Completeness for Ancestral Logic via a Computationally-Meaningful Semantics

Liron Cohen

Cornell University

Abstract. First-order logic (FOL) is evidently insufficient for the many applications of logic in computer science, mainly due to its inability to provide inductive definitions. Therefore, only an extension of FOL which allows finitary inductive definitions can be used as a framework for automated reasoning. The minimal logic that is suitable for this goal is Ancestral Logic (AL), which is an extension of FOL by a transitive closure operator. In order for AL to be able to serve as a reasonable (and better) substitute to the use of FOL in computer science, it is crucial to develop adequate, user-friendly proof systems for it. While the expressiveness of AL renders any effective proof system for it incomplete with respect to the standard semantics, there are useful approximations. In this paper we show that such a Gentzen-style approximation is both sound and complete with respect to a natural, computationally-meaningful Henkin-style semantics for AL.

1 Introduction

In [8] it was forcefully argued that logic plays a central role in computer science. Evidence for this claim was provided by listing a variety of applications of logic in different areas in computer science, such as descriptive complexity, database query languages, program verification and more. However, when examining this list of applications, it turns out that first-order logic (FOL), which is the logic usually associated with 'logic', does not actually suffice for any of the mentioned applications.[1] Evidently, extensions of FOL are needed in almost all of the examples given in [8]:

- The characterization of complexity classes which is done in descriptive complexity always uses logics that are more expressive than FOL, such as second-order logic (SOL), or logics which are intermediate between FOL and SOL.
- Verification of programs involve *inductive arguments* which are not a part of the logical machinery of FOL.
- [8] only mentioned query languages which are directly based on FOL, like SQL. However, its poor expressive power is the reason that the SQL 3 (1999)

[1] Actually, at this point we are only referring to the formal *languages* used in the applications, ignoring (for the time being) other essential components of the notion of a 'logic', like the corresponding consequence relation.

standard added a WITH RECURSIVE construct which allows transitive closures to be computed inside the query processor, and by now such a construct is implemented also in IBM DB2, Microsoft SQL Server, and PostgreSQL. Datalog too implements transitive closure computations.

– Not only do type theories obviously go beyond FOL, but even their presentation and description cannot be done in FOL, since their introduction makes a massive use of *inductive definitions* of typing judgments.
– Most applications of model-checking rely on the notion of *reachability,* which is not first-order definable. It is noted in [18] that "In all interesting applications of model-checking, reachability properties have to be checked, which are not expressible in the FOL-signature of labeled graphs (transition systems)".
– A crucial notion for reasoning about knowledge is that of *common knowledge.* This notion is *inductively defined* in terms of the basic knowledge operators. However, this definition is not expressible in FOL and so is usually introduced by brute force.

All these examples (as well as many others) reveal that what FOL is lacking is the ability to provide inductive definitions. More particularly, the notion of the transitive closure of a given binary relation seems to be the key necessary component which is not expressible in FOL. In fact, because of this inability, FOL cannot even serve as its own meta-logic, since all its basic syntactic categories (such as terms, formulas, and formal derivations of formulas) are introduced via inductive definitions. Hence, only some extension of FOL which allows finitary inductive definitions ([5]) can be used as a framework for automated reasoning. While SOL clearly enjoys this property, it does not seem satisfactory that dealing with basic inductive definitions requires using the strong notions involved in SOL, such as quantifying over all subsets of infinite sets. Full SOL also has many disadvantages, among which are its doubtful semantics (as it is based on debatable ontological commitments), and what is more, the fact that it is difficult to deal with from a proof-theoretical point of view.

In [1,2] it was shown that the *minimal* framework that can be used for the above mentioned goal is Ancestral Logic, AL (which is also known in the literature as TC-logic). This is the logic obtained from FOL by the addition of a transitive closure operator. Although several other logics which are intermediate between FOL and SOL have been suggested in the literature (such as: weak second-order logic, ω-logic, logics with a "cardinality quantifier", logics with Henkin quantifiers, etc.), we strongly believe that AL should be taken as the basic logic which underlies most applications of logic in computer science. Its advantages include: being useful in the finite cases[2], having intuitive formal

[2] A great deal of attention has been given to AL in the area of finite model theory, and in related areas of computer science, like complexity classes (see, e.g., [4]). However, not much has been done so far about it in the context of *arbitrary* structures, or from a proof theoretical point-of-view.

proof systems, and entering very naturally in computer science applications[3]. Another important advantage of AL is the simplicity of the transitive closure notion. Anyone, even with no mathematical background whatsoever, can easily grasp the concept of the ancestor of a given person (or, in other words, the idea of the transitive closure of a certain binary relation).

In order for AL to be used as the foundational logic in computer science applications, its theory must first be developed to the point it can serve as a reasonable (and in many cases, better) substitute for the use of FOL (or higher-order logics). Since our goal is to explore the use of this logic in such applications, the emphasis should be on the construction of adequate, user-friendly formal systems for AL. Due to the expressiveness of AL, there can be no sound and complete effective proof system for it (see, e.g., [15]). Instead, one should look for useful approximations (like in the case of SOL). In [1,2] a Gentzen-style proof system for AL was presented and its proof-theoretical properties were explored.[4] It was shown to be natural and effective, as well as sound with respect to the intended semantics. In this paper we provide further evidence for the usefulness of the system by proving that it is both sound and complete with respect to a generalized Henkin-style semantics.[5] This semantics for AL is based on the one used for the completeness proof for SOL given in [9].

