Information and Computation **147**, 57–88 (1998)
Article No. IC982731

Verifiable Properties of Database Transactions

Bell Laboratories, 1000 E. Warrenville Road, Naperville, Illinois 60566
E-mail: benedikt@research.bell-labs.com

and

Timothy Griffin and Leonid Libkin*

Bell Laboratories, 600 Mountain Avenue, Murray Hill, New Jersey 07974
E-mail: griffin@research.bell-labs.com, libkin@research.bell-labs.com

It is often necessary to ensure that database transactions preserve integrity constraints that specify valid database states. While it is possible to monitor for violations of constraints at run-time, rolling back transactions when violations are detected, it is preferable to verify correctness statically, *before* transactions are executed. This can be accomplished if we can verify transaction safety with respect to a set of constraints by means of calculating *weakest preconditions*. We study properties of weakest preconditions for a number of transaction and specification languages. We show that some simple transactions do not admit weakest preconditions over first-order logic and some of its extensions such as first-order logic with counting and monadic Σ^1_1. We also show that the class of transactions that admit weakest preconditions over first-order logic cannot be captured by any transaction language. We consider a strong local form of verifiability, and show that it is different from the general form. We define robustly verifiable transactions as those that can be statically analyzed regardless of extensions to the signature of the specification language, and we show that the class of robustly verifiable transactions over first-order logic is exactly the class of transactions that admit the local form of verifiability. We discuss the implications of these results for the design of verifiable transaction languages.

1. INTRODUCTION

Databases are typically required to satisfy a collection of *integrity constraints*—sentences that specify the valid states of the database. Database transactions that update the database must preserve these constraints. One approach to the integrity

* To whom correspondence should be addressed.

0890-5401/98 $25.00

maintenance problem is to defer the detection of potential violations of integrity constraints until run-time, thus preventing the violation of integrity that may be caused by transactions. However, this approach results in potentially expensive roll-back operations. To avoid this, one can attempt to verify *statically*, before a transaction is executed, that it will necessarily maintain all constraints. If the result of this verification shows that the transaction may sometimes fail to maintain some of the constraints, then modifications can be made until correctness is achieved.

If T is a transaction and α is a constraint, we say that T preserves α if for every database D:

$$D \models \alpha \qquad \text{implies} \quad T(D) \models \alpha.$$

We use the notation $D \models \alpha$ to mean that α is true for the database D.

According to the approach described above, we would like to be able to check transactions to see if they will always preserve α. But, as the following result tells us, one cannot hope to have an algorithm for checking this, even for simple transactions and simple constraint specification languages.

Fact A (cf. [2]). For transactions specified as select-project-join expressions of the relational algebra, and integrity constraints specified as first-order sentences, it is undecidable to check if a given transaction preserves a given integrity constraint.

Similar results hold for transactions that consist of simple updates and deletions. Therefore, any approach to automatic verification of transaction safety (such as the approach of [35, 37] that uses theorem-provers) is inherently limited. Instead, we adopt an approach based on modifying transactions to ensure safety.

Given a transaction T and a constraint α, we would like to find another constraint β that is satisfied by a database D *before* T is committed if and only if α is satisfied *afterward*. In other words, for every database D,

$$D \models \beta \qquad \text{if and only if} \quad T(D) \models \alpha.$$

We shall call such a β a *weakest precondition* for α, with respect to T, and denote it by $\mathsf{wpc}(T, \alpha)$. We will require that $\mathsf{wpc}(T, \alpha)$ be computable for given T and α. Since we consider only transactions that do not diverge, this notion of weakest precondition is equivalent to the classical one of Dijkstra [11, 12].

The existence of weakest preconditions would not contradict Fact A, as we would not be able to determine the validity of the implication $\mathsf{wpc}(T, \alpha) \to \alpha$. At the same time it would solve the problem of statically verifying database integrity since any transaction T could be transformed to

$$\underline{\text{if}}\ \mathsf{wpc}(T, \alpha)\ \underline{\text{then}}\ T\ \underline{\text{else}}\ \textit{abort}$$

which will maintain consistency while avoiding any roll-back operations. Furthermore, we could then modify the resulting transaction by applying simplification algorithms to $\mathsf{wpc}(T, \alpha)$, thus recapturing many of the benefits of approaches based

on validity-checking. This is the fundamental idea underlying many algorithms for the automatic maintenance of integrity constraints [29, 21, 22, 31, 28].

