction between concept and object/function and argument. One and the same thing can be both.

Given a state $\rho \in U$, $R_\rho = \{<\alpha, \beta>: R\rho\alpha\beta\}$. By $R[A]$ let us mean the set of relations determined by *A*, i.e., $R[A] = \{R_\alpha : \alpha \in A\}$.

Suppose we have two such propositions *A* and *B*. We can turn *A* in for the set $R[A]$ of relations which it determines, and similarly we can get $R[B]$. We can then do various things with $R[A]$ and $R[B]$. Thus as one example we can take $R[A]$ and "apply" it to *B*:

$$R[A](B) = \{\gamma : \exists\alpha \in \mathrm{A}, \exists\beta \in \mathrm{B}(R_\alpha\beta\gamma)\},$$

getting all the states we can get to from *B* using a relation in $R[A]$; treating *A* as a program, *B* as data, and applying *A* to *B*. We shall return to this idea in the next section when we examine how to model combinatory logic.

This uses only the implicit relation character of *A*. But we can use the implicit relational character of both *A* and *B*; taking the relations in $R[A]$ and the relations in $R[B]$ and forming their relative products in all possible ways. Let us recall that given two relations *R* and *S* their relative product is defined by

$$x(R \otimes S) \Leftrightarrow \exists u(xRu \,\&\, uRy).$$

So we define

$$R[A] \circ R[B] = \{R_\alpha \otimes R_\beta : \alpha \in \mathrm{A}, \beta \in \mathrm{B}\}.$$

This is like viewing both *A* and *B* as programs, and composing *A* with $B : A \cdot B$. We shall return to this idea in a short while when we examine how it can be used in the representation of relation algebras.

8. Models for combinatory logic

Dunn and Meyer (1997) gives particularly simple models for combinatory logic based on the Routley-Meyer ternary semantics for the relevance logic **R**[9]. Various conditions have to be put on the ternary accessibility relation to get models for the logic **R**. It is well-known by now that by fussing with those conditions one can get models for various logics closely related to **R**.

[9] As is spelled out in Dunn and Meyer (1997), other models for combinatory logic have been provided by various researchers, including D. Scott, G. Plotkin, and R.K. Meyer, M. Bunder and L. Powers.

We illustrate this with a simple example. The logic **R** has the "contraction axiom"

$$(\varphi \to (\varphi \to \psi)) \to (\varphi \to \psi),$$

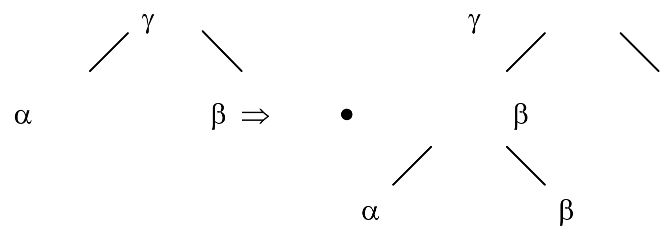
which corresponds to Gentzen's "structural rule" of contraction:

$$\frac{\Gamma, \varphi, \varphi, \Delta \mid\!- \psi}{\Gamma, \varphi, \Delta \mid\!- \psi}.$$

This amounts to saying that ° "duplicates", i.e., $A \circ B \subseteq (A \circ B) \circ B$. On the accessibility condition we need the postulate:

$$R\alpha\beta\gamma \Rightarrow \exists\chi(\mathrm{R}\alpha\beta\chi \ \& \ \mathrm{R}\chi\beta\gamma).$$

In a more or less obvious diagram this becomes:



We choose to algebraize the combinators as a *combinatory poset, i.e.*, as a partially-ordered groupoid $(X, \leq, \circ, C)$, where X is thought of as a set of untyped functions, $\leq$ is a partial order on X (thought of as "reducibility"), ° is a binary isotonic operator on X (thought of as "application"), and $C \leq X$[10]. C is thought of as a set of combinators, and we shall call its members combinator elements. Each combinator element $c \in C$ is subject to some postulate of the form

$$cx_1 \ldots x_m \ \leq \tau(x_1, \ldots, x_m), \qquad (6)$$

where $cx_1 \ldots x_m$ are variables, ° is indicated by concatenation, the sequence $cx_1 \ldots x_m$ is a left-associated sequence, i.e. is of the form $(((cx_1)x_2)) \ldots x_m)$, and $\tau(x_1, \ldots, x_m)$ is a term constructed only out of

[10] Note that we use ≤ where Curry and Feys (1972) use ≥. Note also that we do assume the usual conventions of suppressing ° in favour of juxtaposition, and assuming associativity to the left.

the variables x_1, …, x_m (not necessarily all of them) and $\circ$[11]. $\tau(x_1, \ldots, x_m)$ can be regarded as a sequence containing some or all of the variables x_1, …, x_m, with parentheses scattered through it so as to indicate binary grouping of any pattern.