The rest of this paper is organized as follows: In Section 2 the formal definition of the reflexive transitive closure operator and ancestral logic are given. Then, some of the most important model-theoretic properties of ancestral logic are presented. Section 3 provides a natural Gentzen-style system which is adequate for ancestral logic in the sense that it is sound with respect to the standard semantics, and captures the properties that govern the transitive closure operator. Section 4 contains the main result of the paper: a completeness theorem for the proof system for AL with respect to a natural Henkin-style semantics. Finally, in Section 5 we conclude with some remarks and ideas for further research.

2 The Language and its Semantics

The essential idea in embedding the general concept of the transitive closure operator into a logical framework is that one may treat a formula with two (distinct) free variables as a definition of a binary relation. Below is the formal definition of first-order logic augmented by a transitive closure operator, and its semantics. In this paper (following suggestions made in, e.g., [12,13,14]) we take the reflexive form of the transitive closure operator as the primitive notion. In

[3] To demonstrate one such application of AL in computer science, in [3] a constructive version of AL was shown to subsume Kleene algebra with tests [11] (as the reflexive transitive closure operator is essentially Kleene's star operator), while offering much more expressive power. This demonstrates that AL can serve as a natural programming logic for specifying, developing and reasoning about programs.

[4] In fact, [2] presented several proof systems for different variations of AL, and the connection between them was investigated.

[5] To be precise, we take here an equivalent variant of a system presented in [2].

[2] it was shown that the two forms of the operator, the reflexive one and the non-reflexive one, are equivalent in the presence of equality.

Throughout the paper we use the following standard notations:

- $Fv(\varphi)$ for the set of free variable in the formula φ.
- $v[x := a]$ for the x-variant of the assignment v which assigns a to x.
- $\varphi\left\{\frac{t_1}{x_1}, ..., \frac{t_n}{x_n}\right\}$ for the result of simultaneously substituting t_i for the free occurrences of x_i in φ $(i = 1, ..., n)$.

Definition 1. *Let σ be some first-order signature, and let $\mathcal{L}$ be the corresponding first-order language. The language $\mathcal{L}_{RTC}$ is obtained from $\mathcal{L}$ by the addition of the reflexive transitive closure operator (RTC), together with the following clause concerning the definition of a formula:*

- *$(RTC_{x,y}\varphi)(s,t)$ is a formula in $\mathcal{L}_{RTC}$ for any formula φ in $\mathcal{L}_{RTC}$, distinct variables x, y, and terms s, t.*

The free occurrences of x and y in φ become bound in this formula.

Note that φ in the above definition can be *any* formula in $\mathcal{L}_{RTC}$. That is, it may contain free variables other than x, y (treated as parameters), or it may not contain x, y at all. Also, φ can have a RTC-subformula, i.e. nesting of the RTC operator are allowed.

The intended meaning of a formula of the form $(RTC_{x,y}\varphi)(s,t)$ is the "infinite disjunction":

$$s = t \vee \varphi\{\frac{s}{x}, \frac{t}{y}\} \vee \exists w_1(\varphi\{\frac{s}{x}, \frac{w_1}{y}\}) \wedge \varphi\{\frac{w_1}{x}, \frac{t}{y}\}) \vee$$
$$\exists w_1 \exists w_2(\varphi\{\frac{s}{x}, \frac{w_1}{y}\} \wedge \varphi\{\frac{w_1}{x}, \frac{w_2}{y}\} \wedge \varphi\{\frac{w_2}{x}, \frac{t}{y}\}) \vee \dots$$

where $w_1, w_2, ...$ are all fresh variables.

Definition 2. *Let M be a structure for $\mathcal{L}_{RTC}$, and v an assignment in M. Ancestral logic (AL) is semantically defined as classical first-order logic, with the following additional clause concerning the satisfaction relation:*

- *The pair $\langle M, v\rangle$ is said to satisfy the formula $(RTC_{x,y}\varphi)(s,t)$ (denoted by $M, v \models (RTC_{x,y}\varphi)(s,t)$) if $v(s) = v(t)$, or there exist $a_0, ..., a_n \in D$ $(n > 0)$ such that $v(s) = a_0$, $v(t) = a_n$, and $M, v[x := a_i, y := a_{i+1}] \models \varphi$ for $0 \leq i \leq n-1$.*

A simple compactness argument shows that the reflexive transitive closure operator is in general not first-order definable. However, it is definable in second-order logic by the formula: $\forall X\left((Xs \wedge \forall x \forall y\left(\varphi(x,y) \wedge Xx \rightarrow Xy\right)) \rightarrow Xt\right)$. Therefore, ancestral logic is intermediate between first- and second-order logics. An important indication that the expressive power of ancestral logic captures a very significant and natural fragment of SOL is provided by the fact that AL is equivalent in its expressive power to several other logics between FOL and SOL that have been suggested and investigated in the literature (such as those mentioned in the introduction).

The natural numbers can be categorically characterized in AL using only equality, zero and the successor function (see [2]). This implies that the upward Löwenheim-Skolem theorem fails for AL, as well as the compactness theorem (see [15]). Moreover, if addition is added to the language, all recursive functions and relations are definable in AL (see [1]), and thus the set of valid formulas of AL in this language is not even arithmetical. Hence AL is inherently incomplete, i.e., any formal deductive system which is sound for AL is incomplete. Nevertheless, as we shall demonstrate, there are very natural formal approximations which are sound, and seem to encompass all forms of reasoning for this logic that are used in practice.

3 Formal Proof System for *AL*

As in the case of SOL, since there can be no sound and complete formal system for AL, one should instead look for useful approximations. Such approximations should be:

- natural and effective,
- sound with respect to the intended semantics,
- both sound and complete with respect to some natural generalization of the intended semantics.