Given the attractiveness of this approach to integrity maintenance, it is important to understand the tradeoffs involved in designing transaction and specification languages with respect to our ability to express weakest preconditions. In this paper, we will concentrate on the basic principles behind developing safe transactions via preconditions, rather than specific algorithms that attempt to detect violations of integrity.

The main goal of this paper is to study the properties of weakest preconditions for a variety of transaction and specification languages. In particular, we concentrate on specification languages that are relevant to integrity constraints, such as first-order logic over the database schema. Below we give a brief overview of the main results of the paper.

Assume that we have two languages $\mathscr{L}_1$ and $\mathscr{L}_2$ in which constraints may be specified. Also assume that we have a transaction language $\mathscr{TL}$. *We say that* $\mathscr{TL}$ *is* $\mathscr{L}_1$*-verifiable over* $\mathscr{L}_2$ iff for every $T \in \mathscr{TL}$ and every $\alpha \in \mathscr{L}_2$, it is the case that a weakest precondition for α with respect to T is in $\mathscr{L}_1$. If $\mathscr{L} = \mathscr{L}_1 = \mathscr{L}_2$, we speak of $\mathscr{L}$*-verifiable* transaction languages. Similar concepts have been studied in the more general programming language context, see [6, 9].

Verifiable transaction languages are exactly those for which the weakest-precondition approach to integrity maintenance is possible. An example of a language verifiable over first-order logic is the first-order transaction language of [32]. However, no results exist for transaction languages that possess more power than first-order logic (for example, for languages that allow some forms of recursion). The first result of this paper is negative: it shows that adding recursion destroys first-order verifiability.

THEOREM B. *Let* $\mathscr{TL}$ *be a transaction language that expresses at least one of the following*:

- *transitive closure*;
- *deterministic transitive closure*;
- *same-generation* (*even restricted to trees*).

Then $\mathscr{TL}$ *is not verifiable over any of the following languages*: *first-order logic*, *first-order logic with counting*, *first-order logic supplemented with a recursive collection of recursive functions and predicates*, *monadic* Σ_1^1.

We next fix our specification language to be first-order logic, and look at the set $\mathscr{WPC}(\mathrm{FO})$ of transactions (that is, total maps from databases to databases) that have weakest preconditions for first-order sentences. We show that this collection cannot be captured by any transaction language.

THEOREM C. *There is no transaction language that expresses exactly all the transactions in* $\mathscr{WPC}(\mathrm{FO})$. *This remains true if* FO *is replaced by first-order logic over a signature containing a recursive collection of interpreted recursive functions and predicates.*

The ability to statically check global conditions—that is, conditions that must be satisfied by the database as a whole—is, as noted above, extremely helpful. One might also demand a stronger "local" version of this, in which one can statically check properties of individual tuples in the resulting database. That is, one often would like to find a condition that must be satisfied by a database D and an object t (typically, a record) in order for t to belong to $T(D)$. It is natural to compare the class $\mathscr{WPC}(\mathscr{L})$ with the class of transactions for which this *local* checking can be carried out. For a specification language $\mathscr{L}$, the natural definition of such a class is the set of transactions admitting *prerelations*. Informally, a transaction T admits prerelations over $\mathscr{L}$ if it is definable by $\mathscr{L}$-formulae on a set that extends the active domain of the database. That is, for every relation R of arity n there exists a $\mathscr{L}$-formula $\mathsf{pre}_R^T(x_1, ..., x_n)$ such that, for every n-tuple $t = (d_1, ..., d_n)$ and every database D:

$$D \models \mathsf{pre}_R^T(d_1, ..., d_n) \qquad \text{and} \qquad d_1, ..., d_n \in \Gamma(D) \Leftrightarrow T(D) \models R(d_1, ..., d_n).$$

Here $\Gamma(D)$ is a finite set that extends the active domain of the database. The precise definition will be given in the next section.

The condition $T(D) \models R(d_1, ..., d_n)$ means that t belongs to the relation R after the transaction T is applied to the database D. Thus, prerelations allow us to test membership before transactions are committed.

In papers like [32] prerelations are used in place of weakest preconditions. The question of the relationship of these two notions, however, has not been addressed. We shall see that for most specification languages of interest, it is the case that any transaction that admits prerelations also admits weakest preconditions. The converse, however, is not true even in very simple settings.

