

Adapting Logics

Andreas Blass

University of Michigan
Ann Arbor, MI 48109

and

Microsoft Research
Redmond, WA 98052

ablass@umich.edu

Prehistory

At first, just logic, not logics. Cover all reasoning.

Boole: “An Investigation of the Laws of Thought”

Peano’s axioms included: Num ε Cls.

Hilbert and Ackermann (1928) refer to first-order logic as “der engere Funktionenkalkül”.

New edition in 1959 changed it to “der engere Prädikatenkalkül”.

First-Order Logic

Restricting to first order is essential technically for Löwenheim's early version of the Löwenheim-Skolem theorem.

It is essential conceptually for Skolem's clarification of Zermelo's axioms for set theory. Zermelo's "definite properties" are to be identified with first-order definable ones.

This identification presupposes Dedekind's discovery, extended by Zermelo, that inductive definitions can be replaced by explicit definitions within set theory.

Inductive Definitions

First-order logic is inadequate for database queries. Need transitive closure and other inductively defined queries.

Assume either that X occurs only positively in $\varphi(X, x)$ or that $x \in X \rightarrow \varphi(X, x)$. Define

$$\Phi^0 = \emptyset, \quad \Phi^{n+1} = \{a : \varphi(\Phi^n, a)\}.$$

In infinite case, let n range over ordinals and adjoin for limit λ

$$\Phi^\lambda = \bigcup_{\alpha < \lambda} \Phi^\alpha.$$

Equivalently, for all ordinals,

$$\Phi^\beta = \bigcup_{\alpha < \beta} \{a : \varphi(\Phi^\alpha, a)\}.$$

Immerman-Vardi theorem: First-order plus least fixpoint captures polynomial time on ordered structures.

Still open: Gurevich conjectured that no logic can capture PTime on arbitrary (un-ordered) finite structures, even allowing a very generous meaning for “logic”.

Possible counterexample: Choiceless polynomial time ($\tilde{\text{CPT}}$) with counting. This “logic” is a computation paradigm allowing arbitrary finite structures as inputs, allowing very complicated data structures, allowing counting, but not allowing arbitrary choices.

There are several PTime Boolean queries that we think should not be computable in $\tilde{\text{CPT}}$ with counting, but none for which this has been proved. Ben Rossman has a proof for a non-Boolean query. In the Boolean case, there have been two queries that were expected to be outside $\tilde{\text{CPT}}$ plus counting but turned out not to be.

- Shelah gave a $\tilde{\text{CPT}}$ plus counting algorithm to determine whether a bipartite graph has a complete matching.
- Rossman gave a $\tilde{\text{CPT}}$ (without counting!) algorithm to distinguish Cai-Fürer-Immerman graphs.

Rossman's algorithm uses sets of high rank, increasing with the size of the input. Intuitively, the complexity of the data structures increases with the input size. Dawar and Richerby showed that such an increase is unavoidable, even if one allows counting.

Shelah's and Rossman's algorithms involve so much cleverness as to make it look unlikely that a general method can be found to express PTime computable properties in $\tilde{\text{CPT}}$ plus counting.

Earlier uses of inductive definitions, especially least fixed points:

- Recursion theory — e.g., characterization of Π_1^1 using LFP of arithmetical operators.
- Admissible sets.
- “Improve” the axioms of set theory.

The last refers to allowing inductively defined predicates in the axiom schemas of replacement and separation. But what should the axioms say about these predicates?

FO+LFP can define the standard natural numbers, without even using quantifiers: $a \in \mathbb{N}$ iff

$$0 \in \text{LFP}_{X,x}[x = a \vee X(x + 1)].$$

So there is no hope for a finitary deductive system to be complete for FO+LFP, even if LFP is applied only to quantifier-free, LFP-free formulas. Compton has a nice, complete, but (of course) infinitary deductive system for existential-LFP. What could one do with a natural but incomplete finitary system?

New formation rule: Given a formula $\varphi(X, x)$ in which X occurs only positively, introduce a new predicate C for the closure, $\text{LFP}_{X,x}[\varphi(X, x)]$. Assume the axiom $\varphi(C, x) \rightarrow C(x)$ and the axiom schema

$$\forall x[\varphi(\theta(\cdot), x) \rightarrow \theta(x)] \implies \forall x[C(x) \rightarrow \theta(x)].$$

Can one deduce from

$$\begin{aligned} & (\forall x, y, z)[X(x, y) \wedge X(x, z) \rightarrow y = z] \implies \\ & (\forall x, y, z)[\varphi(X, x, y) \wedge \varphi(X, x, z) \rightarrow y = z] \end{aligned}$$

that

$$(\forall x, y, z)[C(x, y) \wedge C(x, z) \rightarrow y = z]?$$

If each step preserves the property of being a partial function, does the whole iteration produce a partial function?

Least Fixed Point and Inflationary Fixed Point

Adding them to first-order produces equivalent logics (Kreutzer in general; Gurevich and Shelah for finite structures).

Intuitively they're very different. The iteration leading to a LFP can be done asynchronously.

$$S^0 = \emptyset, \quad S^n \subsetneq S^{n+1} \subseteq \{a : \varphi(S^n, a)\}$$

eventually stabilizes at the least fixed point. For IFP, this process can lead to the wrong fixed point.