Such equivalent Hilbert-style approximations were suggested already in [12,13,14]. Nevertheless, the use of Hilbert-type systems is impractical, since they are not suitable for mechanization. A better, computationally-oriented approach would be to explore Gentzen-style systems for AL. This was done in [2], and we here review the system and its main properties.

Definition 3. *Let G be a Gentzen-style system (see, e.g., [6]).*

- *A sequent s is said to be provable from a set of sequents S in G, denoted by $S \vdash_G s$, if there exists a derivation in G of s from S.*
- *A formula φ is said to be provable from a set of formulas T in G, denoted by $T \vdash_G \varphi$, if there is a derivation in G of $\Rightarrow \varphi$ from the set $\{\Rightarrow \psi \mid \psi \in T\}$.*

In what follows the letters Γ, Δ represent finite (possibly empty) multisets of formulas, φ, ψ arbitrary formulas, x, y, z, u, v variables, and r, s, t terms.

Let $\mathcal{LK}$ be the Gentzen-style system for classical first-order logic [6,17], including the substitution rule (though it was not a part of the original system).

Definition 4. *The system AL_G for $\mathcal{L}_{RTC}$ is defined by adding to $\mathcal{LK}$ the following axiom*:

$$\Gamma \Rightarrow \Delta, (RTC_{x,y}\varphi)\,(s, s) \tag{1}$$

and the following inference rules:

$$\frac{\Gamma \Rightarrow \Delta, (RTC_{x,y}\varphi)(s,r) \quad \Gamma \Rightarrow \Delta, \varphi\left\{\frac{r}{x}, \frac{t}{y}\right\}}{\Gamma \Rightarrow \Delta, (RTC_{x,y}\varphi)(s,t)} \tag{2}$$

$$\frac{\Gamma, \psi(x), \varphi(x,y) \Rightarrow \Delta, \psi\left\{\frac{y}{x}\right\}}{\Gamma, \psi\left\{\frac{s}{x}\right\}, (RTC_{x,y}\varphi)(s,t) \Rightarrow \Delta, \psi\left\{\frac{t}{x}\right\}} \tag{3}$$

In all the rules we assume that the terms which are substituted are free for substitution, and that no forbidden capturing occurs. In Rule (3) x *should not occur free in* Γ *and* Δ*, and* y *should not occur free in* Γ, Δ *and* ψ*.*

For languages with equality, the system $AL_G^{=}$ *is obtained from* AL_G *by the addition of standard equality axioms (see, e.g., [17]).*

Rule (3) is a generalized induction principle. It states that if t is a φ-descendant of s or equal to it, then if s has some hereditary property which is passed down from one object to another if they are φ-related, then t also has that property. In the case of arithmetic this rule captures the induction rule of Peano's Arithmetics PA (see [2]).[6]

The system AL_G is adequate for handling the RTC operator, in the sense that it is sound and it gives the RTC operator the intended meaning of the reflexive transitive closure operator. Furthermore, all fundamental rules concerning the RTC operator that have been suggested in the literature (as far as we know) are derivable in it. The Lemma below provides some examples.

Lemma 5. *The following rules are derivable in* AL_G*:*

$$\frac{\Gamma \Rightarrow \Delta, (RTC_{x,y}\varphi)(s,r) \quad \Gamma \Rightarrow \Delta, (RTC_{x,y}\varphi)(r,t)}{\Gamma \Rightarrow \Delta, (RTC_{x,y}\varphi)(s,t)} \tag{4}$$

$$\frac{\Gamma, \varphi \Rightarrow \Delta, \psi}{\Gamma, (RTC_{x,y}\varphi)(s,t) \Rightarrow \Delta, (RTC_{x,y}\psi)(s,t)} \tag{5}$$

$$\frac{(RTC_{x,y}\varphi)(s,t), \Gamma \Rightarrow \Delta}{(RTC_{u,v}(RTC_{x,y}\varphi)(u,v))(s,t), \Gamma \Rightarrow \Delta} \tag{6}$$

$$\frac{\Gamma \Rightarrow \Delta, (RTC_{x,y}\varphi)(s,t)}{\Gamma \Rightarrow \Delta, (RTC_{y,x}\varphi)(t,s)} \qquad \frac{(RTC_{x,y}\varphi)(s,t), \Gamma \Rightarrow \Delta}{(RTC_{y,x}\varphi)(t,s), \Gamma \Rightarrow \Delta} \tag{7}$$

In (5) x, y *should not occur free in* Γ *and* Δ*, and in* (6) u, v *should not occur free in* φ*.*

[6] In fact, it was shown in [2] that in the case of arithmetics the ordinal number of AL_G is ε_0, like in the case of PA.

4 Henkin-Style Completeness

Though AL_G cannot be complete for its intended semantics, it can be shown to be complete for a more liberal yet natural semantics, in the spirit of the Henkin semantics used for the completeness of SOL (see, e.g. [9,15]). Thus, in this section we introduce a similar Henkin-style semantic characterizations for $\mathcal{L}_{RTC}$, and prove the completeness of AL_G with respect to it. This will establish that AL_G indeed meets also the third criterion of a useful approximation for AL given at the beginning of Section 3.
First we recall the concepts of Henkin structures. A σ-Henkin structure is a standard structure together with a subset of the power-set of its domain (called its set of admissible subsets) which is closed under parametric definability.