THEOREM D. *There exists a transaction T in $\mathscr{WPC}(\mathrm{FO})$ that does not admit prerelations. Moreover, T can be chosen to be* $\mathrm{Datalog}^{\neg}$*-definable.*

However, this result is not robust: if FO is extended by adding a constant for each element of the domain, then the transaction from Theorem D no longer provides a separation example. Motivated by this, we say that a transaction language $\mathscr{TL}$ is *robustly verifiable* over a first-order signature Ω if, for any extension of Ω to a signature Ω' via a set of recursive predicates and functions, all transactions in $\mathscr{TL}$ remain verifiable over the corresponding extension of first-order logic. We then prove a positive result that characterizes maximal robustly verifiable languages when integrity constraints can refer to the individual elements of the domain. Let FO_c be first-order logic over the relational schema supplemented with symbols for all constants.

THEOREM E. *There exists a transaction language that captures the class of transactions admitting prerelations over* FO_c*. Moreover, this language is the maximal robustly verifiable language over* FO_c*. In particular, every transaction robustly verifiable over* FO_c *is equivalent to a transaction that admits prerelations.*

This has implications for transaction language design. If we are interested in a "nice" transaction language that is verifiable in an extensible way over all first-order

languages, then we cannot hope that the language will be more expressive than the first-order transaction language defined in [32, 33].

Organization. We rigorously define our formal setting in the next section. In Section 3 we study the limitations of weakest preconditions. Section 4 studies the relationship between weakest preconditions and prerelations. Robustly verifiable transaction languages are the subject of Section 5. Concluding remarks are given in Section 6.

The extended abstract of this paper appeared in [8].

2. FORMAL SETTING

We fix the domain to be a countably infinite set $\mathscr{U}$. A *relational schema* is a non-empty set $SC = \langle R_1, ..., R_k \rangle$ of predicates, with a finite arity $n_i > 0$ associated with each R_i. A *database* over SC is an interpretation of each R_i as a finite subset of $\mathscr{U}^{n_i}$. Most often we will use the schema that consists of a single binary predicate E. Then databases are interpreted as finite graphs whose nodes are elements of $\mathscr{U}$.

We shall use two notions—*specification language*, in which constraints are specified, and *transaction language*. A specification language $\mathscr{L}$ is a recursive subset of a set of strings with an associate subset $sent(\mathscr{L})$ of *sentences*, which is also recursive. As stated in the introduction, we wish to focus on the specification languages that are relevant to the integrity constraints. In particular, we will study:

- FO—(pure) first-order logic (that is, first-order logic over the relational schema SC).
- FO_c—first-order logic over the signature that consists of SC supplemented with constant symbols for all elements of $\mathscr{U}$.
- $FO_c(\Omega)$—first-order logic over the signature that consists of SC supplemented with constant symbols as above and a recursive collection Ω of recursive functions and predicates over $\mathscr{U}$.

We shall also consider the following two logics, which are more powerful than the first-order logic and play an important role in finite model theory and descriptive complexity.

- FO_{count}—first-order logic with counting. For precise definition, see [13, 19, 24]. This is a two-sorted logic, with the second sort of natural numbers whose universe is $\{1, ..., n\}$, where n is the size of the universe of the first sort. First-order logic is extended with counting quantifiers, $\exists i x.\psi(x)$ meaning that here are at least i elements of the first-sort universe satisfying ψ. This quantifier binds x but not i. The order, the constants for 1 and the maximal element, and the **bit** predicate are available on natural numbers, where $\mathbf{bit}(i, j)$ is true iff the jth bit in the binary representation of i is one. For example, $\exists i \exists i x.\psi(x) \wedge \mathbf{bit}(i, 1) \wedge (\forall j(\exists j x.\psi(x)) \rightarrow j \leqslant i)$ says that there are odd number of elements satisfying ψ. Another example of non-first-order property definable in FO_{count} is equal cardinality.
- Monadic Σ_1^1, whose formulae are of the form $\exists A_1 \cdots \exists A_k . \Psi$ where A_i's are monadic (unary) predicates and Ψ is a first-order formula over $SC \cup \langle A_1, ..., A_k \rangle$.

For all specification languages it will be the case that we shall have a binary relation $\models$ for validity between databases and sentences of the languages. For all the languages above, validity is defined in the usual way.

