# THE LOGIC IN COMPUTER SCIENCE COLUMN

by

Yuri GUREVICH

Microsoft Research
One Microsoft Way, Redmond, WA 98052, USA
gurevich@microsoft.com

## The Underlying Logic of Hoare Logic[1]

Andreas Blass[2] and Yuri Gurevich[3]

**Abstract**

Formulas of Hoare logic are asserted programs $\varphi\ \Pi\ \psi$ where $\Pi$ is a program and $\varphi, \psi$ are assertions. The language of programs varies; in the survey [Apt 1980], one finds the language of `while` programs and various extensions of it. But the assertions are traditionally expressed in first-order logic (or extensions of it). In that sense, first-order logic is the underlying logic of Hoare logic. We question the tradition and demonstrate, on the simple example of `while` programs, that alternative assertion logics have some advantages. For some natural assertion logics, the expressivity hypothesis in Cook's completeness theorem is automatically satisfied.

The readers of this column know Quisani, an inquisitive former student of one of us. This conversation is somewhat different because Quisani talks to both of the authors at once. Taking into account the magnitudes of our literary talents, we simplified the record of the conversation by blending the two authors into one who prefers "we" to "I". While it is great to talk to Quisani, his insatiable appetite to motivation and detail can be a burden. In the appendix, we leave Quisani and address some decision problems related to existential fixed-point logic, one of the alternative assertion logics.

[1]Bull. of the Euro. Assoc. for Theoretical Computer Science, 70, Feb. 2000, 82–110.
[2]Mathematics, University of Michigan, Ann Arbor, MI 48109-1109, USA.
[3]On leave of absence from the University of Michigan.

We presume that the reader is familiar with the basics of first-order logic and computation theory. We do not presume any familiarity with Hoare logic. Our main reference on Hoare logic is [Apt 1981].

# 1 Background, `while` Programs, and Global Relations

**Quisani:** In the introduction of your paper on existential fixed-point logic, you promise to "show in Section 2 that the use of existential fixed-point logic removes the need for an expressivity hypothesis in Cook's completeness theorem for Hoare logic" [BG 1987, page 20]. I don't think you do that.

**Author:** The material on Cook's completeness theorem was our original motivation for the study of existential fixed-point logic. We are surprised that it isn't in the paper and we don't have a good explanation.

**Q:** Did you intend to use existential fixed-point logic instead of first-order logic?

**A:** Exactly. First-order logic is used a little too readily in computer science. Sometimes there are logics more appropriate for the purpose.

**Q:** That sounds interesting. Tell me more about it. But please start at the beginning and remind me what Hoare logic is. And don't rely on my knowledge of [BG 1987] either; I didn't read it that carefully.

**A:** We start with the `while` programming language. Programs in that language are defined inductively.

- For every variable $x$ and every (first-order) term $t$,

  $x := t$

  is a program.

- If $\Pi_1, \Pi_2$ are programs and *guard* is a quantifier-free first-order formula, then

  $\Pi_1; \Pi_2$
  `if` *guard* `then` $\Pi_1$ `else` $\Pi_2$ `fi`
  `while` *guard* `do` $\Pi_1$ `od`

  are programs.

A program of the form $x := x$ will be denoted `skip`. Else-clauses of the form `else skip` may be omitted. We will sometimes also omit the keywords `fi` and `od`; if necessary, parentheses will be used to insure unambiguous parsing.

**Q:** What is the input to a program? What does the program compute?

**A:** First we define the states of given program $\Pi$. Let $\bar{v} \rightleftharpoons (v_1, \ldots, v_k)$ be the list of the (individual) variables of $\Pi$. We fix once and for all a canonic ordering of all variables and we always list variables in the canonic order. Fix a structure $A$ whose vocabulary contains all function and relation symbols of $\Pi$. Over $A$, a state of $\Pi$ is (given by) a tuple $\bar{a} \rightleftharpoons (a_1, \ldots, a_k)$ of values for the variables of $\Pi$. (The point of the canonic ordering is to let us write states as tuples rather than as (partial) functions on the set of variables.)

Any state $\bar{a}$ is a possible input to $\Pi$ over $A$. We hope that the standard notation for programs is clear enough so that we don't need to explicitly define the computation of a program on an input state. The definition is written out in full in [Cook 1978] and [Apt 1981]. The computation of $\Pi$ on input $\bar{a}$ may or may not terminate. If it terminates, set $\hat{\Pi}^A(\bar{a})$ equal to the final state. Over $A$, $\Pi$ computes the partial function $\bar{y} = \hat{\Pi}^A(\bar{x})$. The graph of $\hat{\Pi}$ was called the profile of $\Pi$ in [BG 87]:

$$\text{Profile}_\Pi(\bar{x}, \bar{y}) \rightleftharpoons \hat{\Pi}(\bar{x}) = \bar{y}.$$

**Q:** You didn't use the superscript $A$. I guess $\text{Profile}_\Pi$ is one of those global relations that we talked about once. Please refresh my memory.

**A:** A $j$-ary *global relation* $\rho$ of vocabulary $\Upsilon$ is a function that assigns to every $\Upsilon$-structure $A$ a (local) $j$-ary relation $\rho^A$ on $A$; it is required that $\rho$ be *abstract* in the following sense: every isomorphism between $\Upsilon$-structures $A$ and $B$ is also an isomorphism between the expanded structures $(A, \rho^A)$ and $(B, \rho^B)$ [Gurevich 1984]. Indeed, $\text{Profile}_\Pi$ is a global relation.

**Q:** What if you want to restrict attention to a class of structures, say, to finite structures?

**A:** Let $K$ be a class of structures. A $j$-ary *$K$-global relation* $\rho$ of vocabulary $\Upsilon$ is a function that assigns to every $\Upsilon$-structure $A$ in $K$ a (local) $j$-ary relation $\rho^A$ on $A$. Again it is required that $\rho$ is abstract, i.e., that it respects isomorphism between structures in $K$.

# 2 Hoare Logic: Syntax and Semantics

**A:** We are ready to recall the Hoare logic (for `while` programs). A formula of the Hoare logic is an *asserted program*

$$\varphi \; \Pi \; \psi$$

where $\Pi$ is a `while` program and the assertions $\varphi, \psi$ are first-order formulas called the *precondition* and *postcondition* respectively. The *vocabulary* of the asserted program consists of the function and relation symbols that occur in $\varphi, \Pi$, or $\psi$.

**Q:** Define precisely the validity of an asserted program $\varphi\ \Pi\ \psi$.

**A:** There are two distinct definition of the validity or *correctness* of asserted programs. Without loss of generality, we may assume that the variables $\bar{v}$ of $\Pi$ are exactly the free variables of $\varphi$ and exactly the free variables of $\psi$. Suppose that $A$ ranges over structures whose vocabulary includes that of $\varphi\ \Pi\ \psi$.

**Partial Correctness** $\varphi(\bar{v})\ \Pi(\bar{v})\ \psi(\bar{v})$ is *partially correct* (over $A$) if the formula

$$\Big(\varphi(\bar{x}) \wedge \text{Profile}_{\,\Pi}(\bar{x}, \bar{y})\Big) \rightarrow \psi(\bar{y}),$$

is valid (over $A$).

**Total Correctness** $\varphi\ \Pi\ \psi$ is *totally correct* (over $A$) if it is partially correct and the formula

$$\varphi(\bar{x}) \rightarrow (\exists \bar{y})\text{Profile}\,(\bar{x}, \bar{y})$$

is valid (over $A$).

**Q:** As I see it, both forms of correctness say that $\varphi$ at the beginning of a computation guarantees $\psi$ at the end. The difference is that partial correctness gives this guarantee only *if* the computation terminates, while total correctness requires the computation to terminate *and* gives the guarantee.

**A:** Exactly. Let us concentrate today on partial correctness. We can discuss total correctness some other time.

# 3 Proof System for Hoare Logic

**Q:** I guess Hoare logic has also axioms and rules of inference. Otherwise you would not talk about its completeness.

**A:** Right. The traditional proof system $\mathcal{H}$ of Hoare logic for `while` programs comprises an axiom schema and four inference rules.

**Assignment Axiom Schema**

$$\varphi(t) \quad x := t \quad \varphi(x)$$

where $\varphi(t)$ is the result of substituting the term $t$ for the variable $x$ in $\varphi(x)$. (We tacitly assume that bound variables in $\varphi(x)$ are renamed if necessary to avoid clashes with variables in $t$.)