Definition 6. *Let σ be a first-order signature. A σ-Henkin structure M is a triple $\langle D, I, D' \rangle$, such that:*

- *$\langle D, I \rangle$ is a standard structure for σ (i.e., D is a non-empty domain and I is an interpretation function on σ in D)*
- *$D' \subseteq P(D)$ such that for each formula φ in σ, and assignment v in M[7]:*

$$\{a \in D \mid M, v[x := a] \models \varphi\} \in D'$$

In case $D' = P(D)$, the σ-Henkin structure is called a standard structure.

Notice that in finite structures every subset of the domain is parametrically definable, hence non-standard σ-Henkin structures are necessarily infinite.

It should be noted that the notion of "non-standard" structures is commonly used in mathematical logic, but in a different sense. There are two ways in which a σ-Henkin structure can be non-standard. The "standard way" for it to be non-standard is by having a non-standard first-order part $\langle D, I \rangle$ (in which case D' must necessarily be non-standard). However, a σ-Henkin structure can be non-standard even in case its first-order part is standard, simply by having $D' \subsetneq P(D)$. The latter is what we here mean by a non-standard σ-Henkin structure.

Definition 7. *Let $\mathcal{L}_{RTC}$ be the language based on the signature σ. $\mathcal{L}_{RTC}$ formulas are interpreted in σ-Henkin structures as in standard structures, except for the following clause:*

- *$M, v \models (RTC_{x,y}\varphi)(s, t)$ if for every $A \in D'$, if $v(s) \in A$ and for every $a, b \in D$: $(a \in A \wedge M, v[x := a, y := b] \models \varphi) \rightarrow b \in A$, then $v(t) \in A$.*

Example 8. To give an example of a non-standard σ-Henkin structure, consider the relational language of arithmetic $\sigma = \{0, S, =\}$, where S stands for the successor relation (note that we here use equality in the signature). Let M be the structure whose first-order part is the standard structure of the natural numbers, and let D' be the collection of subsets of the natural numbers that are definable

[7] An assignment v in M is defined as in the standard semantics.

without parameters in the language of AL (i.e. definable by a formula with only one free variable). A set that is definable relative to definable parameters is definable without parameters, so D' is closed under definability. Thus, M is a σ-Henkin structure which is clearly non-standard as $D' \subsetneqq P(D)$. Now, AL has a categorical characterization of the natural numbers (see, e.g., [1,2,15]), and it is straightforward to verify that M indeed satisfies all the characterizing axioms.

The next proposition shows that the generalized Henkin-style semantics coincides with the standard semantics on standard structures.

Proposition 9. *Let M be a standard structure and v an assignment in M. Then, the followings are equivalent:*

1. *$v(s) = v(t)$ or there exist $a_0, ..., a_n \in D$ $(n > 0)$ such that $v(s) = a_0$, $v(t) = a_n$, and $M, v[x := a_i,\ y := a_{i+1}] \models \varphi$ for $0 \leq i \leq n-1$.*
2. *for every $A \subseteq D$, if $v(s) \in A$ and for every $a, b \in D$: $M, v[x := a, y := b] \models \varphi$ and $a \in A$ implies $b \in A$, then $v(t) \in A$.*

Proof. Suppose (1). Let $A \subseteq D$ be a set that is closed under φ, and v an assignment such that $v(s) = a_0 \in A$. If $v(s) = v(t)$ we are done. Otherwise, by induction on the sequence $a_0, ..., a_n$ it is straightforward to prove that $v(t) = a_n \in A$. For the converse, assume by contradiction that (1) does not hold. Take A to be that set which includes $v(s)$ as well as all $a_n \in D$ such that there exist $a_0, ..., a_{n-1} \in D$ $(n > 0)$ where $v(s) = a_0$, and $M, v[x := a_i,\ y := a_{i+1}] \models \varphi$ for $0 \leq i \leq n-1$. By assumption, $v(t) \notin A$, which contradicts (2), since A is obviously φ-closed. □

Before proceeding, a discussion of the value of this type of generalized Henkin semantics for AL is in order. This semantics originated in the completeness result for SOL [9]. There, in order to achieve completeness, the semantics of the non first-order part of the language (the second-order variables) had to weakened. Similarly, we here form a relaxation of the intended semantics for AL by taking a more liberal condition for the non first-order part of the language, the RTC operator. On standard structures, this semantics gives to an RTC-formula its intended top-down meaning (as in Prop. 9(2)). That is, $(RTC_{x,y}\varphi)(s,t)$ holds when *any* property (represented by A) which is closed under φ and contains the interpretation of s also contains the interpretation of t. This corresponds to the standard mathematical definition of the operator as the union of the identity relation with the intersection over *all* transitive binary relations that contain the interpretation of φ (to see this, notice that $a \in A \to b \in A$ may be considered as a transitive binary relation). Now, this is a strong requirement which also renders this definition non-constructive (apart from in trivial cases). The Henkin-style semantics given above relaxes this definition by referring not to all $A \subseteq D$, but only to certain ones. The closure condition on Henkin structures entails that those A's on which the property should be verified are those which are definable (with parameters) in the language. This is a definitional approach to the transitive closure operator which is very much computationally-oriented.

The meaning of the transitive closure is what gives AL its inductive power, which is required for many applications in computer science (as surveyed in the Introduction). But the inductive power actually needed and used in such applications is not over arbitrary elements, but over elements which can be defined.

This generalization of the semantics is what entails the completeness result for AL_G in the sequel. This is because in the induction rule of the formal system AL_G there is an implicit condition that the hereditary property can be defined by a formula, there denoted by ψ. Actually, this condition holds in any formal system, and thus is a critical property in any computational framework. In light of that, the completeness result also suggests that those "standard truths" of AL which are not provable in AL_G hold due to inductive reasoning on some non-definable (non-computable) set.