A transaction language consists of (1) Syntax, a recursive subset of the set S of strings over a finite alphabet, and (2) Semantics, a total recursive function M that takes a string s, and an encoding of a database, and returns either another database encoding or "error," provided $s \in S$. Examples include relational algebra and calculus, SQL, first-order transaction language of [32], a variety of languages from [3, 4], and so on. We shall always use the same symbol for both syntactic and semantic objects. For example, for the language given by $\langle S, M \rangle$, $T \in S$ and a database D, we will write $T(D)$ to denote the result of M on T and the encoding of D.

We are interested in maintaining database integrity. To formalize this problem, let $\mathscr{L}$ be a specification language and $\mathscr{TL}$ a transaction language. We define the following problem:

PRESERVE($\mathscr{TL}$, $\mathscr{L}$). Given $T \in \mathscr{TL}$ and α in $\mathscr{L}$, is it the case that $D \models \alpha$ implies $T(D) \models \alpha$ for every database D?

According to Fact A, this problem is undecidable even for transactions specified by the simplest form of SQL statements and for first-order constraints. This can be derived from a result of [2]. Since the transaction model of [2] differs from the one that we consider, we present a simple proof below for the sake of completeness.

PROPOSITION 1. *Let $\mathscr{L}$ be* FO, *and let $\mathscr{TL}$ contain the transactions given by the select-project-join expressions of the relational algebra. Then* PRESERVE($\mathscr{TL}$, $\mathscr{L}$) *is undecidable.*

Proof. Assume that PRESERVE is decidable, and consider two transactions on graphs: T_1 takes a graph on a finite set of nodes V and produces its diagonal ($\{(x, x) \mid x \in V\}$) and T_2 produces the complete graph without loops ($\{(x, y) \mid x, y \in V,\ x \neq y\}$). $T_1(E)$ can be implemented as $\Pi_{1,3}(\sigma_{1=3}(E \times E))$ and T_2 as $\Pi_{1,3}(\sigma_{1\neq 3}(E \times E))$ (we assume that V is the union of the first and the second projections of E).

It suffices to restrict our attention only to nonempty graphs. Observe that if a first-order sentence ϕ is not satisfied in any $T(D)$, then PRESERVE(T, ϕ) iff $\neg\phi$ is valid. Now let $\psi \equiv \exists x.E(x, x)$, and let β be an arbitrary first-order sentence in the language of $E(\cdot, \cdot)$. Then β is valid in every finite graph if and only if both $\neg\psi \vee \beta$ and $\psi \vee \beta$ are valid in every finite graph. Using the observation above, we derive that $\beta \vee \psi$ is valid iff PRESERVE(T_1, $\neg\beta \wedge \neg\psi$), because $\neg\beta \wedge \neg\psi$ fails in all nonempty outputs of T_1. Similarly, $\beta \vee \neg\psi$ is valid iff PRESERVE(T_2, $\neg\beta \wedge \psi$). Thus, if PRESERVE is decidable, then it is decidable whether an arbitrary sentence β is valid in all finite graphs. Since undecidability of the latter is known (cf. [14]), we have a contradiction. ∎

As we explained in the Introduction, a way around this is to introduce weakest preconditions. Below is the main definition.

DEFINITION. (1) Let $\mathscr{L}_1$ and $\mathscr{L}_2$ be two specification languages. We say that a transaction T in a transaction language $\mathscr{TL}$ has $\mathscr{L}_1$-weakest preconditions over $\mathscr{L}_2$

if for every $\mathscr{L}_2$-sentence α there exists an $\mathscr{L}_1$ sentence $\mathsf{wpc}(T, \alpha)$, computable from T and α, such that for every database D on which T is defined,

$$D \models \mathsf{wpc}(T, \alpha) \qquad \text{if and only if} \quad T(D) \models \alpha.$$

The class of transactions having $\mathscr{L}_1$-weakest preconditions over $\mathscr{L}_2$ is denoted by $\mathscr{WPC}(\mathscr{L}_1, \mathscr{L}_2)$. We also write $\mathscr{WPC}(\mathscr{L})$ for $\mathscr{WPC}(\mathscr{L}, \mathscr{L})$.

(2) We say that a transaction language $\mathscr{TL}$ is $\mathscr{L}_1$-verifiable over $\mathscr{L}_2$ if every $\mathscr{TL}$ transaction T is in $\mathscr{WPC}(\mathscr{L}_1, \mathscr{L}_2)$. If $\mathscr{L} = \mathscr{L}_1 = \mathscr{L}_2$, we speak of a language verifiable over $\mathscr{L}$.

