Logical Constants: ***the variable fortunes of an elusive notion***

Johan van Benthem, Amsterdam & Stanford

in W. Sieg, R. Sommer, & C. Talcott, eds., 2002, *Reflections on the Foundations of Mathematics. Essays in Honor of Sol Feferman*, ASL Lect. Notes in Logic 15, 426–446.

1 Logic: inferential and expressive power

Logic is usually taken to be the study of reasoning – and the highest rank of nobility for an assertion or an inference is 'logical validity'. There are proof-theoretic methods for describing all validities for a given language, and completeness theorems assure us that we have found 'all there is' to a particular style of reasoning. But in addition to this inferential power, logic is also about expressive power. Reasoning needs a language supplying 'logical forms', and what determines the choice of *these*? Logical languages contain various forms of expression: propositional connectives, quantifiers, or modalities, whose meanings are analyzed in addition to their inferential behavior. But what makes these particular notions 'logical', as opposed to others? Perhaps the usual expressive completeness argument reassures us that the Boolean connectives capture all there is to two-valued propositional reasoning. But no similar result is known for first-order predicate logic, the major working system of modern logic.

This concern is not part of the basic 'agenda' in textbooks, partly because it is seldom raised, and partly because there is no consensus on an answer. Sol Feferman is one of the small group of authors who do think about this basic issue (cf. Feferman 1999), and I am happy to contribute a little piece of my own thinking on these matters. But before doing so, let me note that not all great logicians would find the effort worthwhile. E.g. Bernard Bolzano's system, pioneering in so many ways, did not contain any privileged set of logical operations. For him, the distinction logical/non-logical is merely one of methodology. Forms of assertion for a style of reasoning arise by fixing the meanings of some expressions (these will have a 'logic' then), and letting others vary. But this distinction can go in more than one way. Granting this liberality, I still would like to find out what makes the usual 'logical constants' tick.

My aims in this are different from Sol's excellent analysis, who mentions a 'demarcation' of logic as a major concern – and first-order logic as a preferred target.

I myself feel no need for a principled separation of logic from other territories, mathematics, linguistics, computer science, psychology, or whatever. And fixing some target set of expressions, no matter how noble its ancestry, is too conservative for me. I prefer accounts of logicality which bring to light interesting ‘parameters’ that can be set in different ways, giving us a handle on *analogies* with other fields. Note the phrasing here. ‘Logical constants’ suggests some expressions in general use among logicians, perhaps a finite set. What intrigues me is not the book-keeping job of capturing these – but the analysis of their appealing general features. Out of that we get notions of *logicality,* that many expressions might have to greater or smaller degrees. Now you may find this a slippery slope. Soon, someone who thinks like this will also start doubting our dearest intuitions, and consider even logical *validity* a matter of degree. That is exactly what I do think – but enough of this confessional!

2 Logicality: proposals from different stances

There are a few scattered attempts in the literature at elucidating intuitive notions of logicality. Not surprisingly, these follow the major stances found in logic. First, there are *semantic* accounts, in terms of *invariance for permutations* of objects in the domain of discourse. What this mathematical condition intends to capture is the intuitive ‘topic-neutrality’ of logical expressions. They do not depend on any specific individual features of objects or predicates. This proposal, made by many authors independently (including Tarski 1986) seems the most succesful account to date, and it will occupy us in most of this paper. But there are also other semantic aspects of logicality, as we shall see below (cf. van Benthem 1986). Next, there have been *proof-theoretic* accounts of logicality, addressing different intuitions. E.g., one might think of a logical constant as one supporting a particularly rich set of inference patterns – or as one playing a crucial role in the ‘organisation of proof’. The latter is the thrust of the expressive completeness theorems in Zucker 1978, Zucker & Tragesser 1978. Alternatively, Hacking 1979 analyzes logical constants in terms of ‘little content’, being those expressions that are ‘conservative’: when added to a proof calculus, they do not imply any new consequences in which they do not occur.

A word on ‘stances’. By this I mean general perspectives in a field, which support a certain methodology – say like ‘algebra’ and ‘geometry’ in mathematics. Most logicians recognize a semantic stance on validity (“that which transmits truth”), as well as a proof-theoretic one (“that which follows by natural proof steps”). Are there further natural stances in logic? I think there are. E.g. the *game-theoretic* stance goes back to the appealing intuition that valid arguments are the ones that you can *win* as a player in a debate. This stance, too, carries its view on logicality (Hintikka 1973). In a game-theoretic setting, basic operations have to do with various choices that players make, or role switches. ('Putting yourself in my position' is an eminently logical operation!) Thus, on this stance, logicality would have to do with the basic workings of 'control'. This paper takes a semantic stance. But we discuss some other approaches at the end.

Finally, another plausible necessary condition for logicality is *algorithmic* in nature. Should logical expressions have meanings that are 'easiest to compute' in some way? Or should they rather be the process combinations that make for 'maximal effective computing power'? We will discuss this angle, too. I mention these different angles here in order to show that there are many intuitive aspects to 'logicality', and that we need neither expect, nor desire one simple formal characteristic exhausting the notion.

3 Permutation invariance

In this section, we state the proposal, discuss its known properties, putting it in context. Our purpose here is not original. We just want to make some basic things common knowledge – as the subject is rife with re-inventions of wheels and epicycles.

3.1 The proposal Logical structure is not about individual peculiarities. Intuitively, 'logical' operators do not care about specific objects inside their arguments. Thus, permuting these should not make a difference to the workings of the operator. Consider logical negations. A set complement $–A$ just returns the outside of A, without inspecting any particular feature of the objects in A. Likewise, a relational complement $–R$ works uniformly on just the structure of R, the 'arrow pattern' of its ordered pairs. Similar observations hold for predicate intersection (conjunctions) and projection (quantifiers). This is in contrast to non-logical predicate operations like 'Dutch' whose action depends on the content of its arguments. ("Dutch dining" makes one's guests pay for themselves, "Dutch climbers" overtake their companions just at the summit.) The common mathematical generalization behind these examples involves *invariance under permutations* π of the underlying universe of individual objects.

<u>Definition</u> Let π be a permutation of the base universe U. This can be lifted to sets, relations, and all functions in a finite type hierarchy over U in a canonical way. An object F is *permutation-invariant* if $\pi(F) = F$ for all permutations π. ■

One should not think of this proposal as falling out of the sky. There is an analogy here with something we all know, the semantic *Isomorphism Lemma* for standard logics.