Any classical structure $M = \langle D, I\rangle$ for σ induces a set of σ-Henkin structures $H(M) = \{M_H = \langle D, I, D'\rangle \mid M_H \,\text{is a}\, \sigma\text{−Henkin structure}\}$. Conversely, each σ-Henkin structure M corresponds to the classical structure obtained by the forgetfulness of D'.

Definition 10. *Let $T \cup \{\varphi\}$ be a set of formulas in a language based on the signature σ. We say that $T \models_H \varphi$ if every σ-Henkin model of T is a model of φ. We say that $T \models_S \varphi$ if every standard model of T is a model of φ.*

Proposition 11. *Let $T \cup \{\varphi\}$ be a set of formulas. If $T \models_H \varphi$ then $T \models_S \varphi$.*

Proof. Follows from the fact that every standard model for T may be viewed as a Henkin model. □

We start by showing the completeness of AL_G. Therefore in what follows, unless mentioned otherwise, we assume $\mathcal{L}_{RTC}$ does not contain equality.

Theorem 12 (Soundness). *Let $T \cup \{\varphi\}$ be a set of sentences in $\mathcal{L}_{RTC}$. Then, $T \vdash_{AL_G} \varphi$ implies $T \models_H \varphi$.*

Proof. It is straightforward to verify that Axiom (1) and Rule (2) of AL_G are sound with respect to the Henkin-style semantics. For Rule (3) simply take $A := \{a \in D \mid M, v[x := a] \models \psi\}$. Now, $A \in D'$ since σ-Henkin structures are closed under parametric definability. By the assumptions we have that A is φ-closed and $v(s) \in A$, which by the semantics of the RTC formula entails $v(t) \in A$. □

The main result of this section is Theorem 13 below, which we shall prove using several lemmas and definitions.

Theorem 13 (Completeness). *Let $T \cup \{\varphi\}$ be a set of sentences in $\mathcal{L}_{RTC}$. Then, $T \models_H \varphi$ implies $T \vdash_{AL_G} \varphi$.*

We prove the completeness theorem using the standard method, showing that if $T \nvdash_{AL_G} \varphi$, then $T \not\models_H \varphi$. First, we extend the language $\mathcal{L}_{RTC}$ to a language $\mathcal{L}'_{RTC}$ by adding to it countably many new constant symbols, $c_1, c_2, ...$, and countably many new monadic predicates, $P_1, P_2, ...$ It is easy to see that $T \nvdash_{AL_G} \varphi$ in the extended language as well.

Definition 14. *We say that a set of $\mathcal{L}'_{RTC}$ sentences Γ contains Henkin witnesses if the followings hold:*

1. *if $\exists x\varphi \in \Gamma$, then $\varphi\left\{\frac{c}{x}\right\} \in \Gamma$ for some constant c.*
2. *if $\neg\left(RTC_{x,y}\varphi\right)(s,t) \in \Gamma$, then $P(s)$, $\forall x, y\,(P(x) \wedge \varphi(x,y) \to P(y))$, $\neg P(t) \in \Gamma$ for some monadic predicate P.*
3. *if φ is a formula of $\mathcal{L}'_{RTC}$ with $Fv(\varphi) = \{x\}$, then $\forall x\,(P(x) \leftrightarrow \varphi) \in \Gamma$ for some monadic predicate P.*

The next Lemma established that the standard method of relational extension by definitions is conservative.

Lemma 15. *Let T be a set of sentences in $\mathcal{L}'_{RTC}$ such that $T \nvdash_{AL_G} \varphi$, and let θ be a sentence of the form $\forall x\,(P(x) \leftrightarrow \psi)$, where P is a fresh monadic predicate (i.e. does not occur in $T \cup \{\varphi, \psi\}$). Then, $T, \theta \nvdash_{AL_G} \varphi$.*

Proof. Suppose by contradiction that there is a proof from $T \cup \{\forall x\,(P(x) \leftrightarrow \psi)\}$ of φ in AL_G, where P is a fresh monadic predicate. First rename all bound variables in the proof (apart from x in the formula $\forall x\,(P(x) \leftrightarrow \psi)$) with new variables not occurring in the proof or in $\forall x\,(P(x) \leftrightarrow \psi)$. Now, replace all the occurrences of formulas of the form $P(t)$ in the proof by $\psi\left\{\frac{t}{x}\right\}$. Then, every occurrence of $\forall x\,(P(x) \leftrightarrow \psi)$ in the proof becomes an occurrence of $\forall x\,(\psi \leftrightarrow \psi)$, which of course is provable in AL_G. It is straightforward to show that if the replacement is done on an axiom, then the result is still an axiom of AL_G. It is also easy to verify that all the inference rules apply equally to the formulas after the replacement. Also notice that since P does not occur in $T \cup \{\varphi\}$, the replacement procedure applied to a formula in $T \cup \{\varphi\}$ results in the same formula. Hence, the replacement procedure indeed produces a proof of φ from T in AL_G. This shows that $T \vdash_{AL_G} \varphi$, which is a contradiction. $\square$

Lemma 16. Let P be a monadic predicate and θ a formula of $\mathcal{L}'_{RTC}$. Then:

$$P(s), \forall x, y\,(P(x) \wedge \theta(x,y) \to P(y)), \neg P(t) \vdash_{AL_G} \neg\left(RTC_{x,y}\theta\right)(s,t)$$

Proof. The claim immediately follows from Rule (3), taking $\varphi(x,y) := \theta(x,y)$ and $\psi(x) := P(x)$. $\square$