The proposition below shows that while $\mathscr{WPC}(\cdot, \cdot)$ is monotone in the first argument and antimonotone in the second, $\mathscr{WPC}(\cdot)$ need not be (anti)monotone in its one argument. We use the notation $\mathscr{L} \preccurlyeq \mathscr{L}'$ mean that $\mathscr{L}$ is a sublanguage of $\mathscr{L}'$.

PROPOSITION 2. (a) *Let* $\mathscr{L}'_1 \preccurlyeq \mathscr{L}_1$ *and* $\mathscr{L}_2 \preccurlyeq \mathscr{L}'_2$. *Then* $\mathscr{WPC}(\mathscr{L}'_1, \mathscr{L}'_2) \subseteq \mathscr{WPC}(\mathscr{L}_1, \mathscr{L}_2)$.

(b) *It is possible to find languages* $\mathscr{L} \preccurlyeq \mathscr{L}'$ *and a transaction* T *such that* $T \in \mathscr{WPC}(\mathscr{L}) - \mathscr{WPC}(\mathscr{L}')$.

(c) *It is possible to find languages* $\mathscr{L} \preccurlyeq \mathscr{L}'$ *and a transaction* T *such that* $T \in \mathscr{WPC}(\mathscr{L}') - \mathscr{WPC}(\mathscr{L})$.

Proof. (a) follows immediately from the defintions. For (b) and (c), take T to be the transitive closure of a graph. For (b), take $\mathscr{L}'$ to be first-order logic augmented with constants for each element $u \in \mathscr{U}$. Take $\mathscr{L}$ to be the family of Boolean combinations of formulae $\psi_u \equiv \exists x.(E(x, u) \vee E(u, x))$ meaning that u is a node with either incoming or outcoming edge in the graph given by the predicate E. Then $D \models \psi_u$ iff $T(D) \models \psi_u$, and hence $T \in \mathscr{WPC}(\mathscr{L})$. On the other hand, $T \notin \mathscr{WPC}(\mathscr{L}')$ (see Theorem 3). For (c), we take $\mathscr{L}$ to be first-order logic, and $\mathscr{L}'$ to be first-order logic with fixpoint. Since T is definable in the latter, $T \in \mathscr{WPC}(\mathscr{L}')$, but $T \notin \mathscr{WPC}(\mathscr{L})$, see Theorem 2. ∎

Similar notions have been studied in a more general programming language context. For example, weakest preconditions over first-order logic for a simple while loop language can be expressed in infinitary logic [6] or in weak second-order logic [9]. However, Proposition 2 shows that it is not always possible to extend results of this kind to less expressive specification languages. In particular, according to Theorem B, weakest preconditions for transitive closure (which is definable with while loops) are not first-order expressible.

We will also use *prerelations* which allow for testing of local conditions. We are now ready to formalize the definition we presented informally in the introduction. Let Ω be a signature. By $\textsc{Term}(\Omega)$ we denote the set of all Ω-terms which are built up from variables by using the symbols from Ω. (We treat constants as functions of arity zero.) Given a database D, its (active) *domain* $\mathrm{dom}(D)$ is the set of all elements of $\mathscr{U}$ that occur in the database. For a set $\Gamma \subseteq \textsc{Term}(\Omega)$ and a database

D we define $\Gamma(D)$ to be the set of all $x \in \mathscr{U}$ such that $x = \tau(y_1, ..., y_n)$ for some n-ary term $\tau \in \Gamma$ and some $y_1, ..., y_n \in \mathrm{dom}(D)$.

DEFINITION. Let T be a transaction. We say that T admits *prerelations* over $\mathscr{L}$ if there exists a finite collection of terms Γ and a collection of $\mathscr{L}$-formulae $\mathsf{pre}_1^T, ..., \mathsf{pre}_k^T$, pre_i^T having n_i free variables, such that for any R_i in the schema and any $d_1, ..., d_{n_i} \in \mathscr{U}$, it is the case the for every database D:

$$D \models \mathsf{pre}_i^T(d_1, ..., d_{n_i}) \qquad \text{and} \qquad d_1, ..., d_{n_i} \in \Gamma(D) \Leftrightarrow T(D) \models R_i(d_1, ..., d_{n_i})$$

The collection of all transactions that admit prerelations over $\mathscr{L}$ is denoted by $\mathscr{PR}(\mathscr{L})$.