Think of operations defined by some formula of the logic with free variables for objects, predicates, etc. If π is an isomorphism between two models **M** and **N** of the appropriate similarity type, then one proves – by induction on formulas – that

$$\mathbf{M} \models \phi\ [x, y, .., X, Y, ...] \quad \text{iff} \quad \mathbf{N} \models \phi\ [\pi(x), \pi(y), .., \pi[X], \pi[Y], ...]$$

Equivalently, this says that π maps the **M**-denotation of ϕ to its **N**-denotation:

$$\pi\ [\ [[\phi]]^{\mathbf{M}}\ ] = [[\phi]]^{N}$$

which is the obvious cross-model generalization of permutation invariance. This Isomorphism Invariance is a defining constraint on 'logics' in Abstract Model Theory.

3.2 How it works In concrete cases, this notion has useful practical equivalents. E.g., permutation invariance means either of the following for classical set operations:

$$\pi\,[O\,(X, Y, ...)] \quad = \quad O\,(\pi\,[X], \pi\,[Y], ...) \qquad \text{('O commutes with } \pi\text{')}$$
$$x \in O\,(X, Y, ...) \quad \Leftrightarrow \quad \pi(x) \in O\,(\pi\,[X], \pi\,[Y], ...) \qquad \text{(' } \pi \text{ respects O')}$$

Many expressions denote permutation-invariant objects. The account lifts from the standard logical operations in first-order types to other expressions, bringing out unexpected relationships. In particular, many second-order categories in natural language contain invariant items. Useful examples include (a) certain *generalized quantifiers*, such as "ten A are B", "most A are B", taking two predicates (viewed as sets) to an assertion, (b) the *reflexivizer* "self", taking binary verbs R to unary ones $\lambda x \bullet Rxx$. The viewpoint also carries over from natural to programming languages. When the objects are computational states, programs denote binary binary transition relations, and then the standard *program constructs* such as sequential composition, guarded choice, or iteration, are permutation invariant ‘logical constants’.

Let us dispel any air of mystery here. These observations have a simple background in standard logic – since they follow from the above Isomorphism Lemma. Here is why. The Lemma holds for any well-behaved logical language, not just first-order, but also in higher types, or with infinitary expressions. Thus, any definable expression is permutation-invariant. But all examples mentioned so far – and all in the literature that I am aware of – *are* definable in such logical languages. E.g., the above program constructs are operations in Relational Algebra, which have a first-order definition. Likewise, the best accepted readings of linguistic generalized quantifiers like “most” have definitions in higher-order logic which are then invariant by their logical form.

3.3 Classification Even more interestingly, one can also *reverse* the perspective, and classify all invariant items in simple types in some format of definition, as has been done by many authors. Here is a simple illustration (van Benthem 1986):

Fact All invariant set operations must have values, O(X, Y, ..) made up out of exhaustive Boolean combinations of their arguments X, Y, ...

Proof If the value set O (X, Y, ...) properly overlapped any of the 'minimal zones' in the Venn Diagram for any arguments X, Y , one could then permute two objects in such a zone across the boundary (keeping all other objects fixed), leaving all images π [X] equal to X , and yet changing the image set π [O (X, Y, ...)] . Note: We get no *uniform* Boolean definition: the value of O may depend on argument sizes. ■

More complex classifications are found in van Benthem 1989, and 1991, chapter 10. These are the attractions of the approach when one comes to it first. Permutation invariance is simple, it covers all standard logical constants and relates them to others across categories – and complete classifications of invariants may be viewed as interesting ‘expressive completeness theorems’ for certain types of expression.

Permutation invariance has been proposed in many places independently, in full type-theoretic generality by Tarski 1986, Läuchli 1970, van Benthem 1986, and others. Papers are scattered across logic, linguistics, and computer science (cf. Trakhtenbrot 1987). Philosophers have caught up a bit later, again in relative isolation (McCarthy 1981, Sher 1991, and others). Since the ideas are so simple, one good logic textbook should do the job of making this 'universal knowledge' into common knowledge.

3.4 Historical background Dual viewpoints on definability of basic concepts and their invariance under suitable transformations date back to the mid-19th century. In the ground-breaking work by Helmholtz on the perceptual foundations of mathematics, invariants of spatial *movement* (moving in a straight line, turning around) correspond to natural primitives for geometrical theories. For instance, my observing that one point lies in between two others will not change, no matter how I view this constellation across my natural movements in space. Thus, primitive 'betweenness' emerges out of experience, not linguistic convention. Movements are given, their invariants force themselves upon us. Later mathematicians took these ideas further. Klein's 'Erlanger Programm' defines a mathematical structure by some family of transformations over a given base space, which then generate the invariants underlying one’s mathematical theory. A major traditional habitat of invariance analysis in mathematics is the theory of Lie groups.

Transformations and their invariants as the source of our primitives are also quite alive to-day in empirical fields like psychology and computational image processing.

Now think of logical constants as the *most robust invariants* of all, resistant to every permutation. The latter are the roughest transformation, leaving no structure intact except bare distinctions between objects. The dark side of this robustness, of course, is that logical constants have little to no content. Moreover, as with Klein's Erlanger turn, the 'transformations' are now in the mind, chosen by us, rather than by Nature. Again, we see the same move here that we noticed below. Mathematicians sometimes talk about invariants inside one single structure, and then again across structures. This seems no big issue, and the two viewpoints have been around from the start. Either way, this is the appropriate setting for invariance accounts of logicality, as an extreme case in a broader scientific tradition which is still very much alive.

3.5 General logical results Permutation invariance has been around before the 1990s, and some of its basic properties deserve to be known more broadly. These include expressive completeness results, but also some types of result that are less obvious from the above. First, the Isomorphism Lemma implied that any operation defined in a logical formalism that does not refer to specific objects (such as first-order logic with identity, higher-order logic, or finite type theory) will be permutation invariant. And the observation is more general. By the same reasoning,

> Formulas in a language with a non-logical vocabulary L define
> type-theoretic objects that are *invariant for L-isomorphisms.*

Inside one model, these objects are invariant for *L–automorphisms* – i.e. those permutations that leave L-predicates and operations the same. As for converse results, the earliest reference I have found so far is Hermann Weyl's 1930s book "Philosophy of Mathematics and the Natural Sciences" (Weyl 1963), where he observes that all logically definable geometrical relations, starting from basic primitives such as 'betweenness', are invariant under the usual geometric transformations. Weyl then mentions the converse question, *whether all such invariants are logically definable,* and says that this is a difficult question which will probably never be solved.

Here are some well-known facts, showing that at least some things of interest have been discovered since. No general equivalence between L-automorphism invariance and L-definability can hold for finitary languages, for cardinality reasons. The only <-automorphism of IN is the identity – and so there are uncountably many invariant subsets of natural numbers, which outrun the resources of standard logical languages. Things get better on special models, witness the following folklore result.