As examples consider a few of the more common combinators:

$$\mathbf{i}x \leq x \tag{7}$$

$$\mathbf{k}xy \leq x$$

$$\mathbf{c}xyz \leq xzy$$

$$\mathbf{w}xy \leq xyy$$

$$\mathbf{b}xyz \leq x(yz) \tag{8}$$

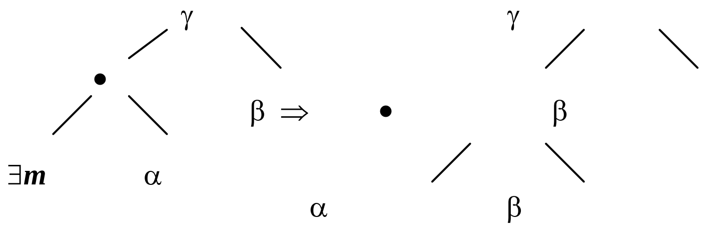
$$\mathbf{s}xyz \leq (xz)(yz)$$

The notion of a combinatory poset is similar to the notion of a "combinatorial algebra" in Barendregt (1981), except that he bases things throughout on equality rather than the inequality, and his combinators are required to be **S**, **K**.

It is well-known that contraction corresponds to the combinator **W**. Using inclusion in place of reduction, the postulate for that combinator is $\mathbf{W}AB \subseteq ABB$. If we view **W**, A, and B as sets of states, we need the following postulate:

$$\exists \boldsymbol{m} \in \mathbf{W}, \exists\chi(R\boldsymbol{m}\alpha\chi \ \&\ \mathrm{R}\chi\beta\gamma) \Rightarrow \exists\chi(\mathrm{R}\alpha\beta\chi \ \&\ \mathrm{R}\chi\beta\gamma)$$

The left-most terminus is "kicked out of the way" by the combinator state $\boldsymbol{m}$:



11 Because of the form of these laws, the combinator elements correspond to what have been called "proper" combinators (cf. Curry and Feys (1972), and which may beviewed as closed lambda terms. We donotbother with the more accurate but more awkward terminology "proper combinator poset".

Conditions can thus be put on ternary frames that force various subsets to behave like combinators[12].

Using this method it was shown in Dunn and Meyer (1997) that every combinatory algebra can be represented using an appropriate ternary frame.

9. Models for the algebra of relations

In Dunn (2001) it was shown how to represent relation algebras using a modified version of Routley-Meyer frames that are not unlike those they used to model relevance logic[13]. We will not repeat definitions here, but part of the trick was to give a carefully layered definition of relation algebras that showed the close connections to relevance logic and which matched a carefully layered definition of the appropriate Routley-Meyer frames.

We begin with the notion of *lattice-ordered monoid* $(L, \leq, \vee, \circ, e)$, where $(L, \leq, \vee)$ is a lattice and $(L, \leq, \vee, \circ, e)$ is a monoid, with $\circ$ distributing over $\vee$ from both directions. It follows that $\circ$ is isotone in each of its arguments.

We recall that a l.o.m. is said to be residuated if it had two additional binary operators $\rightarrow$ and $\leftarrow$ (called the *right* and *left residuals*) satisfying:

$$x \circ y \leq z \textit{ iff } y \leq x \rightarrow z \textit{ iff } x \leq z \leftarrow y. \tag{9}$$

De Morgan and Peirce observed in the last century that these structures arise naturally in the context of the algebra of relations (cf. e.g., Maddux (1991)). We think of relations in the usual way as sets of ordered pairs. Given a set X and two binary relations R and S on X, the relative product $R \circ S = \{(x, y) : \exists z((x, z) \in R \ \&\ (z, y) \in R)\}$ is an associative operation, and e is the identity relation (restricted to X). Then $R \rightarrow S = \{(x, y) : \forall z((z, x) \in R \textit{ implies } (z, y) \in R)\}$ and $S \leftarrow R = \{(x, y) : \forall z((x, z) \in R \textit{ implies } (y, z) \in R)\}$.