The intuition behind the definition above is this. Since $T(D) \models R_i(d_1, ..., d_{n_i})$ says that the tuple $(d_1, ..., d_{n_i})$ belongs to R_i after T is applied, using prerelations we can test, *before* T is committed, whether a given tuple will be in the database afterward. We need the set Γ of terms to be able to define transactions that extend the active domain of a database (for example, insertion of a new tuple may extend the domain). The set $\Gamma(D)$ is a superset of the active domain of $T(D)$, and the formulae pre_i^T are used to determine when a tuple from $\Gamma(D)$ belongs to the ith relation of $T(D)$.

Prerelations can be viewed as providing the ability to verify local properties of databases as well as global properties. Prerelations have been used extensively in [32]. If $\mathscr{L}$ is pure first-order logic (in particular, Ω is empty and the only terms are variables), then the notion of having prerelations reduces to the familiar notion of a first-order definable transaction.

For all the languages we shall consider in this paper it is the case that $\mathscr{PR}(\mathscr{L}) \subseteq \mathscr{WPC}(\mathscr{L})$. Indeed, to construct $\mathsf{wpc}(T, \alpha)$ for any $T \in \mathscr{PR}(\mathscr{L})$ we can use Γ and pre_i^T to simulate the new database state after the transaction, and then substitute all symbols for R_i in α by the formulae defining the new state. It is easy to see that the power of first-order logic is sufficient for doing this. Later in the paper we examine this inclusion and show that for most languages of interest it is strict.

3. WEAKEST PRECONDITIONS

The goal of this section is to study weakest preconditions for first-order logic and its extensions. We first prove that a language that can express (deterministic) transitive closure or the same-generation query cannot have weakest preconditions over FO. We then extend this result to more powerful specification languages, and finally prove that no transaction language captures the class of transactions having weakest preconditions over first-order logic.

We need a few technical definitions. Given a graph $G = \langle V, E \rangle$, its *deterministic transitive closure* (cf. [23]), $dtc(G)$, is the graph $\langle V, E' \rangle$ where $(x, y) \in E'$ iff $(x, y) \in E$ or there is a sequence of nodes $x = x_1, x_2, ..., x_n = y$ such that each $(x_i, x_{i+1}) \in E$, and furthermore, each x_i has out-degree 1, $i = 1, ..., n-1$. The transitive closure of a graph is denoted by $tc(G)$.

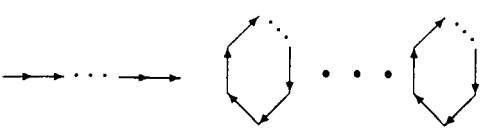
Given a graph $G = \langle V, E \rangle$, the *same-generation* query returns the graph $sg(G) = \langle V, E' \rangle$ where $(x, y) \in E'$ iff there is a node $v \in V$ and two walks in G from v to x

and from v to y that have the same length. Usually the same-generation query is asked for trees. So we define the class of queries

$$\mathscr{SG}_{\text{tree}} = \{q \mid \text{if } G \text{ is a tree, then } q(G) = sg(G)\}.$$

In particular, $sg \in \mathscr{SG}_{\text{tree}}$.

A *chain* is a graph of the form $\{(x_1, x_2), ..., (x_{n-1}, x_n)\}$ where all x_is are distinct. A *simple cycle* is a graph of the form $\{(x_1, x_2), ..., (x_{n-1}, x_n), (x_n, x_1)\}$ where all x_is are distinct. A *chain-and-cycle graph*, or a *C&C-graph* consists of $n \geqslant 1$ connected components, of which exactly one is a chain, and others (perhaps zero) are simple cycles. An example of a C&C-graph is shown below.



LEMMA 1. *The class of C&C-graphs is first-order definable. Neither chains nor cycles are first-order definable.*

Proof. A graph is a C&C-graph iff it has exactly one root (node with in-degree zero), exactly one endpoint (node with out-degree zero), the root has out-degree 1, the endpoint has in-degree one, and all other nodes have both in- and out-degrees 1. That is,

$$\begin{aligned}
&\forall x\, \forall y\, \forall z.\, E(x, y) \wedge E(x, z) \rightarrow z = y && \text{outdegrees are at most 1}\\
\wedge\ &\forall x\, \forall y\, \forall z.\, E(y, x) \wedge E(z, x) \rightarrow z = y && \text{indegrees are at most 1}\\
\wedge\ &\exists! x\, \forall y.\, \neg E(y, x) && \text{unique root}\\
\wedge\ &\exists! x\, \forall y.\, \neg E(x, y) && \text{unique endpoint}
\end{aligned}$$

defines the C&C-graphs.