Proposition *Invariance Equals First-Order Definability on Finite Models*

On finite models, the L-automorphism invariant predicates of individuals are exactly the first-order L-definable ones.

Model theory has a more sophisticated result implying this, viz. *Svenonius' Theorem*:

If a predicate Q is invariant for all L-automorphisms of the models for some first-order theory T, then T implies some finite disjunction of explicit definitions for Q in terms of L.

The Proposition follows by taking T to be the complete first-order theory of the finite model. If one wants to have the Proposition on all models without restrictions, the language must adapt, and become infinitary (McGee 1996):

Theorem *Invariance Equals Infinitary Definability on All Models*

On any model, the L-automorphism invariant predicates are those definable in the infinitary language $L_{\infty\infty}(L, =)$.

There are other ways of approaching this match: cf. the use of categories of Boolean-valued models in Butz & Moerdijk 1998, which does get by with first-order logic. Going in another direction, one can also lift the analysis to more general linguistic categories of expressions using finite type theory (van Benthem 1989, 1991):

Theorem *Invariance for Type-Theoretic Objects*

For all terms τ_b with n free variable occurrences of types $a_1, ..., a_n$, and all permutations π of the individuals, suitably lifted to higher types,

π_b ($[[\tau (u_1, ..., u_n)]]$) = $[[\tau (\pi_{a1}(u_1), ..., \pi_{an}(u_n)]]$.

Theorem *From Invariance to Definability*

Over *finite* base domains, all permutation invariants in their finite type function hierarchy are explicitly definable in finite type theory.

These results are not all there is to say in this area. Here is a classic result:

Theorem (Laüchli, 1970) *Invariants Need Intuitionistic Types*

The purely functional types that contain invariant items are just those whose implicational forms are provable in *intuitionistic logic*.

The 'truth value' type t is read as "true" or "provable" in this setting. Thus, this explains why there are permutation invariants taking objects to predicates (e.g., the 'Quiner' $\lambda x \bullet \lambda y \bullet y{=}x$ lives in the functional type $e \to (e \to t))$), while there are no invariant 'choice functions' in the type $(e \to t) \to e$: the latter is not a valid formula.

Here is another type of result. Usually, one allows permutations of entities, while keeping the truth values fixed. As a result every object in the pure t–type hierarchy over the base set $\{0, 1\}$ counts as 'permutation invariant'. Now we know that the operations of first-order type here are all definable by standard Booleans $\neg$, &, $\vee$. Here is an expressive completeness result for the full hierarchy (van Benthem 1991), extending the usual reasoning in propositional logic to higher types like $(t \to t) \to t$:

<u>Theorem</u> All objects in the finite t–type hierarchy are definable in the t–typed *lambda calculus* with just Boolean $\neg$, &, $\vee$.

Thus, at least for 'truth table objects', our usual languages do just the job required. Finally, here is yet another type of question, which has remained open so far:

Find a counting formula *for the number of invariant items in a finite type.*

Though most of these results are elementary qua proofs, they do show that there is some technical substance to the notion of invariance, backing up its intuitive appeal.

3.6 Cross-model versions Next, we consider some 'cross-model results' related to logicality. This time, our purpose is to show that some well-known model-theoretic results normally not related to invariance are highly related to the above intra-model versions. After that, we discuss the significance of this perspective. Again, we follow up on the Isomorphism Lemma with a simple folklore result.

<u>Proposition</u> For finite models **M**, **N** with tuples of objects **d**, **e** , TFAE:
(a) there is an L–isomorphism between **M**, **N** sending **d** to **e**
(b) **d** , **e** satisfy the same first-order formulas in the vocabulary L .

This result is often interpreted as a constraint on language design. A good language matching a structural similarity relation should be invariant, but it should also detect any pair of 'dissimilar' models by providing some specific definable difference. Thus, expressive power of a language can be measured, either by its being able to *define enough invariants* inside models (our focus in Section 3.4), or by *providing enough distinctions* between models. But the technical connection at this level is not tight yet.

Remark Specialized to single models, the Proposition does not claim directly that all automorphism invariants in a model are definable. It only says that, if a predicate P is invariant, each tuple **e** outside of it will disagree on at least one first-order formula with each tuple **d** inside P. In finite models, these formulas can be 'patched' to provide a first-order definition for P : but in general we need infinitary combinations.

Remark As a constraint on language design, logicality becomes a 'holistic' notion concerning a whole language, rather than on its separate expressions. This shift would be in line with folklore views among logicians holding that truly 'logical' languages should satisfy desirable 'system properties', such as an *interpolation theorem*. We will not pursue this alternative holistic take on logicality here.

But then, something much stronger holds. Here is a generalization to arbitrary models and the infinitary logic $L_{\infty\omega}$ of *Scott's Theorem*, which says that each countable model has one formula in countably infinitary logic defining its isomorphism class.

Theorem (Barwise 1975) For every model **M** and tuple of objects **d** there is a formula $\phi^{\mathbf{M}}(\mathbf{d})$ of the infinitary language $L_{\infty\omega}$ such that TFAE:
(a) $\mathbf{N}, \mathbf{e} \models \phi^{\mathbf{M}}(\mathbf{d})$ (b) **N**, **e** is potentially isomorphic to **M**, **d** .

Potential isomorphisms are non-empty families of finite partial isomorphisms which satisfy the usual back-and-forth extension properties. This notion is the set-theoretic 'absolute' analogue of isomorphism. It is easy to derive corresponding intra-model versions for this theorem in terms of invariance for 'potential automorphisms'.

At another extreme of syntactic resources, these results are related to another well-known topic in elementary logic, which is not usually considered as doing the job of analyzing 'logicality'. Consider the usual analysis of first-order logic in terms of *Ehrenfeucht-Fraïssé games*. At least with finite relational vocabularies, we can define models using single formulas up to Duplicator's finite winning levels. In a sense then, the usual Adequacy Theorem is also an invariance characterization of first-order logic!

Theorem For any two models **M**, **N** with object tuples **d**, **e** TFAE
(a) **M**, **d**, **N**, **e** agree on all first-order formulas up to quantifier depth k
(b) Duplicator has a winning strategy in the k-round comparison game between the two models **M**, **N** starting from the match **d** – **e** .

This is a combination of two intuitive requirements: (a) invariance plus (b) finiteness. It might be worth the philosophers' while thinking about the plausibility of this.

Finally, invariance for cross-model relations is also used in analyzing other languages. E.g. in the *lambda calculus*, Plotkin has partially captured the set of lambda-definable items via invariance for ‘logical relations’ that may also identify individuals.