**Consequence Rule**

Premises: $\varphi \to \varphi', \quad \varphi' \ \Pi \ \psi', \quad \psi' \to \psi.$
Conclusion: $\varphi \ \Pi \ \psi.$

**Composition Rule**

Premises: $\varphi \ \Pi_1 \ \chi, \quad \chi \ \Pi_2 \ \psi.$
Conclusion: $\varphi \ \Pi_1; \Pi_2 \ \psi.$

**Conditional Rule**

Premises: $(\varphi \wedge \mathit{guard}) \ \Pi_1 \ \psi, \quad (\varphi \wedge \neg \mathit{guard}) \ \Pi_2 \ \psi.$
Conclusion: $\varphi$ (`if` $\mathit{guard}$ `then` $\Pi_1$ `else` $\Pi_2$ `fi`) $\psi$

**Iteration Rule**

Premise: $(\varphi \wedge \mathit{guard}) \ \Pi \ \varphi.$
Conclusion: $\varphi$ (`while` $\mathit{guard}$ `do` $\Pi$ `od`) $(\varphi \wedge \neg \mathit{guard})$.

$\mathcal{H}$ is obviously sound for the Hoare logic, that is for the partial correctness version of the logic. It is not sound for the total correctness version because of the Iteration Rule.

**Q:** The Consequence Rule is impure. Two of the three premises are first-order implications rather than asserted formulas. One can dress up implications as asserted `skip` programs.

**A:** One can, but it would make little difference. The logic would not provide a way to prove such asserted `skip` programs. These two premises of the consequence rule, whether viewed as implications or dressed up as asserted programs, have to be obtained from someplace outside Hoare logic.

**Q:** Where would that be?

**A:** That depends on the context, so let us describe a few possible contexts and see what the corresponding sources for implications might be. We defined validity of asserted programs over a structure $A$, and indeed one often has a single structure $A$ in mind (for example the standard model of arithmetic) and wants to use Hoare logic to deduce asserted programs valid over this $A$. Then of course the implications in the consequence rule should also be valid in this fixed $A$. For some structures, such as the field of real numbers, the first-order sentences true in $A$ constitute a recursively axiomatizable theory. In such a case, the source of the implications in the consequence rule could be such an axiomatization plus the standard deductive rules of first-order logic. For other structures, like the standard model of arithmetic, the first-order theory is not recursively axiomatizable, and indeed there is

no effective source for all arithmetical truths. But there are effective sources for all the arithmetical truths ever actually needed. For example, the Zermelo-Fraenkel (ZFC) axioms of set theory, generally regarded as the appropriate foundation for all of mathematics, suffice to yield, via the deductive rules of first-order logic, all the usual facts of number theory. So these axioms could be regarded as the source of the implications in the consequence rule.

But Hoare logic can be used in other contexts. Specifically, we might have a class $K$ of structures in mind, and we might want to deduce asserted programs valid in all structures from $K$. Then of course the implications in the consequence rule should be valid in all structures from $K$ as well. Whether there is an effective source for these implications (i.e., whether they form a recursively axiomatizable theory) depends, of course, on $K$. But even if there is no effective source for all the $K$-valid implications, ZFC will usually provide an effective source for all the implications one actually needs. Notice, by the way, that if $K$ is the class of *all* structures of a fixed vocabulary, then Gödel's completeness theorem for first-order logic provides the effective source we need.

# 4 Incompleteness of Hoare Logic

**Q:** But if Hoare logic needs to be supplemented with an external source of valid implications, then it really isn't complete. Why does the problem remain if the implications in the consequence rule are regarded as asserted `skip` programs and thus as the sort of thing $\mathcal{H}$ might conceivably prove?

**A:** The only asserted `skip` programs that could be proved without the consequence rule are the trivial ones of the form $\varphi$ `skip` $\varphi$, which are assignment axioms. Now consider any non-trivial but partially correct

$$\varphi \texttt{ skip } \psi.$$

For example, $\varphi$ may be `false` while $\psi$ is anything but `false`, or $\psi$ may be `true` while $\varphi$ is anything but `true`. Any proof of this asserted program would have to invoke the Consequence Rule and thus would require some valid implications, or some asserted `skip` programs. And trivial ones are of no use here. So in order to prove any non-trivial asserted `skip` program, we'd need some previously proved, non-trivial, asserted `skip` programs.

So, to avoid an infinite regress, $\mathcal{H}$ needs an external source of implications.

**Q:** Why not incorporate the deductive apparatus of first-order logic into $\mathcal{H}$? This would allow us to derive all valid first-order implications.

**A:** Good idea, but we are not interested in going into the details of a proof system for first-order logic. It is simpler just to declare that every valid first-order implication is an axiom:

**FO Axiom Schema**

All valid first-order implications $\varphi \to \psi$

Notice the separation of concerns. Assertions apply to one state, and programs transform states. Thus, the FO axiom schema reflects the static part of the proof system, and $\mathcal{H}$ reflects the dynamics.

**Q:** In the FO Axiom Schema, does "valid" mean true in all structures for the vocabulary, as usual in first-order logic?

**A:** Yes. So this schema is intended to be used in Hoare logic for deducing asserted programs valid in all structures at once.

**Q:** If you were dealing with validity of asserted programs in just one structure $A$ or in a class of structures $K$, then you could just replace "valid" in the FO Axiom Schema with "true in $A$" or with "valid in $K$."

**A:** Right. We'll call the resulting schemas the FO($A$) or FO($K$) Axiom Schemas.

Unfortunately, $\mathcal{H}$ + FO Axiom Schema isn't complete either. An asserted program

`true` $\Pi$ `false`

is partially correct if and only if $\Pi$ never halts. $\mathcal{H}$ + FO Axiom Schema cannot prove all partially correct assertions of that form.

**Q:** Sketch a proof.

**A:** This will require some basic computation theory. Notice that the set of asserted programs provable in $\mathcal{H}$ + FO Axiom Schema is recursively enumerable. It suffices to prove that the set of never halting `while` programs is not recursively enumerable.

Consider Turing machines that start on the empty tape. (We adopt the convention that the tape is potentially infinite to the right but that it has a leftmost square, which is the scanned square at the start of the computation.) Computation theory tells us that the set of nonhalting machines is not recursively enumerable. Every Turing machine $M$ can be simulated by a `while` program $\Pi_M$ such that $\Pi_M$ never halts if and only if $M$ is non-halting. It follows that the set of never halting programs is not recursively enumerable.

We sketch the construction of $\Pi_M$. Without loss of generality, we assume that control states and tape symbols of $M$ are natural numbers, the initial state is 1,

the halting state is 0, and the blank is also 0. The idea is that $\Pi_M$ will halt if and only if its input structure $A$ encodes a halting computation of $M$. Here's how the encoding works.

Given a halting computation of $M$, fix an integer $n$ larger than the number of steps in the computation and therefore also larger than the length of the segment of the tape visited by $M$ during the computation. Identify that segment of the tape with an initial segment of $[0, n]$, so 0 represents the leftmost tape square. Also assume that $n$ exceeds the integers serving as control states or as tape symbols. Let the domain of the structure $A$ be the interval $[0, n]$ of integers. The vocabulary of $A$ consists of two constant symbols 0 and End, four unary function symbols $S$, $P$, $Q$, and $X$, and a binary function symbol $U$. The interpretations of 0 and End in $A$ are 0 and $n$, respectively. The interpretations of $S$ and $P$ in $A$ are the successor and predecessor functions, extended to the endpoints by $S^A(n) = n$ and $P^A(0) = 0$. $Q^A(k)$ is the control state of $M$ at step $k$ of the computation, $X^A(k)$ is the square scanned by $M$ at step $k$, and $U^A(k, l)$ is the symbol in square $l$ at step $k$. (If $k$ is larger than the number of steps in the computation, then we take "step $k$" to be the same as the last actual step.)

Note that there is some ambiguity in this specification of $A$, because we could increase $n$. In fact, it is convenient to allow somewhat more ambiguity. We regard a structure $A$ (for the vocabulary specified above) as coding the given computation of $M$ provided that

(1) Starting from 0 and iterating $S$, one eventually reaches End; say End $= S^n(0)$.

(2) The set $\{0, S(0), S^2(0), \ldots, S^{n-1}(0), End\}$ is the domain of a substructure $A_0$ of $A$; that is, this set is closed under the interpretations in $A$ of all the function symbols.

(3) This substructure $A_0$ is isomorphic to one of the sort described above.

Now we can describe a `while` program that halts exactly on those structures that code halting computations of $M$. The program uses two variables, $s$ and $t$ (which we think of as space and time) and it has the form

$s := 0;\ t := 0;$
`while` $s \neq$ `End do` $s := S(s);$
$s := 0;$
`Check,`

where Check is a program to be described in a moment. The first two lines here verify that condition (1) on our coding structures is satisfied; if (1) is violated, then the `while` command never terminates. The third line just reinitializes $s$. The real work will be done by the Check program which, as its name suggests, checks that the functions behave properly on $A_0$. It systematically searches for violations of requirements (2) and (3). In some detail, it looks like

```
if Q(0) ≠ 1 ∨ Q(End) ≠ 0 ∨ X(0) ≠ 0 ∨ S(End) ≠ End then loop fi;
while s ≠ End do
   if U(0, s) ≠ 0 then loop fi;
   s:=S(s) od;
if U(0, End) ≠ 0 then loop fi;
s := 0;
while t ≠ End do
   if t = 0 ∧ P(t) ≠ 0 then loop fi;
   if P(S(t)) ≠ t then loop fi;
   while s ≠ End do Local-Check; s:=S(s) od;
   Local-Check;
   s := 0; t := S(t)
od.
```

Here `loop` is a program that never terminates, for example `while true do skip`, and Local-Check verifies that what happens in the step from stage $t$ to stage $S(t)$ is in accordance with the instructions of $M$, i.e., that $Q(t)$, $X(t)$, and $U(t, X(t))$ are updated as specified by the instructions of $M$ and that $U(t, s)$ is not changed if $s \neq X(t)$. Local-Check is obtained by stringing together with semicolons several conditional statements, each saying "if something is improperly updated then loop." Writing it out in detail seems a waste of time, unless you have some questions about it.