That chains and cycles are not first-order definable can be proved straightforwardly using standard techniques (cf. [18, 27]). ▮

For the rest of the paper, $\Psi_{\text{c\&c}}$ denotes the first-order sentence in the language of $E(\cdot, \cdot)$ above that defines C&C-graphs.

3.1. *Languages Not Verifiable over FO*

In this subsection we prove part of Theorem B from the Introduction. That is,

THEOREM 2. *Assume that $\mathscr{TL}$ can express one of the following*:

- *transitive closure*;
- *deterministic transitive closure*;
- *any query from $\mathscr{SG}_{\text{tree}}$.*

Then $\mathscr{TL}$ is not verifiable over FO.

Proof. Follows from the following three claims.

Claim 1. $tc \notin \mathscr{WPC}(\mathrm{FO})$.

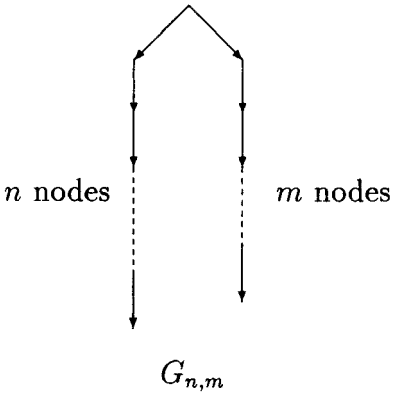
Proof of Claim 1. Assume $tc \in \mathscr{WPC}(\mathrm{FO})$ and let $\alpha \equiv \forall x\, \forall y.E(x, y)$. Then a graph satisfies $\mathsf{wpc}(tc, \alpha)$ iff it is connected, which means connectivity is FO-definable. This contradiction proves $tc \notin \mathscr{WPC}(\mathrm{FO})$.

Claim 2. $dtc \notin \mathscr{WPC}(\mathrm{FO})$.

Proof of Claim 2. Assume $dtc \in \mathscr{WPC}(\mathrm{FO})$. Let $\alpha \equiv \forall x\, \forall y.(x \neq y \to E(x, y) \vee E(y, x))$, and let $\beta = \mathsf{wpc}(dtc, \alpha)$. Define $\gamma \equiv \Psi_{\mathrm{c\&c}} \wedge \beta$. Then $G \models \gamma$ iff $G \models \Psi_{\mathrm{c\&c}}$ and in $dtc(G)$ there are no two unconnected nodes. For any graph G satisfying $\Psi_{\mathrm{c\&c}}$ we have $tc(G) = dtc(G)$, so $G \models \gamma$ iff G is a chain. However, it is known (see [13, 27]) that testing for chain is not FO-definable. Thus, $dtc \notin \mathscr{WPC}(\mathrm{FO})$.

Claim 3. If $q \in \mathscr{SG}_{\mathrm{tree}}$, then $q \notin \mathscr{WPC}(\mathrm{FO})$.

Proof of Claim 3. We define a family of graphs, $\{G_{n,m} \mid n, m \geqslant 1\}$ that will be used in this proof as well as in the proof of Theorem 3. A graph $G_{n,m}$ is shown on the picture below. It is a tree, with two branches, each being a chain. The subtree rooted at one of the root's children is an n-node chain, and the other is an m-node chain.



Now assume that $q \in \mathscr{WPC}(\mathrm{FO})$. If G is a tree, then $q(G)$ consists of a number of connected components, each of them being a complete graph. In particular, if for some node x there is no other node in the same generation as x, then x is an isolated node in $q(G)$. Let α_i, where $i \geqslant 1$, be a first-order sentence saying that there exist exactly i isolated nodes (i.e., those with a loop and no other incoming or outgoing edge). Let $\beta_i = \mathsf{wpc}(q, \alpha_i)$. Then it is easy to see that $G_{n,m} \models \beta_i$ iff $|n - m| = i - 1$. Thus, for any $n \geqslant 1$, $G_{n-1,n+1} \models \beta_3$ and $G_{n,n} \models \neg\beta_3$.