3.7 Digression: invariance and definability in science Logical theory does not *quite* fit invariance thinking in the sciences. Physicists and mathematicians work with ‘symmetries’, invariances inside one model, or ‘transformations’ which may also work across models. Consider the Galilei or Lorentz transformations in physics. These coordinate mappings from, say, IR^4 to itself preserve relevant 'physical assertions', including fundamental laws. But the precise languages here are never specified. Indeed, important results often have a different flavor. E.g., consider the celebrated theorem that "Causality implies the Lorentz Group" (Zeeman 1964) – a phrase which a well-bred logician might have trouble parsing… In Minkowski Space, the Lorentz transformations preserve ‘causal precedence’: the relation which holds between a point and all points in its future light cone. Zeeman showed that conversely, each automorphism w.r.t. causal precedence is a composition of Lorentz transformations with a few minor domestic ones. Notice that there is no explicit role of any language here. Even so, the result is reminiscent of Robb's logical work on special relativity, showing that the second-order theory of causal precedence suffices for defining the full metric structure of Minkowski space. Here is another interesting gap to be bridged. Consider Beth's Theorem, a classic on logical definability. Its transition from implicit definability (an invariance property!) to explicit definability seems directly applicable to elementary geometrical reasoning. We often 'see' that some geometrical object is determined by others. E.g., the angles of a triangle are fixed once we have the sides. A geometrical rule of thumb then says one should be able to explicitly ‘define’ the former in terms of the latter, as is indeed the case for angles and sides. But Beth's Theorem only helps us here when the problem can be stated in *first-order* terms (a constraint quite alien to working scientists), and the ‘determination’ holds in *all* models of the geometrical theory, not just 'ordinary space'… It would be of practical interest to analyze more of such concrete examples, confronting logical theory with invariance practice.

4 Critique

Let us now look at the merits of permutation invariance as an account of logicality. In this section, I will state what I dislike and what I like about it. Also, I will briefly discuss Sol’s verdict in Feferman 1999.

4.1 Bad points First, there is a problem of 'size': *too many items* for comfort *are permutation-invariant*. For instance, in the type of monadic generalized quantifiers (denoting properties of sets), already infinitely many items $\lambda X \bullet Q(X)$ are definable in higher order-logic. Related to this is a second problem. Permutation-invariance is local, and *allows wild variations* across models: say, a quantifier meaning "some" on domains of even size and "all" on domains of odd size. Isomorphism invariance does work across models, but still allows for these variations across different cardinalities. Third, the over-inclusiveness also involves the following. *Arbitrary* infinite set conjunctions and disjunctions, no matter how complex and non-constructive, of invariant objects are themselves invariant. Now these are all 'book-keeping objections': the set of items that passes is 'not quite to our taste'. But there are further problems. To begin with, permutation invariance ignores *further semantic aspects* of logicality, such as intuitive features of 'uniformity' or 'finiteness'. But my main objection is what follows.

Permutation invariance is *blatantly circular* as an account of logicality! And it shares this feature with any transformation/invariant account of a set of notions. For, to get going, we must make a number of prior decisions that manipulate the final outcome:

Which objects are taken? And then, which transformations?

E.g., in geometry, affine transformations are supposed to 'justify' "betweenness" as a geometrical primitive. But betweenness is needed to define these transformations in the first place, when all is said and done. The same holds for logicality. Do we want identity to come out as a logical notion? Then look at permutations, whose definition presupposes the notion of identity (surjective functions respecting non-identities). The case is even more blatant with Boolean truth value operations. These come out as invariant for the totally trivial reason that type-theoretic permutations leave the truth values the same. So, they do not effect truth value operations at all. Evidently, this is no 'explanation', and permutation invariance is not even a 'test' that the Boolean operations pass. Examples of this are easily multiplied at many places in Section 3.

The prior choice of objects is even more insidious. For instance, geometrical invariance works on extensionless 'points', and therefore only derives primitive relations in dependence on some prior notion of basic object. But we might want to say, for instance, that 'betweenness' is a primitive notion in spatial reasoning even without making any commitment as to the underlying objects: extensionless points, primitive extended regions, etc. More generally, should truly logical notions not be *independent* from particular choices of objects over which they are supposed to work? Here is a linguistic example. Natural languages tend to have *two* quantifier systems. One system works on ‘count nouns’ (“every girl left”, “many boys left”), the other on so-called ‘mass nouns’ (“all wine was spilt”, “much wine was spilled”). The first requires what linguists call countable objects, the second so-called measurable ones. Now, the ‘same’ logical quantifiers seem involved over both these domains. But how to get this out of invariance for transformations? And though a unification may be feasible in this particular case, we will still be left with the worry that we are not really done. We may have to redo the exercise once we find yet other sorts of object. Thus,

permutation invariance cannot serve as a foundational account of logicality.

As an independent story of logicality, it is about as circular as – to mention one blatant hoax – the ‘justification’ of first-order axioms by informal soundness arguments which presuppose all the axioms they are justifying (plus usually a whole lot more).

There, we have said it! But please realize that this is only a knock-down objection if one had fundamentalist designs in the first place. I myself think that there cannot be absolute foundations, neither for logical laws nor for logical notions. Therefore the circularity does not bother me at all, and my story will continue.

4.2 Feferman’s proposal Here are some points from the discussion in Feferman 1999. Its main critical points concerning permutation invariance are its (a) ‘making logic dependent on set theory’, (b) ‘set-theoretic non-absoluteness’ and (c) ‘non-uniformity across domains’. The former two are of a mathematical nature, more tangential to my main concerns here. But as for (c), he proposes a stronger version of isomorphism invariance, which is of immediate interest. It specializes a notion from the Lambda Calculus (Plotkin 1980), viz. *invariance for logical relations*:

Not just bijections or isomorphisms, but *any* binary relation R between individuals in two models $\mathbf{M}$, $\mathbf{N}$ can be lifted canonically to a relation between objects in all types:

$$f\ R_{a\bullet b}\ g \quad \text{iff} \quad \mathrm{proj}^L(f)\ R_a\ \mathrm{proj}^L(g)\ \textit{ and }\ \mathrm{proj}^R(f)\ R_b\ \mathrm{proj}^R(g)$$
$$f\ R_{a\to b}\ g \quad \text{iff} \quad \forall x \in D_a^{\mathbf{M}}\ \forall y \in D_a^{\mathbf{N}},\ x\ R_a\ y \text{ only if } f(x)\ R_b\ g(y)$$

Now, Plotkin observed that all lambda-terms F denote definable functions $F^{\mathbf{M}}$, $F^{\mathbf{N}}$ which are *relation-invariant* in the following sense:

if we feed R-related arguments, then their F-values are also R-related.