Lemma 17. *There exists an extension of T to a set of sentences T' in the language $\mathcal{L}'_{RTC}$ such that:*

1. *T' is a maximal theory in $\mathcal{L}'_{RTC}$ such that $T' \nvdash_{AL_G} \varphi$.*
2. *T' contains Henkin witnesses.*

Proof. Fix two enumerations: one of all sentences of $\mathcal{L}'_{RTC}$: $\psi_1, \psi_2, ...$; and one of all the formulas of $\mathcal{L}'_{RTC}$ with one free variable x: $\theta_1, \theta_2, ...$ Define a sequence of theories $T_0, T_1, ...$ inductively in the following way: $T_0 = T$, and for $i > 0$ T_i is constructed from T_{i-1} as follows:

1. If $i = 2n - 1$ for some $n \in \mathbb{N}$, then:

(a) If $T_{i-1} \cup \{\psi_n\} \vdash_{AL_G} \varphi$, then $T_i = T_{i-1}$.
(b) If $T_{i-1} \cup \{\psi_n\} \nvdash_{AL_G} \varphi$, then:
    i. If ψ_n is not of the form $\exists x\psi$ or $\neg(RTC_{x,y}\psi)(s,t)$, $T_i = T_{i-1} \cup \{\psi_n\}$.
    ii. If $\psi_n = \exists x\psi$, then $T_i = T_{i-1} \cup \left\{\psi_n, \psi\left\{\frac{c_j}{x}\right\}\right\}$, for c_j a fresh constant symbol not in T_{i-1}.
    iii. If $\psi_n = \neg(RTC_{x,y}\psi)(s,t)$, then $T_i = T_{i-1} \cup \{\psi_n, P_j(s), \neg P_j(t), \forall x, y(P_j(x) \wedge \psi(x,y) \to P_j(y))\}$, for P_j a fresh monadic predicate not in T_{i-1}.

2. If $i = 2n$ for some $n \in \mathbb{N}$, then $T_i = T_{i-1} \cup \{\forall x(P_j(x) \leftrightarrow \theta_n)\}$, for P_j a fresh monadic predicate not in T_{i-1}.

We show by induction that for every $i \in \mathbb{N}$, $T_i \nvdash_{AL_G} \varphi$. Lemma 15 entails that if $T_{2n-1} \nvdash_{AL_G} \varphi$, then $T_{2n} \nvdash_{AL_G} \varphi$. For $i = 2n-1$: Cases (a) and (b i) are trivial, and Case (b ii) is provable just as in the standard completeness proof for FOL. Thus, we here prove Case (b iii). Assume by contradiction that $T_{i-1}, \neg(RTC_{x,y}\psi)(s,t), P_j(s), \forall x, y.P_j(x) \wedge \psi(x,y) \to P_j(y), \neg P_j(t) \vdash_{AL_G} \varphi$. Since $P_j(s), \forall x, y.P_j(x) \wedge \psi(x,y) \to P_j(y), \neg P_j(t) \vdash_{AL_G} \neg(RTC_{x,y}\psi)(s,t)$, by Lemma 16 we have $T_{i-1}, P_j(s), \forall x, y.P_j(x) \wedge \psi(x,y) \to P_j(y), \neg P_j(t) \vdash_{AL_G} \varphi$. Now, P_j is a fresh monadic predicate which does not appear in $T_{i-1} \cup \{\varphi\}$. Therefore, it is straightforward to verify that replacing all occurrences of formulas of the form $P_j(r)$ in the above proof with $(RTC_{x,y}\psi)(s,r)$ results in a proof in AL_G of φ from the set $T_{i-1} \cup \{(RTC_{x,y}\psi)(s,s), \neg(RTC_{x,y}\psi)(s,t), \forall x, y.(RTC_{x,y}\psi)(s,x) \wedge \psi(x,y) \to (RTC_{x,y}\psi)(s,y)\}$. Now, $(RTC_{x,y}\psi)(s,s)$ is an axiom of AL_G, and $\forall x, y((RTC_{x,y}\psi)(s,x) \wedge \psi(x,y) \to (RTC_{x,y}\psi)(s,y))$ is provable in AL_G using Rule (2). Hence, we get that $T_{i-1}, \neg(RTC_{x,y}\psi)(s,t) \vdash_{AL_G} \varphi$, which contradicts the original assumption that $T_{i-1} \cup \{\psi_i\} \nvdash_{AL_G} \varphi$. Therefore, $T_i \nvdash_{AL_G} \varphi$.

Now, take $T' = \bigcup_{i=0}^{\infty} T_i$. The construction of T' entails that it satisfies the two requirements of the claim. □

Next we construct a Henkin model for T', which does not satisfy φ.

Definition 18. *Define M by:*

- $D = \{t \mid t \, \text{is a closed term}\}$
- $D' = \{\{t \mid P(t) \in T'\} \mid P \, \text{is a monadic predicate}\}$
- $\langle t_1, ..., t_n \rangle \in I(P)$ *iff* $P(t_1, ..., t_n) \in T'$
- $I(c) = c$ *for a constant symbol* c
- $I(f)(t_1, ..., t_n) = f(t_1, ..., t_n)$ *for a n-ary function symbol* f

Notice that $D' = \{I(P) \mid P \, \text{is a monadic predicate}\}$.