**Q:** Writing out Local-Check seems like an easy enough exercise. But what's the point of its second occurrence in the program?

**A:** The first occurrence, inside the `while` loop, ran the check only for $s \neq$ End; the second occurrence runs it for $s =$ End also.

**Q:** Oh, it's just like the line "`if` $U(0, \text{End}) \neq 0$ `then loop fi`", which also covers an End situation missed by the preceding loop.

**A:** Exactly.

**Q:** Was this incompleteness phenomenon known when Cook proved his completeness theorem?

**A:** In his paper [Cook 1978] containing the completeness theorem, Cook points out this same incompleteness for Hoare logic in the context of a single structure, namely the standard model of arithmetic. This is the context of primary interest in that paper.

**Q:** What exactly is "the same incompleteness" in a different context?

**A:** We mean incompleteness that is attributable to asserted programs of the form

`true` $\Pi$ `false` and the fact that the non-halting problem isn't recursively enumerable.

# 5 Cook's Completeness Theorem

**Q:** If Hoare Logic is so hopelessly incomplete, then what is Cook's theorem about?

**A:** To explain Cook's theorem, we need a couple of definitions.

Let $\bar{v}$ be the list of variables of a program $\Pi$, let $\varphi(\bar{v})$ be a first-order formula with free variables in $\bar{v}$, and let $A$ be a structure of sufficiently rich vocabulary. The *strongest postcondition* $\mathrm{Post}_\Pi(\varphi)$ is the global relation

$$(\exists \bar{x})[\varphi(\bar{x}) \wedge \mathrm{Profile}_\Pi(\bar{x}, \bar{v})].$$

So $\mathrm{Post}^A_\Pi(\varphi)$ holds of just those states (in structure $A$) that are the final results of (terminating) runs of $\Pi$ starting in states that satisfy $\varphi$ in $A$.

**Q:** I suppose the name "strongest postcondition" means that $\mathrm{Post}^A_\Pi(\varphi)$ is the strongest relation $R$ on $A$ such that $\varphi\, \Pi\, R(\bar{v})$ is valid.

**A:** Right.

Say that first-order logic is *Cook expressive* for a structure $A$ if for all $\Pi$ and $\varphi$ as above, the relation $\mathrm{Post}^A_\Pi(\varphi)$ is first-order expressible in $A$. In other words, first-order logic is expressive for $A$ if and only if, for all $\Pi$ and $\varphi$, there is a first-order formula $\psi(\bar{v})$ such that, for every state $\bar{x}$ of $\Pi$ over $A$, we have

$$\bar{x} \in \mathrm{Post}^A_\Pi(\varphi) \iff A \models \psi(\bar{x})$$

Later, when we consider logics other than first-order, the same definition will be applied to them: For each assertion $\varphi$ in the logic, the strongest postcondition should be expressible by an assertion in the logic. Also, we can talk about a logic being expressive for a whole class $K$ of structures, meaning that, for each program $\Pi$ and each assertion $\varphi$, there is a single $\psi$ defining $\mathrm{Post}^A_\Pi(\varphi)$ in all structures $A \in K$ simultaneously.

**Theorem 1 (Cook 1978)** *If first-order logic is Cook expressive for $A$ then $\mathcal{H}$ + FO($A$) Axiom Schema is complete for Hoare Logic, so that every asserted program partially correct over $A$ is provable in $\mathcal{H}$ + FO($A$) Axiom Schema.*

The proof is in [Cook 1978] and in the survey paper [Apt 1981].[4]

[4] We shall review this proof in Section 6.

**Q:** I have a whole bunch of questions. Let me ask them in some random order. You defined the strongest postcondition from the profile. It seems that the profile can be defined from the strongest postcondition, so that the strongest postcondition can be replaced by the profile in the definition of expressivity.

**A:** You are right. Let us leave the problem of defining the profile from the strongest postcondition as an exercise; the solution is in [BG 1987].

**Q:** Let me try to formulate the weakest precondition $\mathrm{Pre}_\Pi(\psi)$, the dual of the strongest postcondition. It should be the global relation

$$\{\bar{v} : (\exists \bar{y})[\mathrm{Profile}\,(\bar{v}, \bar{y}) \wedge \psi(\bar{v})\}.$$

**A:** This is the weakest precondition with respect to total correctness. That is, it is the weakest global relation $R$ such that $R(\bar{v})\,\Pi\,\psi$ is valid in the sense of total correctness. In the case of partial correctness you have a weaker global relation, namely:

$$\{\bar{v} : (\forall \bar{y})[\mathrm{Profile}\,(\bar{v}, \bar{y}) \rightarrow \psi(\bar{v})]\}.$$

**Q:** I see. It seems that the profile can also be defined from the weakest precondition, so that the strongest postcondition can be replaced by the weakest precondition in the definition of expressivity. Is this another exercise with a solution in [BG 1987]?

**A:** Sure. The observation that Cook's theorem remains true (and its proof is somewhat simplified) when strongest postcondition expressivity is replaced with weakest precondition expressivity is due to Clarke [Apt 1981, p. 439].

**Q:** Give me an example of a structure for which first-order logic is expressive.

**A:** The most important example is arithmetic, that is the standard model $\langle N, +, *, 0, 1\rangle$ of arithmetic [Cook 1978, Lemma 5]. Here $N$ is the set of natural numbers. More generally, first-order logic is expressive for a structure $A$ if in $A$, as in arithmetic, arbitrary finite sequences of elements can be coded in first-order way by means of single elements.

**Q:** Why do you need such a coding?

**A:** Prove, by induction on program $\Pi$, that the profile of $\Pi$ is first-order expressible. The only difficult case arises when $\Pi$ has the form `while` *guard* `do` $\Pi_0$. For simplicity, let us consider the case when $\Pi$ has just one variable $v$, so that states of $\Pi$ over $A$ are given by elements of $A$. By the induction hypothesis, you know that $\mathrm{Profile}^A_{\Pi_0(x,y)}$ is expressible by some formula $\chi_0(x, y)$. To express $\mathrm{Profile}^A_\Pi(x, y)$ of

$\Pi$, you say that there exists an element $s$ coding a sequence $\langle s_0, s_1, \ldots, s_n \rangle$ such that the following holds in $A$:

- $s_0 = x$, $s_n = y$,
- $guard(s_i) \wedge \chi_0(s_i, s_{i+1})$, for all $i < n$,
- $\neg guard(s_n)$.

If $\Pi$ contains $k$ variables then $s$ should code a sequence of $k$-tuples. I trust you can make the necessary modification.

**Q:** Sure, but this only shows that you need coding for a particular way of expressing profiles. I admit that it looks like the natural way to do this, but might there be a different approach that doesn't need to code sequences? Can you give an example of a structure for which first-order logic is not expressive?

**A:** Yes, the field $\langle R, 0, 1, +, * \rangle$ of real numbers is such an example. The program $\Pi$:

```
x:=0;
while x ≠ y do x := x + 1
```

halts if and only if the initial value of $y$ (which is never altered) is a non-negative integer. So if $\psi(x, y)$ defined $\mathrm{Post}^R_\Pi(\mathtt{true})$ then $\psi(0, y)$ would define the set $N$ of natural numbers in the field $R$. But, by a celebrated theorem of Tarski, the only subsets of $R$ first-order definable from the field structure are finite unions of intervals (including degenerate and infinite intervals) with algebraic endpoints. Since $N$ is not of that form, $\mathrm{Post}^R_\Pi(\mathtt{true})$ is not first-order definable.

**Q:** OK, let us consider the case when $A$ is arithmetic. I know that the first-order theory of $A$ is not recursively enumerable. The FO($A$) Axiom Schema is a fake.

**A:** You have a point there. What Cook gives us is a clean separation of concerns. The FO(Arithmetic) Axiom Schema takes care of the statics, and $\mathcal{H}$ takes care of the dynamics. One may hope and expect that, in applications, proofs of asserted programs will not require subtle arithmetical facts. Remember what we said earlier about the arithmetical facts provable in ZFC: They from a recursively enumerable set adequate for just about any application of $\mathcal{H}$.