But relative product and the residuals are not the only important operations on relations. There is also the converse $R^{-1} = \{(y, x) : (x, y) \in R)\}$. Given a lattice, a unary operation x^{-1} (which we shall think of as

[12] Alternatively, if one is interested not just in one-way reduction but rather equality of combinators, one can define a combinator as a set of states satisfying a given condition. For example:
$\mathbf{W} = \{\boldsymbol{m} : \exists\chi(R\boldsymbol{m}\alpha\chi \ \&\ \mathrm{R}\chi\beta\gamma) \Rightarrow \exists\chi(\mathrm{R}\alpha\beta\chi \ \&\ \mathrm{R}\chi\beta\gamma)\}$.

[13] As explained in detail in Dunn (2001), it was pointed out to me by R. Maddux that this idea was foreshadowed by Lyndon, and by Jénsson and Tarski, although my treatment is more explicit and general and makes the connections to relevance logic and other substructural logics.

"conversion") is an automorphism of period two when it is a 1-1, order preserving, and of period two. i.e., $(x^{-1})^{-1} = x$.

Definition 4. *A structure* $(L, \wedge, \vee, \circ, \rightarrow, \leftarrow, -, {}^{-1}, e)$ *is a* ***relation algebra*** *iff*

$$(L, \wedge, \vee, -) \text{ is a Boolean algebra,} \tag{10}$$

$$(L, \wedge, \vee, \circ, \rightarrow, \leftarrow, e) \text{ is a residuated lattice ordered monoid,} \tag{11}$$

$$x^{-1} \text{ is a lattice automorphism of period two on } (L, \wedge, \vee), \tag{12}$$

$$(x \circ y)^{-1} = y^{-1} \circ x^{-1}. \tag{13}$$

$$a \rightarrow b = -(a^{-1} \circ -b), \qquad b \leftarrow a = -(-b \circ a^{-1}). \tag{14}$$

The above definition derives from Tarski (1941), and can be found in Dunn (2001). This definition is presented in a way that makes it similar to the definition of a Boolean De Morgan monoid, which is the algebraic structure corresponding to the relevance logic **R** supplemented with Boolean negation in addition to its standard De Morgan negation. These structures were in effect introduced by Meyer and Routley (1973) and studied more explicitly as algebraic structures by Meyer (1979), where he explicitly introduced an operator x^*. Meyer observes that one can then define the usual De Morgan complement from the Boolean one and the * operator as follows:

$$\sim x = -(x^*). \tag{15}$$

Putting $^{-1}$ in place of * we get:

$$\sim x = -(x^{-1})$$

which was anticipated as a definition of a negation-like operator by Białynicki-Birula and Rasiowa (1957).

14 above does not look very pretty from a logical point of view, but it can be replaced equivalently with:

$$a \rightarrow b = \sim(\sim b \circ a), \qquad b \leftarrow a = \sim(a \circ \sim b). \tag{16}$$

The only differences between a Boolean De Morgan monoid and a relation algebra is that for a De Morgan monoid we assume in addition that $\circ$ is commutative and square increasing ($x \leq x \circ x$). This last corresponds to contraction.

Next we turn to the ternary frames appropriate to representing relation algebras. We start with a plain vanilla ternary frame $F = (U, R, \check{s}, Z)$ and add to it a unary map $^{\cup}$ on U into itself such that for $\chi \in U$.

$$\chi^{\cup\cup} = \chi \quad \textit{(period two).} \tag{17}$$

Such a map is commonly called an involution[14]. We also add (writing $\chi^{\vee}$ in place of $\chi^{\cup}$).

$$R\alpha\beta\gamma \textit{ only if } R\beta^{\vee}\alpha^{\vee}\gamma^{\vee} \textit{ (tagging).} \tag{18}$$

$$R\alpha\beta\gamma \textit{ iff } R\gamma^{\vee}\alpha\beta^{\vee} \textit{ (antilogism)} \tag{19}$$

Let us define for $A \in P(U)^{\uparrow}$,

$$\mathrm{A}^{-1} = \{\alpha^{\vee} : \alpha \in A\}. \tag{20}$$

By virtue of (17), this is equivalent to:

$$\mathrm{A}^{-1} = \{\alpha : \alpha^{\vee} \in A\}. \tag{21}$$

This does not yet give us the class of frames appropriate for relation algebras, because we have to assure that ° is associative.