Lemma 19. *Let ψ be a formula in $\mathcal{L}'_{RTC}$. The following holds:*

- $M, v \models \psi$ *iff* $M \models \psi\left\{\frac{v(x_1)}{x_1}, ..., \frac{v(x_n)}{x_n}\right\}$, *where* $Fv(\psi) = \{x_1, ..., x_n\}$.
- $T' \models_H \forall x\psi$ *iff* $T' \models_H \psi\left\{\frac{t}{x}\right\}$ *for every closed term* t.

Lemma 20. *M is a σ-Henkin structure.*

Proof. The claim follows from the fact that T' contains Henkin witnesses of the third type in Definition 14, i.e., a monadic predicate was introduced for each parametrically definable subset (using the new constant symbols instead of the parameters). To see this, let v be an assignment in M, and let ψ be a formula with $Fv(\psi) = \{x_1, ..., x_n\}$. Then, $\{a \in D \mid M, v[x_1 := a] \models \psi\} = \left\{a \in D \mid M, v[x_1 := a] \models \psi\left\{\frac{v(x_2)}{x_2}, ..., \frac{v(x_n)}{x_n}\right\}\right\}$. In T' there exists a monadic predicate which forms a Henkin witness for $\psi\left\{\frac{v(x_2)}{x_2}, ..., \frac{v(x_n)}{x_n}\right\}$, denote it by $P_k(x_1)$. This entails that $\left\{a \in D \mid M, v[x_1 := a] \models \psi\left\{\frac{v(x_2)}{x_2}, ..., \frac{v(x_n)}{x_n}\right\}\right\} = I(P_k) \in D'$. $\square$

Lemma 21. *For every sentence θ in $\mathcal{L}'_{RTC}$: $M \models \theta$ iff $\theta \in T'$.*

Proof. By induction on θ. The base case follows immediately from the definition of M. For the connectives and quantifiers the proof is similar to the standard proof for FOL (using Henkin witnesses for existential formulas). We next prove the case for $\theta = (RTC_{x,y}\psi)(s,t)$.
$(\Rightarrow)$: Assume $M \models (RTC_{x,y}\psi)(s,t)$. Hence, for every monadic predicate P, if for every $a, b \in D$:$(a \in I(P) \wedge M, v[x := a, y := b] \models \psi) \rightarrow b \in I(P)$ and $I(s) \in I(P)$, then $I(t) \in I(P)$. Using the induction hypothesis and the base case we get that for any monadic predicate P, if $P(s) \in T'$ and for any two closed terms a, b, if $P(a) \in T'$ and $\psi(a,b) \in T'$ then $P(b) \in T'$, then $P(t) \in T'$. From this we deduce (using Lemma 19) that for any monadic predicate P, if $P(s) \in T'$ and $\forall x, y\,(P(x) \wedge \psi(x,y) \rightarrow P(y)) \in T'$, then $P(t) \in T'$. Assume by contradiction that $(RTC_{x,y}\psi)(s,t) \notin T'$. By the maximality of T', we get that $\neg(RTC_{x,y}\psi)(s,t) \in T'$. Therefore, T' contains Henkin witnesses of the type $P(s)$, $\forall x, y\,(P(x) \wedge \psi(x,y) \rightarrow P(y))$ and $\neg P(t)$ for some monadic predicate P. But this contradicts the consistency of T', since we showed that for any monadic predicate P, if $P(s) \in T'$ and $\forall x, y\,(P(x) \wedge \psi(x,y) \rightarrow P(y)) \in T'$, then $P(t) \in T'$. Hence we conclude that $(RTC_{x,y}\psi)(s,t) \in T'$.
$(\Leftarrow)$: Assume $M \not\models (RTC_{x,y}\psi)(s,t)$. So, $M \models \neg(RTC_{x,y}\psi)(s,t)$ and there exists a monadic predicate P such that $I(s) \in I(P)$, $I(t) \notin I(P)$, and for every $a, b \in D$: $(a \in I(P) \wedge M, v[x := a, y := b] \models \psi) \rightarrow b \in I(P)$. By the induction hypothesis and the base case we get that there exists a monadic predicate P such that $P(s) \in T'$, $P(t) \notin T'$, and for any two closed terms a, b, if $P(a) \in T'$ and $\psi(a,b) \in T'$ then $P(b) \in T'$. Therefore, by the maximality of T', $P(s) \in T'$, $\neg P(t) \in T'$ and $\forall x, y\,(P(x) \wedge \psi(x,y) \rightarrow P(y)) \in T'$ (the latter holds since assuming otherwise leads to a contradiction using a Henkin witness for an existential formulas). This entails, by Lemma 16, $T' \vdash_{AL_G} \neg(RTC_{x,y}\psi)(s,t)$. Assuming $(RTC_{x,y}\psi)(s,t) \in T'$ contradicts the consistency of T', therefore $(RTC_{x,y}\psi)(s,t) \notin T'$. $\square$

From the above series of definitions and lemmas we can finally prove Theorem 13. Since the original theory T is contained in T' and $\varphi \notin T'$, Lemma 21 entails that the model M constructed in Def. 18 satisfies T, but not φ. Hence, we get that $T \not\models_H \varphi$, which concludes the proof of the Completeness Theorem for AL_G.

The completeness of $AL_G^=$ is obtained similarly. Soundness of the additional equality rules is straightforward. The main modification needed in the completeness proof for languages with equality is in the construction of the structure M (Definition 18). In this case M is obtained by taking the domain D to be the quotient set on terms under the equivalence relation: $t_1 \equiv t_2$ iff $t_1 = t_2 \in T'$. The other components of the definition are then altered straightforwardly, taking the equivalence class of closed terms instead of the terms themselves (just as in the standard completeness proof for first-order languages with equality).