**Q:** I like the separation of concerns in Cook's theorem, but FO(Arithmetic) is way too complex. Cook's incompleteness argument shows that the set of partially correct asserted programs is at least as complex as the not recursively enumerable set of nonhalting Turing machines. So there is no regular proof system for $\mathcal{H}$, and the completeness of $\mathcal{H}$ can be achieved only by liberalizing the notion of proof system. However it is reasonable to expect a modest liberalization. The FO(Arithmetic) Axiom Schema does not live up to this expectation. I remember that the set of true arithmetical formulas is much more complex then the set of nonhalting Turing machines.

**A:** You are right. The set (of the Gödel numbers) of nonhalting Turing machines can be described by an arithmetical formula, in fact by a $\Pi^0_1$ arithmetical formula (see for example [Rogers 1967, Section 14.3] or [Enderton 1977, page 556]). Far more complex sets can be described by arithmetical formulas with nested unbounded quantifiers. By Tarski's theorem on undefinability of truth (see for example [Enderton 1977, page 558]), the FO(Arithmetic) Axiom Schema is not expressible in arithmetic.

But notice that this excess complexity is unavoidable as long as we allow arbitrary first-order formulas as assertions. A sentence $\varphi$ is true in the standard model of arithmetic if and only if the asserted program `true skip` $\varphi$ is true there. So the set of true asserted programs is at least as complex as the set of true first-order sentences.

**Q:** My other complaint is that Cook's completeness theorem relates to single structures rather than to all structures at once.

**A:** Cook's theorem can be modified to apply to a class $K$ of structures. One should replace the FO($A$) Axiom Schema by the FO($K$) Axiom Schema, and one then gets deducibility of all asserted programs valid in $K$.

# 6 Generalized Cook's Theorem

**A:** Although Cook's completeness theorem is usually stated in the context of first-order logic, it actually uses far weaker closure conditions on the logic. Let's review the proof and isolate what it assumes about the underlying logic.

Following [Cook 1978], we begin with two first-order vocabularies $\Upsilon_1 \subseteq \Upsilon_2$ and build terms in the usual way from variables and function symbols. Where we may deviate from traditional first-order logic is in the definition of formulas. Actually, three sorts of formulas are involved in Hoare logic and Cook's completeness theorem:

**Guards** These are the formulas that appear within programs, as the guards of conditional statements and while-statements.

**Assertions** These appear as the pre- and post-conditions in asserted programs.

**Sequents** These are implications between assertions. They occur as premises in the consequence rule.

We will consider various logics $L$ but restrict attention to vocabularies $\Upsilon_1 \subseteq \Upsilon_2$. Cook uses the smaller vocabulary $\Upsilon_1$ in programs, for the guards and for the terms

in assignments, and he uses the larger vocabulary $\Upsilon_2$ in assertions. This makes sense because the specification of a program may involve concepts not used in the program itself. For example, a program deciding whether a graph contains a path from one specified vertex to another is likely to use only the adjacency relation, not the path relation. Then $\Upsilon_2$ could contain the path relation while $\Upsilon_1$ would not. For many situations, however, one can assume that the two vocabularies coincide.

**Syntax** A logic $L$ specifies what the formulas of any vocabulary are. It also specifies which $\Upsilon_1$-formulas are guards, which $\Upsilon_2$-formulas are assertions, and what the sequents are (but see condition 5 below). A possibly unfortunate custom, in connection with some of the logics we shall consider, is to define "formula" rather narrowly, so that sequents are not considered formulas. Because of this, we shall formulate our general requirements on $L$ just in terms of guards, assertions, and sequents, not referring at all to $L$'s notion of formula. Later, when dealing with specific logics, we shall use "formula" in the sense customary for those logics.

We require the following of our logics $L$.

1. Conjunction of a guard with a disjunction of negations of equations between variables is again a guard.
2. Substituting an $\Upsilon_1$-term for a variable in an assertion produces an assertion.
3. The conjunction of an assertion with a guard or with the negation of a guard is again an assertion.
4. Existentially quantifying an assertion produces an assertion.
5. Every implication between assertions is a sequent.

**Q:** An implication between assertions is also an assertion, isn't that true?

**A:** Not necessarily. Some logics of interest to us do not employ implication in producing assertions. Requirement 5 makes implication available in sequents even if it isn't available in assertions. But even in sequents, implication is required only between assertions; we never need nested implications.

**Q:** These requirements look more technical than natural.

**A:** That's true, because they were chosen to be just what's needed for the proof of Cook's theorem. They could be replaced by more natural hypotheses that are somewhat stronger than needed. For example, we could assume

- The guards are the quantifier-free $\Upsilon_1$-formulas in the sense of first-order logic.

- All guards are assertions.
- The class of assertions is closed under renaming variables, under conjunction and under existential quantification.
- The sequents are all the implications between assertions.

**Q:** These assumptions look better, and they clearly imply the old ones, except that you forgot the requirement (2) that assertions be closed under substitution.

**A:** Substitution can be simulated with renaming variables, conjunction, and existential quantification. $\varphi(t)$ is equivalent to $\exists y\,(y = t \wedge \varphi(y))$ for a fresh variable $y$.

**Semantics** For our purposes, models for $L$ are structures of vocabulary $\Upsilon_2$. If $A$ is such a structure and $\varphi$ is an $L$ formula in vocabulary $\Upsilon_2$, then the satisfaction relationship $A \models \varphi$ is defined. Equality is interpreted as identity. Any of the standard propositional connectives and quantifiers present in $L$ are interpreted in the usual way. Sequents, like asserted programs, are regarded as true when they are satisfied by all choices of values for their free variables; thus they should be regarded as being implicitly universally quantified.

In what follows, we consider programs in vocabulary $\Upsilon_1$ and fix a class $K$ of $\Upsilon_2$ structures.

**Definition 2** $L$ is *expressive* for $K$ if, for any program $\Pi$ and any $L$ assertion $\varphi$, the $K$-global relation $\mathrm{Post}_\Pi(\varphi)$ is expressible as an $L$ assertion.

In other words, expressivity requires that there exists an $L$ assertion $\psi(\bar{x})$ such that, for every $A \in K$ and every state $\bar{x}$ of $A$, we have

$$\bar{x} \in \mathrm{Post}^A_\Pi(\varphi) \iff A \models \psi(\bar{x}).$$

Let $\mathcal{H}_L$ be the version of Hoare logic $\mathcal{H}$ where the guards, assertions and sequents are those of $L$. Further, let $\mathcal{H}_L(K)$ be the result of adding all $K$-valid $L$-sequents to $\mathcal{H}_L$ as axioms.

**Theorem 3** *Assume that $L$ is expressive for $K$. Then every $K$-valid asserted program is deducible in $\mathcal{H}_L(K)$.*

The proof of Cook's theorem, in the form stated above, requires nothing more than the traditional proof for first-order logic. We just have to pay attention to

the constructions actually used in the logic. Nevertheless, we shall give the proof and point out where the hypotheses are used.

Our hypotheses (2) and (3) are used in the very formulation of $\mathcal{H}_L$. Substitution of terms for variables in an assertion occurs in the assignment axiom schema, and conjunction of assertions with guards and negated guards occurs in the conditional rule and the iteration rule. The purpose of the other three hypotheses will become clear during the proof. Throughout this discussion, validity, both of sequents and of asserted programs, will mean $K$-validity, i.e., truth in all structures from the fixed class $K$.

**Proof** We must show that a valid asserted program $\varphi\ \Pi\ \psi$ is deducible. To do this we proceed by induction on the structure of the program $\Pi$.

Suppose first that $\Pi$ is an assignment, $x := t$. Then validity of the asserted program $\varphi\ \Pi\ \psi(x)$ means exactly that the sequent $\varphi \to \psi(t)$ is valid. (Here we use requirement (5) for the first time.) But from this sequent and the instance $\psi(t)\ \Pi\ \psi(x)$ of the assignment rule (and, to be pedantic, the valid sequent $\psi(x) \to \psi(x)$), we deduce the required $\varphi\ \Pi\ \psi(x)$ by the consequence rule.

Next, suppose $\Pi$ is a sequential composition $\Pi_1; \Pi_2$. Let assertion $\theta$ express $\mathrm{Post}_{\Pi_1}(\varphi)$. Then by definition of this strongest postcondition, $\varphi\ \Pi_1\ \theta$ is valid, and the assumed validity of $\varphi\ \Pi\ \psi$ means that $\theta\ \Pi_2\ \psi$ is also valid. By induction hypothesis, both $\varphi\ \Pi_1\ \theta$ and $\theta\ \Pi_2\ \psi$ are deducible, and they yield $\varphi\ \Pi\ \psi$ by the composition rule.

Third, suppose $\Pi$ is `if` *guard* `then` $\Pi_1$ `else` $\Pi_2$. Now if $\varphi\ \Pi\ \psi$ is valid, then so are $(\varphi \wedge \mathit{guard})\ \Pi_1\ \psi$ and $(\varphi \wedge \neg \mathit{guard})\ \Pi_2\ \psi$. (Here we use hypothesis (3) on our logic.) By induction hypothesis, these asserted programs are deducible, and they yield the required $\varphi\ \Pi\ \psi$ by the conditional rule.