An object F in one model is called relation-invariant if F R F for all relations F . Conversely, Plotkin proved that all relation-invariant objects are lambda-definable, at least *up to second-order types*. The general situation is discussed in Statman 1982. Variations on this definition have been proposed in van Benthem 1991, chapter 12, which lifts Boolean monotonic functions and modal bisimulations in the same style, obtaining richer classes of invariant items. Now here is what Feferman does. He takes the base relations to be 'homomorphisms', viz. relations x R y of the special form

> y = f(x) for some surjective function f from individuals in **M** to those in **N**

This will make the standard operations of negation (N, for short), conjunction (C) and existential quantification (E) relation-invariant. Then he proves what amounts to an invariance characterization of first-order logic:

> (a) all items lambda-definable from N, C, E are homomorphism-invariant
> (b) all homomorphism-invariant *monadic quantifiers* are lambda-definable from N, C, E (in fact, they are definable in monadic first-order logic)

For an assessment of what this achieves, a little technical detail is needed. Here is a typical case. Consider a monadic quantifier taking sets to truth values. Sol's criterion amounts to saying that the same truth value will be assigned in any 'contraction' to a single argument set. It follows immediately that the invariant items, across the board, are determined by two truth-value assignments: to the empty set, and a non-empty set. Here is a more general perspective on what is going on (not mentioned in Sol's paper), which shows its relation to the above discussion in Sections 3.2, 3.3, 3.6. For operators of the type $F(x_1, .., x_k, A_1, .., A_m)$ with object and predicate arguments, the stated invariance criterion works out very simply to a much more familiar one:

> Let f be any surjective strong homomorphism between models **M**, **N**
> (i.e., this function respects all predicates both ways, except for identity).
> Then $F^{\mathbf{M}}(d_1, .., d_k, A_1, .., A_m)$ iff $F^{\mathbf{N}}(f(d_1), .., f(d_k), f[A_1], .., f[A_m])$

Thus, we restrict attention to objects that are *invariant for strong homomorphisms,* not just for isomorphisms. As we know from model theory, this holds for all notions definable in any logical language with the shown parameters *without using identity*. This fact explains why the above monadic definitions will do (the lambdas are not crucial), and it also explains why *well-foundedness* will be 'logical' on this account (as discussed in Sol's paper). In this light, the monadic case is rather special, as we can classify all invariant items. But the same huge variety as before will return with operators of a binary relational type $F(R)$. For, there are many equivalence classes for binary relations under strong homomorphism: e.g., all ordinals fall into different ones. And on each of these, the criterion allows us a different identity-free definition.

I conclude that Sol's account suffers from the same over-generation that permutation invariance has. And this is to be expected. Any reasonable invariance analysis must postulate some relevant equivalence relation on models. On each of its invariance classes, we are free to choose our denotation. Now the relation may be so coarse that only finitely many equivalence classes remain, and no explosion of variety occurs among denotations for 'logical constants'. But then we have given away the game in its definition. Or, like with most interesting semantic equivalence relations, we have infinitely many equivalence classes, and then the variety of denotations will explode.

Another critical point is related. Given the equivalence form of the definition of invariance under transformations/model relations, the following is bound to happen:

all infinitary Boolean combinations of invariants are themselves invariant.

This did not matter in the special monadic case, but it will be a side-effect for whatever more complex types we try to analyze. Thus, infinitary Booleans take a 'free ride' on such analyses – and cannot really be considered as having been 'explained'. Finally, in the earlier conceptual terms, Sol's proposal is as 'circular' as any. The choice of the 'homomorphisms' as the relevant equivalence relation invests precisely the amount of 'indifference' needed to engineer the landing in monadic first-order logic.

I conclude that Sol's account has its merits, but changes nothing in the general limitations of invariance approaches. But to me, this is not a bug, but a feature!

4.3 Good points As for permutation invariance in general, all the objections so far show that it is not a good *foundational* account of logicality. But then, I do not consider this a great tragedy. First, invariance accounts have the virtue of tying in logicality with ways of thinking in other sciences whose utility is beyond doubt – which is a benefit in itself. Next, even circular accounts of notions can be useful, when they couple 'free-floating' notions in *enlightening* ways to others. (As David Lewis once remarked about another 'logical explication', a fixed *connection* between two weak swaying reeds may be a hard fact.) Invariance can even help pinpoint useful connections. Suppose we find some 'non-logical' item F failing the permutation invariance test. The interesting thing is not this negative information by itself. Usually, analyzing the counter-example will point to 'hidden variables': additional structure that needs to be preserved by a transformation to get the expression as an invariant. For instance, *prepositions* ("in", "behind", "along") are ubiquitous linguistic expressions that are not quite 'logical', but not quite 'non-logical' either. Arbitrary permutations of the objects in space do not preserve the relations denoted by prepositions, but this very failure suggests a better-tuned class of automorphisms respecting the right information, e.g., suitable geometrical transformations of visual scenes (cf. Winter & Zwarts 1997). In other words, the invariance approach comes with a hierarchy of levels, and hence it is 'parametrized' in the sense we were looking for in Section 1.

5 A procedural viewpoint

One attraction of the invariance approach are the analogies it offers with other areas where it is used. In this section, we explore one such analogy, which leads us to a more computational or *dynamic* view of 'logical constants'.

5.1 Bisimulation and modal process invariants In computer science, notions of similarity between computation spaces (e.g., labeled transition systems) determine a notion of process. For instance, taking *bisimulation* between labeled transition systems for one's process equivalence, one gets the usual Process Algebra.

Definition A *bisimulation* is a binary relation $\equiv$ between states in models **M**, **N** satisfying these conditions: (1) Atomic Harmony If $x \equiv y$, then **M**, $x \models p$ iff **N**, $y \models p$, for all proposition letters p. (2) Action Zigzag (2.1) If $x \equiv y$ and $xR_a z$, then there exists u such that $yR_a u$ and $z \equiv u$, and (2.2) If $x \equiv y$ and $yR_a u$, then there is a z such that $xR_a z$ and $z \equiv u$. For details, cf. van Benthem 1996, Chapters 3–5. ■

Invariant items are then the properties P of processes mimicked by the simulation:

whenever $x \equiv y$, then **M**, $x \models P$ iff **N**, $y \models P$

These invariant properties can often be classified in some appropriate language. E.g., with bisimulation, the invariants are the items expressible in (infinitary) *modal logic*. In this setting, the role of 'good' process constructions is this:

they are 'safe,' in the sense of being automatically simulated.

More precisely, consider an operation $F(R, S, \ldots)$ sending binary transition relations to a new transition relation. Examples are the usual program constructions of composition, guarded choice, or iteration. Assume we have any two models with a bisimulation $\equiv$ satisfying the back-and-forth clauses for the arguments R, S, …

> F is *safe* if this same $\equiv$ automatically satisfies the back-and-forth clauses for the transition relation which is its value F (R, S, …) .