Let us begin by looking at two notational conventions of Routley and Meyer:

$$R(\alpha\beta)\gamma\delta =_{\mathrm{def}} \exists\chi(R\alpha\beta\chi \;\&\; R\chi\gamma\delta), \tag{22}$$

$$R\alpha(\beta\gamma)\delta =_{\mathrm{def}} \exists\chi(R\alpha\chi\delta \;\&\; R\beta\gamma\chi). \tag{23}$$

Routley and Meyer actually write "R^2" where we write simply "R" above, but for reasons of both suggestiveness and simplicity we shall often simply let the number of terms affixed tell that there is a power.

One of the requirements on a Routley-Meyer frame is that these two compositions are equivalent:

$$R(\alpha\beta)\gamma\delta \textit{ iff } R\alpha(\beta\gamma)\delta \tag{24}$$

This is precisely what is needed to assure that

$$A \circ (B \circ C) = (A \circ B) \circ C \qquad (\textit{associativity})$$

In this section we shall depart from our convention of viewing the binary relation hidden inside a ternary relation as indexed by the first state. For reasons that have nothing to do with anything but making things look pretty we shall view it as indexed by the second state[15]. So in this section only, $R_{\rho} = \{(\alpha, \beta) : R\alpha\rho\beta\}$.

Let us define the composition of two "binary relations" ρ and σ:

$$\alpha(R_{\rho} \otimes R_{\sigma})\beta \textit{ iff } \exists\chi(R\alpha\rho\chi \;\&\; R\chi\sigma\delta)$$

Clearly using this definition and *R-associativity*, we obtain

[14] Following Routley and Meyer, this involution is customarily denoted by * in the relevance logic literature. Here we instead use $^{\cup}$ because in the literature on relations * is customarily used for the "ancestral".

[15] This has something to do with the mathematical tendency to use infix notation for relations and prefix notation for functions.

$$\alpha(R_\rho \otimes R_\sigma)\beta \;\; \textit{iff} \;\; R(\alpha\rho)\sigma\beta \;\; \textit{iff} \;\; R\alpha(\rho\sigma)\beta, \tag{25}$$

where the right-hand sides of the last two conditions are understood using the compositional notations of Routley-Meyer. One could write (25) as

$$\alpha(R_\rho \otimes R_\sigma)\beta \;\; \textit{iff} \;\; R(_{\rho\sigma})\alpha\beta. \tag{26}$$

but one should be clear that "(ρσ)" is a mere notation and does not denote an actual point.

The above expresses what one might label a "notional homomorphism". But it can be made into an actual homomorphism if we "refine" the Routley-Meyer relation into smaller bits, in effect interpreting $R\alpha\beta\gamma$ as $\alpha \bullet \beta$ š γ. We can the restate **??** as:

$$R_{\rho\bullet\sigma} = R_\rho \otimes R_\sigma.$$

We can also easily see that

$$R_{\rho^\vee} = (R_\rho)^{-1}.$$

Proof. $\alpha R_{\rho^\vee}\beta$ iff $R\alpha\rho^\vee\beta$ (antilogism) $R\rho^\vee\beta^\vee\alpha^\vee$ iff (tagging period two) $R\beta\rho\alpha$ iff $\beta R_{\rho^\vee}\alpha$ iff $\alpha(R_{r^\vee})^{-1}\beta$.

It turns out that one can show that $\lambda\rho R_\rho$ is a one-one function taking states to binary relations on states, using the Z-Condition in the definition of a frame. So we in fact have an isomorphism preserving "relative product" and "converse") between states and binary relations.