Finally, suppose $\Pi$ is `while` *guard* `do` $\Pi_1$. Of course, we intend to deduce $\varphi\ \Pi\ \psi$ by means of the iteration rule, but in order to do this we must first produce a suitable loop-invariant (the $\varphi$ in the statement of the iteration rule).

Let $\bar{x}$ be a list of all the variables free in $\varphi\ \Pi\ \psi$, and let $\bar{x}'$ be an equally long list of fresh variables. Let $\Pi'$ be the program

$$\texttt{while}\ \mathit{guard} \wedge \bigvee_i x_i \neq x_i'\ \texttt{do}\ \Pi_1.$$

This program works almost like $\Pi$ on the $x$ variables, leaving the values of the $x'$ variables unchanged; the only difference is that $\Pi'$ may stop earlier than $\Pi$. It may stop before *guard* becomes false if it reaches a state where $\bar{x} = \bar{x}'$, i.e., where $\Pi$ has brought the $x$ variables into agreement with the (original, never altered) $x'$ variables. (Hypothesis (1) on $L$ is used to ensure that $\Pi'$ is a program, i.e., that its guard is a legitimate one.)

Let $\theta(\bar{x}, \bar{x}')$ be a formula expressing $\mathrm{Post}_{\Pi'}(\varphi)$ over $K$. It holds if and only if either $\bar{x} = \bar{x}'$ and the state $\bar{x}$ is reached at some stage during the execution of $\Pi$ starting in some state satisfying $\varphi$ or else $\bar{x}$ is the final state of a run of $\Pi$, starting in some state satisfying $\varphi$, such that state $\bar{x}'$ never occurred during the run.

Accordingly $\exists \bar{x}'\, \theta(\bar{x}, \bar{x}')$ holds if and only if $\bar{x}$ is a state reached at some stage during a run of $\Pi$ starting at some state satisfying $\varphi$. This implies that

$$\Big((\exists \bar{x}'\, \theta(\bar{x}, \bar{x}')) \wedge \mathit{guard}\Big)\ \Pi_1\ \exists \bar{x}'\, \theta(\bar{x}, \bar{x}')$$

is valid. Indeed, if a state is reachable from one satisfying $\varphi$ and if the guard is still true, then one more step of the iteration will, if it yields any state at all, yield one reachable from (the same) one satisfying $\varphi$. (Hypotheses (4) and (3) on $L$ ensure that the assertions used here are legitimate ones.) So, by induction hypothesis, this asserted program is deducible. By the iteration rule, we can deduce

$$(\exists \bar{x}'\, \theta(\bar{x}, \bar{x}'))\ \Pi\ \Big(\exists \bar{x}'\, (\theta(\bar{x}, \bar{x}')) \wedge \neg \mathit{guard}\Big).$$

The sequent $\varphi \to \exists \bar{x}'\, \theta(\bar{x}, \bar{x}')$ is valid because, by definition of $\theta$, the sequent $\varphi \to \theta(\bar{x}, \bar{x})$ is valid. Also, $\Big((\exists \bar{x}'\, \theta(\bar{x}, \bar{x}')) \wedge \neg \mathit{guard}\Big) \to \psi$ is valid because $\varphi\ \Pi\ \psi$ is. So the consequence rule allows us to infer $\varphi\ \Pi\ \psi$, as required. □

**Corollary 4** *The theorem holds if the strongest postcondition is replaced with the profile in the definition of expressivity, provided all conjunctions of assertions are assertions.*

**Proof** Recall that $\mathrm{Post}_\Pi$ is the global relation

$$(\exists \bar{x})[\varphi(\bar{x}) \wedge \mathrm{Profile}_\Pi(\bar{x}, \bar{v})].$$

and take into account that conjunctions and existential quantifications of assertions are assertions. □

**Q:** It is strange that you fixed vocabularies $\Upsilon_1 \subseteq \Upsilon_2$ first and considered logics second. As a result the logics are somehow bound by the fixed vocabularies.

**A:** This is easy to generalize. Let $\Upsilon_2$ be the vocabulary of all function and relation symbols. Some of these relation symbols (or *predicates*) may be used in assertions, but not in programs. Call such predicates *auxiliary*. Let $\Upsilon_1$ be $\Upsilon_2$ minus the auxiliary symbols.

**Q:** These global $\Upsilon_1, \Upsilon_2$ are not restrictive, but $\Upsilon_2$ structures are ugly.

**A:** Let $K$ be a class of structures of any vocabulary $\Upsilon \subseteq \Upsilon_2$ and consider all programs in the vocabulary $\Upsilon \cap \Upsilon_1$. The generalized Cook's theorem remains true.

In the rest of the conversation, let us restrict attention to the case of global $\Upsilon_1 \subseteq \Upsilon_2$.

# 7 Weak Profile Expressive Logics

**Q:** I began this conversation by asking about your claim, in [BG 1987], to remove the need for expressivity hypotheses in Cook's theorem. But the general version of the theorem that you just showed me still assumes expressivity. Are you going to claim that some logics make this hypothesis unnecessary?

**A:** Exactly. Some logics are expressive for all structures. Let's introduce a name for this property of logics.

**Definition 5** $L$ is *profile-expressive* if, for every program $\Pi$ in the vocabulary $\Upsilon_1$, the global relation $\text{Profile}_\Pi$ is expressible as an $L$ assertion.

With this definition, the following proposition is immediate from the last corollary. Assume that $L$ satisfies our general requirements (1) through (5) and (in order to reduce postconditions to profiles) that the conjunction of any two assertions is an assertion.

**Proposition 6** *If $L$ is profile-expressive then, for every vocabulary $\Upsilon$ and every class $K$ of $\Upsilon$ structures, the proof system $\mathcal{H}_L(K)$ is complete for $K$ validity.*

**Q:** Show me one profile-expressive logic.

**A:** OK. We will consider the case when $\Upsilon_1 = \Upsilon_2$. Let EL be the existential fragment of first-order logic. Universal quantification is forbidden in EL. To avoid sneaking in universal quantification by means of negation and existential quantification, we assume that the only propositional connectives of EL are negation, conjunction and disjunction, and that negation can be applied only to atomic formulas.

**Q:** I do not believe that EL is profile-expressive.

**A:** Right. We extend EL with the transitive closure operator TC [Immerman 1998]. The resulting logic is EL+TC.

**Q:** How does TC work?

**A:** Semantically, given a binary global relation $\rho(x, y)$ of vocabulary $\Upsilon$, TC produces a new binary global relation $\rho^*(x, y)$ of vocabulary $\Upsilon$ such that, on every $\Upsilon$ structure $A$, $\rho^*$ is the transitive closure of $\rho$. (Here and in what follows, "transitive closure" means what is often called the reflexive transitive closure. That is, $\rho^*(x, x)$ holds for all $x$.) Actually, TC is more general: $x$ and $y$ could be tuples of

variables of the same length $k$, so that $\rho$ and $\rho^*$ are $2k$-ary (or binary on $k$-tuples of elements rather than on single elements).

Syntactically, TC produces predicate expressions from formulas. By a predicate expression we mean an expression that can be used syntactically as though it were a predicate symbol; semantically it should therefore be interpreted as a predicate (with possibly some built-in free variables). Let $\varphi(\bar{x}, \bar{y})$ be a formula where, for some $k$, $\bar{x}$ and $\bar{y}$ are $k$-tuples of free variables of $\varphi$. It is supposed that these $2k$ variables are all distinct. $\varphi$ may have additional free variables $z_1, \ldots, z_l$.

$$\mathrm{TC}_{\bar{x},\bar{y}}\varphi(\bar{x}, \bar{y})$$

is a predicate expression where $\bar{x}, \bar{y}$ are bound while $z_1, \ldots, z_l$ are free. If $\bar{s}, \bar{t}$ are $k$ tuples of terms, then

$$[\mathrm{TC}_{\bar{x},\bar{y}}\varphi(\bar{x}, \bar{y})](\bar{s}, \bar{t})$$

is a formula. Semantically, if $\varphi$ (with fixed values for $z_1, \ldots, z_l$) defines a binary relation $\rho$ on $k$-tuples, then $\mathrm{TC}_{\bar{x},\bar{y}}\varphi(\bar{x}, \bar{y})$ denotes the transitive closure $\rho^*$ of that relation.

EL+TC can serve as the logic $L$ of the generalized Cook's theorem.

**Q:** What are the guards and assertions in EL+TC?

**A:** Guards are arbitrary quantifier-free formulas, and assertions are arbitrary formulas. In fact, for all the logics we consider here, assertions are the same as formulas. Sequents are, of course, arbitrary implications between assertions.