It should be noted that in [2] a Gentzen-style proof system for the non-reflexive transitive closure operator was presented, and it was shown that there exist provability preserving interpretations between the two logics. Using similar methods to the ones used here, it is straightforward to provide a generalized Henkin-style semantics for the non-reflexive transitive closure operator and to prove that its corresponding proof system is complete with respect to it.

5 Conclusions and Further Research

In this paper we took another step in the development of the theory of AL as a foundational logical framework for computer science applications. A Henkin-style semantics for AL was introduced and a natural formal system for AL was proven to be sound and complete with respect to it. This leads to various open questions and possible research directions in the exploration of the theory of AL.

One important research task is establishing some form of cut-elimination theorem for AL_G. A non-constructive result might be obtainable using methods similar to the ones used for SOL in [7,16]. To achieve constructive cut-elimination result a plausible option is to search for a suitable definition of the notion "subformula" under which some form of analytical cut-elimination can be obtained. It is clear that the usual definition of a subformula should be revised, exactly as the straightforward notion of subformula used in propositional languages is changed on the first-order level, where for example a formula of the form $\psi\{\frac{t}{x}\}$ is considered to be a subformula of $\forall x \psi$, even though it might be much longer than the latter. Thus the induction rule of AL_G satisfies the subformula property only if we take a formula to be a subformula of every substitution instance of it.

The system AL_G is not complete with respect to the intended semantics. It is not difficult to express its consistency in the language $\{=, 0, S, +\}$ as a logically valid (under the standard semantics) sentence $Con_{AL_G^=}$ of AL. By Gödel's theorem on consistency proofs, $Con_{AL_G^=}$ is not a theorem of $AL_G^=$. It would be interesting to find what valid principles of AL (not available in AL_G) can be used to derive it. The completeness result of this paper suggests that those principles are connected with inductive reasoning over arbitrary (undefinable) sets.

Another interesting task is to determine and explore fragments of AL that are more convenient to work with, but are still sufficient for at least some concrete applications. An example of such a fragment may be the one which corresponds to the use of the *deterministic* transitive closure operator (see, e.g., [10]). Another option worth investigating is to restrict the induction rule by allowing only φ's of the form $y = t$, where $Fv(t) = \{x\}$. Implicitly, this is the fragment of AL used in Peano's Arithmetics.

In [15] it is noted that Craig interpolation theorem and Beth definability theorem fail for logics in which the notion of finiteness can be expressed. Thus, a future research task is to find appropriate AL counterparts (whenever such exist) to central model-theoretic properties of FOL such as these.

Acknowledgments

This research was supported by: Ministry of Science, Technology and Space, Israel; Fulbright Post-doctoral Scholar program; Weizmann Institute of Science – National Postdoctoral Award Program for Advancing Women in Science; Eric and Wendy Schmidt Postdoctoral Award program for Women in Mathematical and Computing Sciences; and Cornell University PRL Group. The author is indebt to A. Avron for his invaluable comments and expertise that greatly assisted this research.

References

1. A. Avron. Transitive closure and the mechanization of mathematics. In F. D. Kamareddine, editor, *Thirty Five Years of Automating Mathematics*, volume 28 of *Applied Logic Series*, pages 149–171. Springer, Netherlands, 2003.
2. L. Cohen and A. Avron. The middle ground–ancestral logic. *Synthese*, pages 1–23, 2015.
3. L. Cohen and R. L. Constable. Intuitionistic ancestral logic. *Journal of Logic and Computation*, 2015.
4. HD. Ebbinghaus and J. Flum. *Finite Model Theory.* Springer Science & Business Media, 2005.
5. S. Feferman. Finitary inductively presented logics. *Studies in Logic and the Foundations of Mathematics*, 127:191–220, 1989.
6. G. Gentzen. Investigations into logical deduction, 1934. In German. An English translation appears in 'The Collected Works of Gerhard Gentzen', edited by M. E. Szabo, North-Holland, 1969.
7. JY. Girard. *Proof theory and logical complexity*, volume 1. Humanities Press, 1987.
8. J. Y. Halpern, R. Harper, N. Immerman, P. G. Kolaitis, M. Y. Vardi, and V. Vianu. On the unusual effectiveness of logic in computer science. *Bulletin of Symbolic Logic*, 7(02):213–236, 2001.
9. L. Henkin. Completeness in the theory of types. *Journal of Symbolic Logic*, 15(2):81–91, 1950.
10. N. Immerman. Languages that capture complexity classes. *SIAM Journal on Computing*, 16(4):760–778, 1987.
11. D. Kozen. Kleene algebra with tests. *ACM Transactions on Programming Languages and Systems (TOPLAS)*, 19(3):427–443, 1997.

12. R. M. Martin. A homogeneous system for formal logic. *Journal of Symbolic Logic*, 8(1):1–23, 1943.
13. R. M. Martin. A note on nominalism and recursive functions. *Journal of Symbolic Logic*, 14(1):27–31, 1949.
14. J. Myhill. A derivation of number theory from ancestral theory. *Journal of Symbolic Logic*, 17(3):192–197, 1952.
15. S. Shapiro. *Foundations without foundationalism: A case for second-order logic*. Oxford University Press, 1991.
16. W. W. Tait. A nonconstructive proof of gentzen's hauptsatz for second order predicate logic. *Bulletin of the American Mathematical Society*, 72(6):980–983, 1966.
17. G. Takeuti. *Proof theory*. Courier Dover Publications, 1987.
18. S. Wohrle and W. Thomas. Model checking synchronized products of infinite transition systems. In *Logic in Computer Science*, pages 2–11, 2004.