Examples of safe operations are *relational composition* **;** , *union* $\cup$ (finite or infinite), as well as *strong negation* **~** (being the test that the relation has no successor). Typically non-safe are relational *intersection* and Boolean *complement*. Here is an expressive completeness result from van Benthem 1996, Chapter 5:

Theorem The safe first-order definable relational operations are precisely those defined by means of {**;** , $\cup$, ~} .

Note that these operations are a kind of dynamic counterparts to the usual Booleans.

5.2 Logical constants and simulation Safety is also implicit for logical operations in the inductive proof for the Isomorphism Lemma between two models **M**, **N** . This proof shows how *evaluating* a formula (a process taking place inside one model) can be *simulated* step by step by passing to another model. This, in effect, is what is achieved by familiar subroutines like:

$$\mathbf{M}, \mathbf{d} \models \neg\phi \quad \text{iff} \quad \text{not } \mathbf{M}, \mathbf{d} \models \phi \quad \text{iff} \quad \text{not } \mathbf{N}, f(\mathbf{d}) \models \phi \quad \text{iff} \quad \mathbf{N}, f(\mathbf{d}) \models \neg\phi$$

Also, when evaluating $\mathbf{M}, \mathbf{d} \models \exists x\phi$, two routes lead to the same match between object tuples. We can take a witness d for x in **M** , and then pass from $\mathbf{M}, d, \mathbf{d} \models \phi$ to $\mathbf{N}, f(d), f(\mathbf{d}) \models \phi$. Or we first pass to the other side, observing that $\mathbf{N}, f(\mathbf{d}) \models \exists x\phi$, and then find the witness in **N** . This yields exactly the kind of commutative diagram that we also had with bisimulation. Thus, in simple arguments like this, logical constants appear as natural moves in evaluation processes. In a slogan

> *logical constants are* process glue*!*

Let us apply these ideas to first-order logic. For a process equivalence, take *potential isomorphism* between models **M**, **N** , used in the theory of the infinitary logic $L_{\infty\omega}$. This is a bisimulation between transition systems whose ‘states’ are maps from finite sets of variables to objects in the models. The relevant transitions are of two kinds:

testing a fact	(while staying in the same state)
picking an object	(resetting the value of some existing variable, or assigning a value to some fresh variable)

Clearly, these are the only primitive actions in evaluation of first-order formulas, viewed dynamically. All the rest is indeed 'process glue'. Correspondingly, matched states in a potential isomorphism satisfy the same atomic statements, plus the back-and-forth condition of bisimulation with respect to object-picking moves.

Now, to define invariance as simulation 'safety', we need to view first-order formulas as evaluation processes. But what are the corresponding transition relations? Here we take a leaf from 'dynamic predicate logic' (Groenendijk & Stokhof 1991) which fits very closely with potential isomorphisms as bisimulations (cf. also Fernando 1994):

Definition Update Conditions for Dynamic Predicate Logic

Atoms are tests

s_1 $[[Pt]]^{\mathbf{M}}$ s_2 iff $s_1 = s_2$ *and* $\mathbf{M}, s_1 \models Pt$

Conjunctions are relational compositions

s_1 $[[\phi\&\psi]]^{\mathbf{M}}$ s_2 iff *for some* s_3, s_1 $[[\phi]]^{\mathbf{M}}$ s_3 *and* s_3 $[[\psi]]^{\mathbf{M}}$ s_2

Existential quantification are random assignments

s_1 $[[\exists x\phi]]^{\mathbf{M}}$ s_2 iff *for some* s_3, $s_1 =_x s_3$ *and* s_3 $[[\psi]]^{\mathbf{M}}$ s_2

Negations are strong failure tests

s_1 $[[\sim\phi]]^{\mathbf{M}}$ s_2 iff $s_1 = s_2$ *and for no* s_3, s_1 $[[\phi]]^{\mathbf{M}}$ s_3 ■

By the earlier modal analysis, clearly safe operations in this setting are (a) dynamic conjunction-as-composition, and (b) dynamic negation as impossibility-test. Less clear is the role of existential quantification. The proper way of viewing its clause is as (c) a sequential composition of two actions: one of object picking for the prefix $\exists x$, and another of processing the remaining assertion ϕ. Thus, on this dynamic analysis,

an existential quantifier is not a logical constant!

It is rather an atomic semantic action, which needs to be 'glued' to what follows by a 'hidden logical constant', viz. composition. Thus, replacing invariance by safety as an account of logicality has some surprising aspects, which may look 'non-standard'. On the other hand, the reader may also find them innovative.

5.3 Discussion Saftey effects a drastic reduction in the space of 'logical items'. It is very easy to see that safe operations must also be isomorphism-invariant, but the converse holds by no means. But this ‘counting’ is not our main concern. Neither is the circularity, which will rear its head once more, as we only get going by settling on some prior notion of ‘process equivalence’. In this sense, safety does not change anything fundamental from earlier sections. More interesting, however, is the question whether the above dynamic account has some intrinsic plausibility as a view of logicality. To pursue this, one would need to see whether it generalizes well. A good test case would be simulations for languages with generalized quantifiers.

But our main concern would be another one. Like permutation invariance, safety does not really provide an account of the standard Boolean operations! One reason is that the above transition relations are rather coarse. They only record input-output states of succesful verifications, without recording the fine-structure of evaluation processes ‘along the way’. But if we look at the latter, Boolean operations do play a crucial role in the ‘control’ of where to go. This observation illustrates demonstrates a general point, and at the same time, a further ‘parameter’ for the present analysis. Not just process equivalences over the same class of structures are a choice point for our invariance analysis. Process representations *themselves* may come at various levels of detail. And the level we pick will determine what ‘invariant’ or ‘safe’ items we can discern, and hence, what sort of ‘logical items’ we will encounter. This point is related to our discussion in Section 4.1, and we think this open-endedness is inevitable. More concretely, we do think the step-by-step fine structure of evaluation is of logical relevance, and we expect to find a more genuine analysis of the Booleans at that level. But at the moment, we have nothing more than a vague sense that this requires an analysis in terms of *evaluation games* (van Benthem 2000), of which the above transitions are just the ‘externally visible’ traces. Some logical constants only emerge at this finer level, as the crucial operations regulating control for the players involved.

The proposals in this section are not the final word. But we hope to have shown at least how invariance thinking can also offer genuinely new perspectives on logicality.