EL+TC is profile-expressive. In other words, for every `while` program $\Pi(\bar{v})$, there is an EL+TC formula $\pi(\bar{x}, \bar{y})$ such that, for every $\Upsilon$ structure $A$ and all states $\bar{x}, \bar{y}$ of $\Pi$ over $A$, we have

$$(\bar{x}, \bar{y}) \in \mathrm{Profile}_{\Pi}^{A} \iff A \models \pi(\bar{x}, \bar{y})$$

The proof proceeds by induction on $\Pi$. The only case where we need TC is the case when $\Pi$ is `while` $guard(\bar{v})$ `do` $\Pi_1(\bar{v})$. We assume, without loss of generality, that the variables of $\Pi$ are exactly the variables of *guard*. By the induction hypothesis, $\mathrm{Profile}_{\Pi_1}$ is expressible by an EL+TC formula $\pi_1(\bar{x}, \bar{y})$. The $\mathrm{Profile}_{\Pi}$ is expressible by the formula

$$\Big[\mathrm{TC}_{\bar{x},\bar{y}}(\pi_1(\bar{x}, \bar{y}) \wedge guard(\bar{x}))\Big](\bar{x}, \bar{y}) \wedge \neg guard(\bar{y})$$

**Q:** I guess any logic which is richer that EL+TC is also profile expressive.

**A:** One should be a little careful. A richer logic may have more assertions to serve as profiles, but it may also have more guards and thus more programs in need of profiles.

**Proposition 7** *Assume that every assertion of a profile-expressive logic $L_1$ is also an assertion of logic $L_2$, with the same meaning. Assume further that every $L_2$ guard is also an $L_1$ guard. Then $L_2$ is profile expressive.*

For example, the extension FO+TC of first-order logic with the transitive closure operator is profile expressive, provided of course that we still take the quantifier-free formulas as the guards.

**Q:** Since profile-expressivity is inherited by richer logics, one may be interested in minimal profile-expressible logics. I guess you can get one such logic by extending EL with the profiles of all `while` programs. This is ugly of course. Is there a nicer proper sublogic of EL+TC?

**A:** Instead of the transitive closure operator TC, we can use a more restrictive *deterministic transitive closure operator* DTC [Immerman 1998]. Given a relation $R(x, y)$ consider an auxiliary relation

$$R_0(x, y) \rightleftharpoons R(x, y) \wedge \neg(\exists z)[R(x, z) \wedge z \neq y]$$

The deterministic transitive closure of $R$ is the transitive closure of $R_0$. Here $x, y$ could be tuples of variables of the same length. It is easy to see that EL+DTC is profile expressive. The point is that DTC works just like TC as long as the relation to which it is applied is single-valued.

In fact, one can put additional restrictions on EL+TC and preserve its profile-expressiveness. If you write out the EL+TC formulas expressing profiles of assignments, compositions, and conditional statements (or if you look them up in [BG 1987, page 26]), you'll find that existential quantification is used only in contexts $\exists x\varphi(x)$ where there is at most one $x$ satisfying $\varphi(x)$. So you can restrict the logic by allowing $\exists$ only when it coincides with $\exists!$, the unique existential quantifier. (Unlike the restriction from TC to DTC, this restriction on existential quantification is not very familiar, but it is not entirely unheard of; see for example [Kock and Reyes, 1977, page 292].)

**Q:** This is confusing. It seems that, before using an existential quantifier, you have to somehow determine that there cannot be more than one witness. And that could depend on the structure in which the formula is being interpreted.

**A:** This sort of logic is indeed unusual. One can define the notions of formula and proof by a simultaneous induction, requiring a proof of uniqueness before an existential quantifier can be used.

In a similar vein, the expressions for profiles use disjunction only in contexts where at most one of the disjuncts could be true, i.e., where inclusive and exclusive "or" coincide. Here one can get along without any complicated notion of proving

exclusivity, because in fact disjunction is used only in contexts of the form $(\alpha \wedge guard) \vee (\beta \wedge \neg guard)$, where the syntactic form makes it obvious that the disjuncts are mutually exclusive.

# 8 Existential Fixed-Point Logic

**Q:** Coming back again to the beginning of our conversation, I guess that existential fixed-point logic is also profile-expressive. But please remind me what existential fixed-point logic is.

**A:** Let us start at the beginning. Consider a transformation $\Gamma$ sending arbitrary subsets $X$ of a given set $S$ to subsets $\Gamma(X)$ of $S$. Assume that $\Gamma$ is *monotone*, that is $\Gamma(X) \subseteq \Gamma(Y)$ whenever $X \subseteq Y$. Define the (transfinite) iteration of $\Gamma$ by

$$\begin{aligned} \Gamma^0 &= \emptyset, \\ \Gamma^{\alpha+1} &= \Gamma(\Gamma^\alpha), \\ \Gamma^\alpha &= \bigcup_{\beta<\alpha} \Gamma^\beta \quad \text{if } \alpha \text{ is a limit ordinal} \end{aligned}$$

Continue the induction up to the first ordinal $\alpha$ with $\Gamma^\alpha = \Gamma^{\alpha+1}$; such an $\alpha$ exists because this possibly transfinite sequence is increasing but bounded by $S$. The set $\Gamma^\infty \rightleftharpoons \Gamma^\alpha(X)$ is the least fixed point of $\Gamma$, so that $\Gamma(\Gamma^\infty) = \Gamma^\infty$ and $\Gamma^\infty \subseteq X$ for every $X$ with $\Gamma(X) = X$. (See [Moschovakis 1974, Section 1A].)

**Q:** What has this to do with logic?

**A:** Let $\rho(v)$ be a global relation of vocabulary $\Upsilon \cup \{X\}$ where $X$ is a unary relation. For every $\Upsilon$ structure $A$, we have a transformation

$$\Gamma^A(X) \rightleftharpoons \{v : \langle A, X\rangle \models \rho^A(v)\}$$

Suppose that this transformation is monotone for every $A$ and let $\mathrm{FP}(\rho)$ be the unary global relation such that, for every $\Upsilon$ structure $A$, $(\mathrm{FP}(\rho))^A$ is the least fixed point of $\Gamma^A$. This generalizes to the case when $v$ is a $k$-tuple of variables and $X$ is a $k$-ary relation.

**Q:** How can you ensure that $\Gamma^A$ is monotone for *every* $A$?

**A:** There are syntactical means to guarantee monotonicity. For example, $\rho$ can be given by an EL or FO formula where $X$ has only positive occurrences.

But this was a good question. Let $\Upsilon_1$ be the total vocabulary of the current EL formulas. Predicates in $\Upsilon_1$ will be called *negatable*. In addition, we introduce

auxiliary *positive* predicates. Let $\Upsilon_2$ be the extension of $\Upsilon_1$ with the positive predicates. We modify the definition of existential logic EL. In the new EL, negation may be applied only to atomic formulas whose predicate symbol is negatable.

**Q:** I am happy to see a case where $\Upsilon_1 \neq \Upsilon_2$. But what is the point of introducing positive predicates? To ensure monotonicity?

**A:** There are two points, a technical one and an intuitive one. The technical point is to ensure monotonicity by syntactic means; you will see how it works exactly.

The intuitive point is to model the idea of incomplete information in databases. Suppose, for example, that we have a database containing records of the form (person, book), intended to mean that this person owns (a copy of) this book. This database might be (intended to be) complete in the sense that it lists all the books owned by these people. In that case, if a particular pair, say (Quisani, Bible), is absent from the database, we would conclude that you don't own a Bible. On the other hand, the database might be only partial, listing some but not necessarily all of the books the people own. In that case, it could not be used to infer non-ownership. We would model the first situation by making ownership a negatable predicate and the second by making it a positive predicate. In the second case, this would exclude from our language the unverifiable statements of non-ownership.

By the way, the definition of a homomorphism between structures in [BG 1987] also distinguishes between negatable and positive predicates. For a negatable predicate $P$, homomorphisms $h : A \to B$ must satisfy $P^A(a) \iff P^B(h(a))$, but for a positive predicate one requires only the forward implication. In database example, this means that if $B$ is obtained by adding some new records (like (Quisani, Bible)) to a database $A$, then the identity function from $A$ to $B$ is a homomorphism in the positive situation, where the new record is additional information entirely compatible with what was in $A$, but it is not a homomorphism in the negatable situation, where the new information contradicts what was there before.

This database view of positivity and negatability will not be essential for our present discussion; let us return to the definition of existential fixed-point logic.

Syntactically, FP produces predicate expressions from formulas. Let $\varphi(X, \bar{x})$ be a formula where $X$ is a $k$-ary positive predicate and $\bar{x}$ is a $k$-tuple of variables. $\varphi$ may have additional variables $y_1, y_2, \ldots, y_l$.

$$\mathrm{FP}_{X,\bar{x}}\varphi(X, \bar{x})$$

is a predicate expression where the variables $\bar{x}$ are bound and the predicate $X$ (essentially, a second-order variable) is bound as well. Variables $y_1, y_2, \ldots, y_l$ are free in the predicate expression. If $\bar{t}$ is a $k$-tuple of terms, then

$$[\mathrm{FP}_{X,\bar{x}}\varphi(X, \bar{x})](\bar{t})$$

is a formula.