So the equivalence of the two logics says, intuitively, that a computation that depends on synchronization (IFP) can be simulated by asynchronous agents (LFP). Those agents have to work harder, keeping track of stages that would be automatic for synchronized agents. Hence the heavy use of stage comparison relations in the equivalence proof.

Existential Least Fixed Point Logic

\exists LFP adds the least-fixed-point operator to first-order logic but removes \forall and restricts negation to (some) atomic formulas. It occurs naturally as the logic of while-programs:

- Its formulas can express the profiles of while programs.
- \exists LFP-sequents, $\forall x(\varphi(x) \rightarrow \psi(x))$ for \exists LFP-formulas φ and ψ , express the Hoare logic of while programs.

It was technically convenient to mark the relation symbols in each vocabulary as *negatable* or *positive*.

The distinction reflects a reality: For P to be negatable corresponds intuitively to a completeness assumption on a database. If it doesn't say $P(a)$, then in fact not $P(a)$. Positive predicates are open-ended; $P(a)$ might hold even if that information isn't available (yet).

A related distinction can also be easily reflected in the logic. If, for some sort of entity, we have a closed-world assumption saying that the only objects of that sort are the ones mentioned in the database, then extend the logic to allow universal quantification over that sort.

Homomorphisms of structures were required to preserve positive predicates and to both preserve and reflect negatable ones. Then they preserve \exists LFP-formulas. If we “close some worlds”, allowing \forall over those sorts, then homomorphisms should be surjections on those sorts.

Homomorphisms exhibit the possible future development of the database, if the reality doesn't change but the available data about it improves.

Truth values of \exists LFP-formulas are preserved by the inverse image parts of geometric morphisms of topoi.

The same goes for truth of \exists LFP-sequents.

Consequence: The following are equivalent, for an \exists LFP-sequent.

- It is valid in classical logic.
- It is valid in intuitionistic logic.
- Any condition forcing the antecedent also forces the consequent.

The “conditions” here are finite conjunctions of atomic formulas. “Forcing” is defined by induction on formulas of higher-order logic, similarly to set-theoretic forcing.

Open problem: What formulas of higher-order logic are preserved by inverse-image parts of geometric morphisms of topoi?

Such formulas would share the properties of \exists LFP above. I conjecture that they would have computational significance.

Game Logic

Lorenzen (1959): Constructive validity means defensibility in a debate between proponent and opponent. Lorenz, Felscher made the idea precise.

My “degrees of indeterminacy” (1972) involved natural operations on infinite (undetermined) games.

These operations looked formally like propositional connectives, and I pointed out that some of them gave a Brouwerian lattice of games. But I didn't have the idea to introduce a “logic” with, e.g., two kinds of conjunction. That idea was provided later, in a very different context, by Girard.

Determined games (where one player has a winning strategy) yield only classical propositional logic. For an “interesting” logic, one must avoid determinacy, as infinite games do.

Another way to evade determinacy: Require computable winning strategies. Rabin showed that even if the rules of the game are computable and each play is finite (so there is a winning strategy), there need not be a computable winning strategy. I added that one can rig the game so that its winning strategies are arbitrarily high in the hyperarithmetic hierarchy. That remains true even if the rules are feasibly computable, say PTime.

Games and Linear Logic

Girard introduced linear logic, motivated by proof theory and some aspects of denotational semantics (coherence spaces). I noticed that my operations on games satisfy the axioms of linear logic, and more.

To get an exact match with linear logic, the game semantics was modified by Abramsky, Jagadeesan, Hyland, and Ong. In fact they got full completeness: proofs correspond exactly to winning strategies. The cost includes requiring uniformity and history-freeness for the strategies. The payoff includes fully abstract models of some programming languages and many other developments.

Japaridze took the opposite approach: Keep the operations on games, don't require history-freeness or uniformity, and ask for the logic of the resulting system (where strategies are required to be computable). He has sound and complete axiomatizations for large fragments of the system. As a result, whenever all instances of a formula have winning strategies, then there is a uniform strategy covering all these instances.

The axiomatizations look very unlike traditional ones; they retain a certain flavor of games rather than logic.

Open problem: Find a more traditional-looking axiomatization of this game logic, or understand why none is possible.

Abstract State Machines, Logic Without Deduction

Gurevich's abstract state machine (ASM) model of computation uses concepts from first-order logic, other than the deductive apparatus. First, it uses first-order structures to model the states of a computation. The transition from one state to the next is modeled by rules, whose exact nature depends on the sort of computation being modeled (sequential or parallel, interactive or isolated, ...). In one important case, the transitions are given by interpretations.

Interpretations between theories were introduced by Tarski, Mostowski, and Robinson (1953) as a tool for transporting undecidability results from one theory to another. An interpretation from a theory T into T' tells how to interpret the basic predicates and functions of T by definitions in T' . It therefore converts a model of T' into a model of T .

When $T = T'$ and no axioms are involved, an interpretation is a transformation of structures for a fixed vocabulary. Those transformations are the transition functions for (non-interactive, Lipari-style) ASMs.