Lyndon (1950) showed that not every relational algebra can be interpreted as a set of relations, but using the observations above we can easily show that this can be done one type-level higher. Given a set of states A, let $\boldsymbol{R}_A = \{R_\alpha : \alpha \in A\}$. For any frame U, we have just observed that $\boldsymbol{R}_U$ is closed under relative product and converse and is in one-one correspondence with U. This means that we can give "point-wise" definitions: $\boldsymbol{R}_A \circ \boldsymbol{R}_B = \{(R_\alpha \otimes R_\beta) : \alpha \in A, \beta \in B\}$ and $\boldsymbol{R}_A^{-1} = \{R_\alpha^{-1} : \alpha \in A\}$. Note further that $\boldsymbol{R}_A \cap \boldsymbol{R}_B = \boldsymbol{R}_{A\cap B}$, $\boldsymbol{R}_{A\cup B} = \boldsymbol{R}_A \cup \boldsymbol{R}_B$, and $\boldsymbol{R}_{U-A} = \boldsymbol{R}_U - \boldsymbol{R}_A$.

Theorem 5. *Every relation algebra* $(A, \wedge, \vee, -, \circ, ^{-1})$ *is isomorphic to a set of sets of relations, with* $\wedge$ *interpreted as union,* $\vee$ *as intersection,* $-$ *as complement relative to a certain subset of* $\wp(U \times U)$*, and* $\circ$ *being point-wise relative product, and* $^{-1}$ *being point-wise converse.*

10. Glimpses Ahead: Pratt's Dynamic Logic and Hoare Logic

Proof. V. Pratt (1980) makes a distinction between states and programs and defines things such as $[p]\phi$ to mean intuitively that the sentence ϕ is true of every *state* α that arises while the *program* p is executing.

In symbols we express this as

$$\chi \Vdash [p]\phi \ \ \textit{iff} \ \ \forall\beta(\chi R_p\alpha \Rightarrow \beta \Vdash \phi), \tag{27}$$

where $\chi R_p\alpha$ is read "the state α is accessible from state χ by running the program *p*". Note that R_p is an accessibility relation indexed by the program *p*. Pratt assumes a non-deterministic notion of computation or else this could be replaced by the notion $p(\chi) = \alpha$.

As we have seen, any state ρ can be thought of as determining an accessibility relation R_ρ. Note this is "rho" not "pee" – we are talking about a *state* and not a *program*. Pratt's dynamic logic has accessibility relations indexed by programs. A state can be viewed as an assignment to variables (storage location) of the values 0, 1. If we focus on the substate where the program is stored we get a partial assignment. Since thus a program can be viewed as a partial state, using ρ in place of *p* generalizes Pratt.

Another way to go is to look at a program as not a single (partial) state but rather as a set of states (intuitively the set of states that implement the same program). This suggests we write 27 in an untyped way, replacing the single state ρ with a set of states (a proposition) *B*. We can then rephrase the above as:

$$\chi \Vdash [B]\phi \ \ \textit{iff} \ \ \forall\alpha, \forall\rho \in B(\chi R_p\alpha \Rightarrow \alpha \Vdash \phi),$$

and we can rephrase this further, replacing the proposition B with the sentence φ that expresses it:

$$\chi \Vdash [\psi]\phi \ \ \textit{iff} \ \ \forall\alpha, \forall\rho \in |\psi| \, (\chi R_p\alpha \Rightarrow \alpha \Vdash \phi),$$

where $|\psi| = B = \{\beta : \beta \Vdash \psi\}$.

Another possible application of ternary frames is to Hoare's (1969) "Logic of Programs", but we only mention this here. It would also be nice to model action algebras (cf. Pratt (1991)).

REFERENCES

1. *Anderson A.R.*, *Belnap N.D.* and *Dunn J.M.* et al (1992), Entailment: The Logic of Relevance and Necessity, vol. 2, Princeton University Press (Princeton).
2. *Barendregt H.P.* (1981), The Lambda Calculus: Its Syntax and Semantics, North -Holland Studies in Logic and the Foundations of Mathematics, 103, (Amsterdam) Elsevier Science Publishers (Amsterdam).
3. *Barwise J.* (1993), "Constraints, Channels, and the Flow of Information", in Situation Theory and its Applications, ed. by S.Peters and D. Israel, CSLI Lecture Notes, vol. 3, University of Chicago Press (Chicago).