This gives rise to the extension EL+FP of existential logic with the least fixed-point operator.

**Q:** I think I understand the syntax and semantics of EL+FP. Coming back to the generalized Cook's theorem, what are the guards and assertions in EL+FP.

**A:** Assertions are arbitrary EL+FP formulas. Guards are arbitrary quantifier-free EL+FP formulas which do not use positive predicates. Sequents are, as always, arbitrary implications between assertions.

**Q:** I suppose you'll claim that EL+FP is profile-expressive.

**A:** Yes, and this will follow immediately from Proposition 7 once we know that EL+FP includes EF+TC. And this inclusion is just a matter of expressing transitive closures by means of least fixed points. This is easy; the predicate expression $\mathrm{TC}_{\bar{x},\bar{y}}\varphi(\bar{x},\bar{y})$ is synonymous with $\mathrm{FP}_{X,\bar{x},\bar{y}}\psi(X,\bar{x},\bar{y})$ where $\psi$ is the formula

$$\bar{x} = \bar{y} \vee \exists \bar{w}(\varphi(\bar{x},\bar{w}) \wedge X(\bar{w},\bar{y})).$$

In fact, EL+FP remains profile-expressive even if we enlarge the programming language to allow parameterless procedures (going from Section 2 to Section 3 of [Apt 1981]). This is what we showed in Section 2 of [BG 1987]. It can be deduced that Cook's theorem holds for this class of programs and for EL+FP assertions, with no need for any expressivity hypothesis. And this is probably what we had in mind when we wrote the introduction to [BG 1987].

By Proposition 7, FO+FP is also profile-expressive for the programming language of `while` programs plus parameterless procedures. That result could also be inferred from the information in [Harel and Peleg 1984].

**Q:** What's involved in proving Cook's theorem for EL+FP assertions and for programs in the language of `while` programs plus parameterless procedures?

**A:** Start with the proof for the case of FO assertions, under the hypothesis of expressivity, as presented for example in [Apt 1981, Section 3.8]. Look at all the constructions of assertions used in that proof; for example, one forms the conjunction of an assertion with some equations. Check that all these constructions are also available in EL+FP. So the proof works for EL+FP, under the same expressivity hypothesis. Finally, eliminate the expressivity hypothesis by using the profile-expressivity of EL+FP plus the expression, from the proof of Corollary 4, of strongest postconditions in terms of profiles.

# A Some Decision Problems Related to the Existential Fixed Point Logic

We discuss here several decision problems connected with existential fixed-point logic. Two of these problems concern existential fixed-point sequents; these were defined above as implications of the form $\varphi \to \psi$, where $\varphi$ and $\psi$ are EL+FP formulas. Recall that sequents are always interpreted as if they were preceded by universal quantifiers on all their free variables. Thus sequents can contain, in addition to all the constructs of EL+FP, implication and (implicitly) universal quantification, but only at the outermost level and not nested.

Notice that EL+FP sequents are the sentences whose validity is relevant to $\mathcal{H}_{EL+FP}$(all structures) via the Consequence Rule.

We shall consider the following three decision problems:

**EL+FP formula validity** Given a formula of EL+FP, decide whether it is logically valid.

**EL+FP sequent validity** Given an EL+FP sequent, decide whether it is logically valid.

**EL+FP sequent consequence** Given an EL+FP sequent C and a finite set $\mathcal{P}$ of EL+FP sequents, decide whether the conclusion $C$ is a logical consequence of the premises $\mathcal{P}$.

Here logical validity and logical consequence refer to a semantics allowing arbitrary structures, finite or infinite.

As discussed above, it is the second of these problems, EL+FP sequent validity, that is directly relevant to Hoare logic with EL+FP pre- and post-conditions and with arbitrary structures under consideration. The third problem, EL+FP sequent consequence, becomes relevant if, instead of considering arbitrary structures, we wish to limit attention to the models of some set $\mathcal{P}$ of EL+FP sequents. This includes, for example, the case where we are interested only in the standard model of arithmetic, since this model can be specified up to isomorphism by such a set $\mathcal{P}$ (which we shall explicitly exhibit in one of the proofs below).

The complexity of EL+FP formula validity was already worked out in [BG 1987, Theorem 6]; it is a complete recursively enumerable ($\Sigma_1^0$) set. The following two theorems give the corresponding complexity results, $\Pi_2^0$ and $\Pi_1^1$, for the other two decision problems. We treat consequence before validity because it is a bit easier.

**Theorem 8** *EL+FP sequent consequence is complete* $\Pi_1^1$.

**Proof** It was shown in [BG 1987, Theorem 9] that the inductive definitions involved in the semantics of EL+FP always terminate in at most $\omega$ steps. It follows that every EL+FP formula is equivalent to a formula of the infinitary logic $L_{\omega_1,\omega}$ that allows countable conjunctions and disjunctions. This is proved by induction on formulas, the only non-trivial case being $\mathrm{FP}_{P,\bar{x}}\varphi(P,\bar{x},\bar{y})$. Once the formula $\varphi(P,\bar{x},\bar{y})$ has been expressed in $L_{\omega_1,\omega}$, the finite stages of the inductive definition $\Gamma$ defined by $\varphi$ can also be expressed in $L_{\omega_1,\omega}$:

$$\bar{x} \in \Gamma^0 \iff \texttt{false}, \qquad \bar{x} \in \Gamma^{n+1} \iff \varphi(\Gamma^n, \bar{x}, \bar{y}),$$

And then stage $\omega$ is expressible by an infinite disjunction

$$\bar{x} \in \Gamma^\omega \iff \bigvee_{n\in\omega} \bar{x} \in \Gamma^n.$$

Once we have translated all EL+FP formulas into $L_{\omega_1,\omega}$, we can obviously do the same for EL+FP sequents. Then we can apply the downward Löwenheim-Skolem theorem for $L_{\omega_1,\omega}$, given in [Keisler 1971, Chapter 5, Problem 1], to conclude that the decision problem EL+FP sequent consequence is unchanged if, instead of considering arbitrary structures, we consider only countable ones.

Therefore, we may confine our attention to structures whose domain is either the set $N$ of natural numbers or a finite initial segment of $N$. For such structures, the satisfaction relation for EL+FP formulas is clearly (uniformly) recursively enumerable relative to the basic functions and relations of the structure. (This means that there is a single algorithm for enumerating the pairs (formula, values for variables) that are satisfied; the algorithm uses the basic functions and relations of the structure as oracles. The construction of such an algorithm is by induction on formulas.) That $C$ is a logical consequence of $\mathcal{P}$ is expressible as "for all relations and functions on $N$ (to interpret the symbols of the vocabulary of $C$ and $\mathcal{P}$), if all the sentences in $\mathcal{P}$ are satisfied (by the empty assignment), then so is $C$," and this is clearly $\Pi^1_1$. (This argument would have worked even if, instead of knowing that satisfaction is recursively enumerable, we merely knew that it is arithmetical, or even just hyperarithmetical.)

It remains to show that the decision problem EL+FP sequent consequence is $\Pi^1_1$-hard. The main ingredient in this proof is the observation that the standard model of arithmetic, together with any finitely many primitive recursive functions, can be characterized up to isomorphism by a finite set of EL+FP sequents. The following sequents characterize $(N, 0, S)$ up to isomorphism, where $S$ is the successor operation $S(x) = x + 1$.

$$0 = S(x) \to \texttt{false}$$
$$S(x) = S(y) \to x = y$$
$$\texttt{true} \to [\mathrm{FP}_{P,y}\, y = 0 \vee \exists z\, (P(z) \wedge y = S(z))](x).$$

To include finitely many primitive recursive functions in the characterization, simply add the defining equations of the desired functions as well as any auxiliary functions used in defining them. (Officially, these equations should be written as sequents by prefixing "$\texttt{true} \rightarrow$.")

Now any $\Pi^1_1$ predicate $Q$ of natural numbers can be expressed in the form

$$Q(m) \iff (\forall f : N \to N)(\exists n)\, X(m, f|(n))$$

where $f|(n)$ codes $\langle f(0), \ldots, f(n-1)\rangle$ in some standard, primitive recursive, coding scheme, and where $X$ is primitive recursive. Let $\mathcal{P}$ consist of the sequents characterizing the standard model of arithmetic with $X$ and the coding scheme. Then $Q(m)$ holds if and only if the following sequent is a logical consequence of $\mathcal{P}$; it uses a new function symbol $g$ intended to be $f|$.

$$\texttt{true} \rightarrow \exists n\, X(\bar{m}, g(n)) \vee \exists n\, g(S(n)) \neq g(n)^\frown(g(S(n)))_n \vee g(0) \neq \langle\rangle.$$

Here $\bar{m}$ means the standard numeral for $m$, obtained by applying $S$ to 0 $m$ times. The $^\frown$ and $(-)_n$ refer to the concatenation and component extraction operations of the coding scheme, and $\langle\rangle$ means the code for the empty sequence. (We ordinarily take equality to be a negatable predicate, so the $\neq$'s in the sequent above are legitimate. But the argument would work even if equality were positive, since we could include in $\mathcal{P}$ the defining equations for the primitive recursive characteristic function of $\neq$.) In the sequent exhibited above, the second and third disjuncts simply say that $g$ is not of the form $f|$ for any $f$; thus, the sequent is a consequence of $\mathcal{P}$ if and only if, in every isomorphic copy of $N$ with the appropriate primitive recursive functions, and for every $g$ of the form $f|$, some $n$ satisfies $X(\bar{m}, f|(n))$. But this is precisely what $Q(m)$ means.