6 Further semantic aspects

Does logicality have intuitive semantic aspects different from invariance? There is some evidence for this in the linguistic literature on *Generalized Quantifier Theory* (Keenan & Westerståhl 1997). We review some relevant themes, using notions and results from van Benthem 1986 to show concretely how one can broaden the analysis of logicality.

6.1 Determiners and quantifiers This is the study of linguistic *determiners*. These are expressions like "all", "most", "three", "enough", that take common nouns and verb phrases to form sentences according to the basic pattern

Det N VP

Across languages, determiners satisfy some universal semantic constraints, such as *Conservativity*, restricting the statement made to objects satisfying the first argument:

$$\text{Det } AB \;\Leftrightarrow\; \text{Det } A\,(B\cap A)$$

Next, imposing permutation invariance for special determiners of 'quantification' allows us to represent these 'quantifiers' Q *numerically*, as the set of all couples

(a, b) for which there exists some pair of sets A, B
with Q AB such that $|A–B| = a$, $|A\cap B| = b$

This representation has been used extensively for semantic analysis of quantifier classes, via the so-called 'Tree of Numbers'.

6.2 Smoothness and monotonicity Quantifiers in natural languages are not just permutation invariant. A semantic feature with much extra bite is their

persistence across related situations

which can be viewed either as a semantic property of *smoothness*, or as an inferential one of *preservation*. Thus, all basic quantifiers are monotone, either upward or downward, in both their arguments. E.g. "all" satisfies the two patterns:

$$\frac{Q\,AB \qquad B\subseteq B'}{Q\,AB'} \qquad\qquad \frac{Q\,AB \qquad A'\subseteq A}{Q\,A'B}$$

We can use this property to characterize the four classical quantifiers "all", "some", "no", "not all". For this we need one extra semantic condition:

Variety If A is non-empty, there exist B, B' with Q AB, ¬Q AB' .

Fact The four quantifiers in the Square of Opposition are the only determiners that satisfy Conservativity, Double Monotonicity, and Variety.

<u>Proof</u> Let $Q\,XY$ be any determiner satisfying these three conditions. We do one case, the others are similar. Suppose Q has monotonicity type 'left-down, right-up'.

<u>Claim</u> Q is "all", i.e., $Q\,XY$ iff $X \subseteq Y$

First, by Variety, we have $Q\,AB$ for some A, B. Hence we have $Q\,\varnothing B$ ($Q\,AB$ plus left downward monotonicity) and hence $Q\,\varnothing\varnothing$ (conservativity) and so $Q\,\varnothing Y$ (right upward monotonicity). <u>From right to left</u>. (a) If X is empty, then we have just shown $Q\,\varnothing Y$. (b) If X is non-empty, then we have $Q\,XA$ (variety), $Q\,X(A\cap X)$ (conservativity), and so $Q\,XY$ (by right upward monotonicity). <u>From left to right</u>. Suppose $Q\,XY$, but not $X \subseteq Y$. By left downward monotonicity, $Q\,(X–Y)Y$, and by conservativity, $Q\,(X–Y)\varnothing$. By right upward monotonicity then $Q\,(X–Y)Z$ for all sets Z, which contradicts variety for the non-empty set $X–Y$. ■

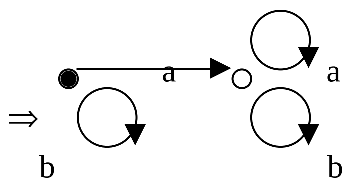
Again, monotonicity is a semantic 'parameter' which expressions can have in degrees. E.g., the non-standard quantifier "most" is still right upward monotone, but it lacks monotonicity either way for its left argument. Without Variety, more quantifiers make the grade, including "at least three" or "at most five". Here is a more complicated expressive completeness result (van Benthem 1986).

<u>Theorem</u> The 'doubly monotone' quantifiers on finite universes are the four classical ones plus all numerical variants "at least n", "at most n", etc.

A nice proof is by geometrical inspection of the number representation for quantifiers. Right upward monotonicity closes the set Q of ordered pairs (a, b) under rightward steps $(a+1, b) \Rightarrow (a, b+1)$, left upward monotonicity makes Q closed under steps $(a, b) \Rightarrow (a+1, b)$ and $(a, b+1)$ that 'shift levels'. The geometrical shapes for quantifiers Q allowed by these conditions are easily enumerated. Higher-order quantifiers such as "most" have a more complex, but still regular 'number geometry'.

Monotonicity is a ubiquitous phenomenon in natural reasoning and semantics. The basic lexical items in the determiner category are monotone across all languages. Surely, this is also a key feature of 'logical' constants: stable description of situations, and supporting corresponding inferences.

6.3 Simple computability and semantic automata Another attractive feature of basic logical items is their *simple uniform computability*. For the basic quantifiers, this involves very simple procedures: so-called *semantic automata*. E.g., to compute whether "all AB" we just need a finite automaton with two states. This will survey the relevant set of objects A (which suffices by conservativity) in some order, reading in symbols a for each object in A–B , and b for each object in A∩B :



We start in a recognizing state, and stay there as long as we read symbols b . As soon as a symbol a is encountered, we move to a second, rejecting state and stay there, whatever is read afterwards. The general result here is this:

<u>Theorem</u> The first-order definable quantifiers are precisely those computed by *acyclic finite automata*.

Again we have a 'parameter' now, as we can vary the type of automaton to get further levels of semantic computability. Arbitrary finite automata add the power of counting (say, "an odd number of"). The next step would be adding memory storage. Indeed, the semantic test procedure for the quantifier like "most" needs the unbounded finite memory of a *push-down store automaton.* Push-down storage seems the most complex computational procedure needed for natural language quantifiers. Van Benthem 1986 characterizes the corresponding numerical content as follows:

<u>Theorem</u> The quantifiers computable by push-down automata are precisely those whose numerical conditions on the characteristic numbers a = |A–B| , b = |A∩B| are definable in purely additive 'Pressburger Arithmetic'.

Pressburger Arithmetic is decidable, and hence many questions about natural language quantifiers will also be. Thus, the usual Automata Hierarchy now provides fine-structure levels for logical constants, via their computational requirements.

6.4 What this means Generalized Quantifier Theory shows how we can zoom in on the standard quantifiers, amidst all permutation-invariant ones, through various natural additional semantic and computational properties. Now this harmony may be a lucky coincidence between a bunch of different notions. This suggests that the core group of 'logical constants' is best analyzed as an intersection of different natural kinds of expression: they are clever, attractive, and good cooks into the bargain…

7 Comparing stances

We have analyzed logicality in a semantic perspective here. But relations to other logical stances have occurred occasionally: computational (Sections 5, 6.3), game-theoretic (Section 5), and inferential (Section 6.2). There are certainly independent intuitions along the latter lines that should be explored. In particular, a sustained analysis of the proof-theoretic stance on logicality might be very illuminating, throwing also additional light on semantic invariance analyses. For instance,

why are the core invariant items so rich in valid inferences?