4. *Bialynicki-Birula A.* and *Rasiowa H.* (1957), "On the Representation of Quasi-Boolean algebras", Bulletin de l'Acadfiemie Polonaise des Sciences, 5, 259-261.
5. *Bishop P.* (1986), Fifth Generation Computers,. Ellis Horwood Limited (Chichester, England).
6. *Boole G.* (1847), Mathematical Analysis of Logic, London, 1847.
7. *Burks A.W., Goldstine H.H.* and *Neumann J. von.* (1963), "Preliminary Discussion of the Logical Design of an Electronic Computing Instrument",. in Taub, A. H., editor, John von Neumann Collected Works, The Macmillan Co.(New York), Volume V, 34-79.
8. *Church A.* (1941), The Calculi of Lambda Conversion, Princeton University Press (Princeton).
9. *Curry H.B.* and *Feys R.* (1972), Combinatory Logic, vol. I, North-Holland Publishing Company (Amsterdam).
10. *Dunn J.M.* (1991), "Gaggle Theory: An Abstraction of Galois Connections and Residuation with Applications to Negation and Various Logical Operations", Logics in AI, Proceedings European Workshop JELIA 1990, ed. J. van Eijck, LNCS 478, Springer-Verlag (Berlin).
11. *Dunn J.M.* (2001), "A Representation of Relation Algebras Using Routley-Meyer Frames", forthcoming in Logic, Language, Computation: Essays in Honor of Alonzo Church, eds. C. A. Anderson and M. Zeleney, Kluwer Academic Publishers (Dordrecht).
12. *Dunn J.M.* and *Meyer R.K.* (1997), "Combinatory Logic and Structurally Free Logic", Journal of the Interest Group in Pure and Applied Logic, 5.
13. *Fine K.* (1974), "Models for Entailment", Journal of Philosophical Logic, 3, 347-372.
14. *Gärdenfors P.* and *Makinson D.* (1988), "Revision of Knowledge Systems Using Epistemic Entrenchment", in Proceedings of the Second Conference on Theoretical Aspects of Reasoning about Knowledge, ed.M. Vardi, Morgan Kaufmann (San Francisco), 83-95.
15. *Geach P.T.* and *Black M.* (1960), Translations from the Philosophical Writings of Gottlob Frege, Basil Blackwell (Oxford).
16. *Hoare C.A.R.* (1969), "An Axiomatic Basis for Computer Programming, Communications of the ACM, 12, 576-585.
17. *Lyndon R.C.* (1950), "The Representation of Relation Algebras", Annals of Mathematics, ser. 2, 51, 707-729.
18. *Maddux R.* (1991), "The Origin of Relation Algebras in the Development and Axiomatization of the Calculus of Relations", Studia Logica, 91, 421-455.
19. *Meyer R.K.* (1979), "A Boolean Valued Semantics for R", Research Paper no. 4, Logic Group, Research School of Social Sciences, Australian National University, Canberra.
20. *Meyer R.K.* and *Routley R.* (1972), "Algebraic Analysis of Entailment", Logique et Analyse, n.s., 15, 407-428.
21. *Meyer R.K.* and *Routley R.* (1973), "Classical Relevant Logics (I)", Studia Logica, 32, 51-66.
22. *Pratt V.*, "Six Lectures on Dynamic Logic", Foundations of Computer Science III, part 2, Mathematical Centre Tracts (Amsterdam), 109, 53-82.

23. *Pratt V.* (1991), "Action Logic and Pure Induction", Logics in AI, Proceedings European Workshop JELIA 1990, ed. J. van Eijck, LNCS 478, Springer-Verlag.
24. *Riley H.N.* (1997), "The von Neumann Architecture of Computer Systems", http://www.csupomona.edu/~hnriley/www/VonN.html.
25. *Routley R.* and *Meyer R.K.* (1972), "The Semantics of Entailment, II-III", Journal of Philosophical Logic, 1, 53-73 and 192-208.
26. *Routley R.* and *Meyer R.K.* (1973), "The Semantics of Entailment I" in Truth, Syntax and Modality, ed. H. Leblanc, North-Holland Publishing Company (Amsterdam), pp. 199-243.
27. *Tarski A.* (1941), "On the Calculus of Relations", The Journal of Symbolic Logic, 6, 73-89.
28. *Urquhart A.* (1972), "Semantics for Relevant Logics", The Journal of Symbolic Logic, 37, 159-169.