Thus, an arbitrary $\Pi^1_1$ predicate $Q$ is Turing reducible (in fact, one-one reducible) to EL+FP sequent consequence. $\Box$

**Theorem 9** *EL+FP sequent validity is complete* $\Pi^0_2$.

**Proof** We begin by proving that the decision problem EL+FP sequent validity is $\Pi^0_2$-hard. It will be convenient to abbreviate as $\mathrm{Num}(x)$ the following EL+FP formula:

$$[\mathrm{FP}_{P,y}\, y = 0 \vee \exists z\, (P(z) \wedge y = S(z))](x).$$

The idea is that, in any structure for a vocabulary containing 0 and $S$, if the interpretation of $S$ is a one-to-one function whose range doesn't contain the interpretation of 0, then Num defines a copy of $(N, 0, S)$ within that structure.

Consider now an arbitrary $\Pi^0_2$ predicate $Q$ of natural numbers. It can be written as

$$Q(m) \iff \forall p\, \exists q\, X(m, p, q)$$

for some primitive recursive $X$. Then, for any $m$, $Q(m)$ holds if and only if the following EL+FP sequent is logically valid.

$$\begin{aligned}\forall p\,[\mathrm{Num}(p) \;\rightarrow\; & \exists q\,(\mathrm{Num}(q) \wedge X(\bar{m}, p, q)) \vee \\ & \exists x\,(\mathrm{Num}(x) \wedge 0 = S(x)) \vee \\ & \exists x \exists y\,(\mathrm{Num}(x) \wedge \mathrm{Num}(y) \wedge S(x) = S(y) \wedge x \neq y) \vee \\ & \ldots].\end{aligned}$$

Here as before $\bar{m}$ is the standard numeral for $m$. The final ... in the sequent represents a list of disjuncts saying that the defining equation of some primitive recursive function is violated by some element satisfying Num; we include one such disjunct for every step in the construction of (the characteristic function of) $X$ as a primitive recursive function. Thus, all the disjuncts except the first serve to make the sequent true in all "bad" models, those where Num is not a copy of $N$ or where a primitive recursive function is interpreted incorrectly. It follows that the sequent is valid if and only if

$$\forall p\,[\mathrm{Num}(p) \rightarrow \exists q\,(\mathrm{Num}(q) \wedge X(\bar{m}, p, q))]$$

holds in all good structures, i.e., if and only if $\forall p \exists q\, X(m, p, q)$ holds in $N$.

We thus have a Turing reduction (in fact a one-one reduction) from the predicate $Q$ to EL+FP sequent validity. This completes the proof that EL+FP sequent validity is $\Pi^0_2$-hard; it remains to prove that it is in $\Pi^0_2$.

Consider an arbitrary EL+FP sequent, $\varphi \rightarrow \psi$. As far as validity of this sequent is concerned, nothing changes if we replace all free variables by new, distinct constants throughout $\varphi \rightarrow \psi$. So, without loss of generality, neither $\varphi$ nor $\psi$ has any free variables.

Also, by [BG 1987, Theorem 5], we can put $\psi$ in strict $\forall^1_1$ form, i.e., a block of universal second-order quantifiers on relations (not functions) followed by an existential first-order formula. Then, without affecting validity, we can delete the universal second-order quantifiers. So, without loss of generality, $\psi$ is an existential first-order formula.

Now consider what it would mean for our sequent $\varphi \rightarrow \psi$ *not* to be valid. We would have a model $A$ satisfying $\varphi \wedge \neg\psi$. Temporarily concentrate on the fact that $A$ satisfies $\varphi$. Proceeding as in the proof of [BG 1987, Theorem 7], we obtain finitely many quantifier-free formulas, called $\neg\theta(\mathbf{b}_i)$ in that proof, such that these formulas are not simultaneously satisfiable in the propositional sense. These $\neg\theta(\mathbf{b}_i)$ were obtained using certain facts from the quantifier-free diagram of $A$ (to eliminate atomic subformulas of $\theta$ that start with predicate symbols from the vocabulary of $A$, leaving only predicate symbols quantified in the strict $\forall^1_1$ form of $\varphi$). The truth of these finitely many facts in $A$ thus ensures the truth of $\varphi$. (In [BG 1987], this

led to a finite substructure of $A$ in which all these facts were already present; for the present discussion we need the facts themselves, not the substructure.)

Whether a particular finite set of quantifier-free sentences ensures (in the sense above) the truth of $\varphi$ is easily seen to be recursively verifiable, i.e., "ensures" is a recursively enumerable relation. (The verification consists of trying all possible $\mathbf{b}_i$.)

So far, we have seen that, if $\varphi \to \psi$ isn't valid, then there is a finite set $\Phi$ of facts, ensuring $\varphi$, such that some $A$ satisfies $\Phi$ but not the existential sentence $\psi$.

Starting with the names mentioned in $\Phi$ (or with one constant symbol if there are no such names), try to build an Herbrand model for $\Phi$ and $\neg\psi$. This amounts to assigning truth values, consistent with $\Phi$, to all atomic formulas built from those names. There is a model of $\Phi$ and $\neg\psi$ if and only if (there is an Herbrand model if and only if) $\Phi$ and all instances of the universal sentence $\neg\psi$ are propositionally consistent. By compactness, this consistency amounts to consistency of every finite subset. Thus, $\varphi \to \psi$ is not valid if and only if there exists a finite set $\Phi$ that ensures $\varphi$, such that every finite set of instances of $\neg\psi$ is propositionally consistent with $\Phi$.

This criterion for non-validity is $\Sigma^0_2$. It begins with an unbounded existential quantifier "there exists $\Phi$" which is essentially over numbers since finite sets $\Phi$ of formulas can be replaced by their Gödel numbers. It continues with another existential number quantifier in "ensures," which we saw is recursively enumerable. Then there is a universal number quantifier "every finite set of instances of $\neg\psi$," and everything else is recursive.

Finally, taking negations, we find that validity of $\varphi \to \psi$ is $\Pi^0_2$, as required. □

# References


**Apt 1981** Krzysztof R. Apt, "Ten years of Hoare's logic: A survey — Part 1", ACM Transactions on Programming Languages and Systems 3:4 (October 1981), 431–483.

**BG 1987** Andreas Blass and Yuri Gurevich, "Existential fixed-point logic", In "Logic and Complexity" (ed. E. Börger), Springer Lecture Notes in Computer Science 270 (1987), 20–36.

**Cook 1978** S. A. Cook, "Soundness and completeness of an axiom system for program verification", SIAM Journal of Computing 7:1 (1978), 70–90.

**Ebbinghaus and Flum 1995** Heinz-Dieter Ebbinghaus and Jörg Flum, "Finite Model Theory", Springer-Verlag 1995.

**Enderton 1977** Herbert B. Enderton, "Elements of recursion theory," in J. Barwise, editor, "Handbook of Mathematical Logic," Studies in Logic and Foundations of Mathematics 90, North-Holland (1977), 527–566.

**Gurevich 1984** Yuri Gurevich, "Toward logic tailored for computational complexity", in M. Richter et al., editors, "Computation and Proof Theory" (Logic Colloquium 1983), Springer Lecture Notes in Math. 1104 (1984), 175–216.

**Harel and Peleg 1984** David Harel and David Peleg, "On static logics, dynamic logics, and complexity classes", Information and Control 60 (1987), no. 1–3, pp. 86–102.

**Hoare 1969** C. A. R. Hoare, "An axiomatic basis for computer programming", Communications of ACM 12:10 (October 1969), 576–580, 583.

**Immerman 1998** Neil Immerman, " Descriptive Complexity" (Graduate Texts in Computer Science), Springer-Verlag, 1998.

**Keisler 1971** H. Jerome Keisler, "Model Theory for Infinitary Logic," Studies in Logic and the Foundations of Mathematics, vol. 62, North-Holland, 1971.

**Kock and Reyes 1977** Anders Kock and Gonzalo Reyes, "Doctrines in categorical logic," in J. Barwise, editor, "Handbook of Mathematical Logic," Studies in Logic and Foundations of Mathematics 90, North-Holland (1977), 283–313.

**Moschovakis 1974** Yiannis Moschovakis, "Elementary Induction on Abstract Structures," Studies in Logic and the Foundations of Mathematics 77, North-Holland 1974.

**Rogers 1967** Hartley Rogers, "Theory of Recursive Functions and Effective Computability", McGraw-Hill, New York, 1967.