I know of no good reason; but one would want to understand the phenomenon. Likewise, I would want to couple the semantic and algorithmic aspects more closely. But to avoid a misunderstanding, on my philosophy, no 'completeness theorem' is needed stating that all these approaches 'produce the same yield'. Indeed, even in the analysis of logical consequence, completeness seems often somewhat engineered. And in any case, it does not show that the logical stances 'reduce' to each other.

Of more interest to me would be *combinations* of perspectives. Intuitively, reasoning seems to involve two kinds of step. Standard inferences are 'local'. We investigate some situation, observe A, and know for general reasons that B 'follows' in this same situation. But often we want to use A to derive a statement about some *other* situation, perhaps even couched in a different language, provided it is suitably related. The latter step is closer in spirit to invariance, which allows us to cross models. The *two activities together* seem basic to how we reason in practice. To account for this theoretically, a notion of 'entailment along a model relation' was proposed in Barwise & van Benthem 1999 , allowing inference from one model to its submodels, bisimilar models, homomorphic images, and whatever jump seems relevant.

Likewise, this combined view makes us think about designing inference systems and languages *at the same time*, and this joint task may put its own constraints on 'logicality'. One wants the right 'balance' between the vocabulary of a proof system, and its expressive power on corresponding models. E.g., *intuitionistic logic*, ignored in the classical setting of this paper, makes stronger demands on logical consequence, demanding *constructive* provability. What are its natural 'invariants'? (Bisimulations between Kripke models have been suggested in this connection.) Is there any reason to assume that the classical 'logical constants' are the right language from a constructivist point of view? Similar issues arise with *non-monotonic logics* in AI. Their designers question the laws of classical logic, replacing valid consequence by new schemes like "the conclusion only has to be true in all *most-preferred* models of the premises". But they do stick to the language in which the old dogmas were formulated. True radicals would seek a balance, designing a set of logical constants for non-monotonic reasoning that reflects the new preference structure up to some suitable invariance.

8 Conclusions

Our discussion of logicality has not produced a demarkation of logic. This is good. The essence of 'logic' does not lie in canonical lists of 'logical laws' or 'logical expressions' anyway, but in its general properties and methods. What we did present were the basic facts about the semantic invariance aspect of logicality. This account is circular from a foundational point of view, but suggestive and useful all the same. Moreover, it still has the potential of suggesting new views of logical constants, witness our dynamic analysis of logicality as safety for semantic process operations. But there are certainly other aspects to logicality, coming from different stances: proof-theoretic, or game-theoretic, which eventually deserve an equal opportunity.

Ah, and yes, the basic invariance folklore definitely belongs in logic textbooks!

References

J. Barwise, 1975, *Admissible Sets and Structures*, Springer, Berlin.

J. Barwise & J. van Benthem, 1999, 'Interpolation, Preservation, and Pebble Games', *Journal of Symbolic Logic* 64:2, 881–903.

J. van Benthem, 1986, *Essays in Logical Semantics*, Reidel, Dordrecht.

J. van Benthem, 1989, 'Logical Constants across Varying Types', *Notre Dame Journal of Formal Logic* 30:3, 315-342.

J. van Benthem, 1991, *Language in Action*, North-Holland, Amsterdam (MIT Press, Boston 1995).

J. van Benthem, 1996, *Exploring Logical Dynamics*, CSLI Publications, Stanford.

J. van Benthem, 2000, *Logic and Games*, Report ILLC-X-2000-03, University of Amsterdam.

C. Butz & I. Moerdijk, 1998, First-orderness as Boolean-valued permutation invariance, *Journal of Symbolic Logic.*

J. Etchemendy, 1990, *The Concept of Logical Consequence*, Cambridge, Harvard University Press.

S. Feferman, 1999, 'Logic, Logics, Logicism', Memorial Symposium for George Boolos, Boston (Mass.).

T. Fernando, 1994, ‘Bisimulations and Predicate Logic’, *Journal of Symbolic Logic* 59, 924--944.

J. Groenendijk & M. Stokhof, 1991, 'Dynamic Predicate Logic', *Linguistics and Philosophy*14, 39-100.

I. Hacking, 1979, 'What is Logic?', *Journal of Philosophy* 76, 285-319.

J. Hintikka, 1973, *Logic, Language Games and Information,* Clarendon, Oxford.

E. Keenan & D. Westerståhl, 1997, 'Quantifiers', a chapter in J. van Benthem & A. ter Meulen, eds., *Handbook of Logic and Language*, Elsevier, Amsterdam.

A. Läuchli, 1970, 'An Abstract Notion of Realizability for which the Predicate Calculus is Complete', in J. Myhill, A. Kino & A. Vesley, eds., *Intuitionism and Proof Theory,* North-Holland, Amsterdam, 227-234.

T. McCarthy, 1981, 'The Idea of a Logical Constant', *Journal of Philosophy* 78, 499-523.

V. McGee 1996, ‘Logical Operations’, *Journal of Philosophical Logic* 25, 567–580.

G. Plotkin, 1980, 'Lambda Definability in the Full Type Hierarchy', in J. Seldin & J. Hindley, eds., *To H.B. Curry. Essays on Combinatory Logic, Lambda Calculus and Formalism*, 363-373.

G. Sher, 1991, *The Bounds of Logic,* MIT Press, Cambridge (Mass.).

R. Statman, 1982, ‘Completeness, Invariance, and λ–Definabilty’, *Journal of Symbolic Logic* 47, 17–26.

A. Tarski, 1986, 'What are Logical Notions?', In J. Corcoran, ed., *History and Philosophy of Logic* 7, 143-154.

B. Trakhtenbrot, 1987, ‘On ‘Logical Relations’ in Program Semantics’, in D. Skordev, ed., *Mathematical Logic and its Applications*, Plenum Press, New York, 213–229.

H. Weyl, 1963, *Philosophy of Mathematics and Natural Science,* Atheneum, New York, reprint from 1930s.

Y. Winter & J. Zwarts, 1997, 'A Semantic Characterization of Locative Prepositional Phrases', *Proceedings of Semantics and Linguistic Theory* (SALT 97).

E.C. Zeeman, 1964, 'Causality implies the Lorentz Group', *Journal of Mathematical Physics* 5, 490-493.

J.Zucker, 1978, 'The Adequacy Problem for Classical Logic', *Journal of Philosophical Logic* 7, 517-535.

J. Zucker & R. Tragesser, 1978, 'The Adequacy Problem for Inferential Logic, *Journal of Philosophical Logic* 7, 501-516.