To prove Claim 3, we now have to show that for any first-order sentence γ there exists a number N such that for all $n > N$, $G_{n-1,n+1} \models \gamma$ iff $G_{n,n} \models \gamma$. To prove this statement, we use Hanf's technique modified for the finite case by [17]. Given a graph G, an r-neighborhood of a node a, $N_r(a)$, is the subgraph of G on the set of

all nodes reachable from a by unordered paths of length at most r. An r-type of a is the isomorphism type of $N_r(a)$, with a as a distinguished node. It can be seen from the structure of the graphs $G_{n,m}$ that for every r, and every $n > 2r+1$, the graphs $G_{n,n}$ and $G_{n-1,n+1}$ have the same number of neighborhoods of every given r-type.

Now if γ of quantifier rank k is given, take r to be 3^k. Then for any $n > 2r+1$, $G_{n,n}$ and $G_{n-1,n+1}$ have the same number of r-neighborhoods that realize each r-type. That is, using the terminology from [17], these two graphs are r-equivalent. According to [17], 3^k-equivalent structures cannot be distinguished by sentences of quantifier rank k; hence $G_{n,n}$ and $G_{n-1,n+1}$ cannot be distinguished by γ. Consequently, no first-order sentence distinguishes the families $\{G_{n,n} \mid n > 1\}$ and $\{G_{n-1,n+1} \mid n > 1\}$. This finishes the proof of Claim 3 and the theorem. ∎

3.2. *Weakest Preconditions for More Powerful Logics*

Here we extend the result of the previous subsection for more powerful logics defined in Section 2. By doing so, we complete the proof of Theorem B from the Introduction.

THEOREM 3. *Let $\mathscr{L}$ be either $\mathrm{FO}_{\mathrm{count}}$, or $\mathrm{FO}_c(\Omega)$ for an arbitrary Ω, or monadic Σ^1_1. Then neither transitive closure, nor deterministic transitive closure, nor any query in $\mathscr{S}\mathscr{G}_{\mathrm{tree}}$ is verifiable over $\mathscr{L}$.*

Proof. We start by showing that the transitive closure and deterministic transitive closure are not verifiable over $\mathscr{L}$. The proof for $\mathrm{FO}_{\mathrm{count}}$ and $\mathrm{FO}_c(\Omega)$ is essentially the same as the proof of Theorem 2, if we can show that connectivity and testing for chain are not definable in those logics. For $\mathrm{FO}_{\mathrm{count}}$ this follows immediately from [27] (or [13]), and for $\mathrm{FO}_c(\Omega)$ this follows from the result of [7] that generic Boolean queries definable in $\mathrm{FO}_c(\Omega)$ are definable in relational calculus with the order relation, since directed connectivity is not definable in the latter [36]. That testing for chain is not definable in first-order logic with built-in order relation follows from [13], which shows that this problem is first-order complete for deterministic logspace.

In the case of monadic Σ^1_1, the proof for *tc* is the same as before, since connectivity is not expressible in monadic Σ^1_1 [5]. For *dtc*, we have to show that testing whether a graph is a chain is not monadic Σ^1_1-definable. It was shown in [17] that for any monadic Σ^1_1 sentence ψ there exists a number n such that ψ can not distinguish C^1_n, one cycle of length $2n$, from C^2_n, two cycles of length n. However, C^1_n can be distinguished from C^2_n for any $n > 2$ if testing for chain is available. Indeed, consider the sentence $\gamma \equiv \forall x\, \forall y.E(x, y) \to \mathit{chain}[E'_{xy}/E]$. Here *chain* is the sentence that tests whether a graph is a chain, and everywhere in this sentence we replace the basic predicate $E(z, v)$ with $E'_{xy}(z, v) \equiv E(z, v) \wedge (\neg(x = z) \vee \neg(y = v))$. Then γ holds iff for every edge (x, y), the graph resulting from removing this edge is a chain. Such a sentence γ clearly distinguishes C^1_n from C^2_n. This finishes the proof for the case of (deterministic) transitive closure.

Now let $q \in \mathscr{S}\mathscr{G}_{\mathrm{tree}}$; we show that q is not verifiable over $\mathscr{L}$. When $\mathscr{L} = \mathrm{FO}_{\mathrm{count}}$, we can apply the proof of Claim 3 in Theorem 2, since, by the result of [30] (see

also [15]), for each k it is possible to find a number r such that any two structures that realize the same number of all r-neighborhoods cannot be distinguished by a $\mathrm{FO}_{\mathrm{count}}$ sentence of quantifier rank k.

For $\mathrm{FO}_c(\Omega)$, first define an order relation $\prec$ on $\mathscr{U}$ that is isomorphic to ω (that is, the usual order on $\mathbb{N}$). Now we show that q is not verifiable over $\mathrm{FO}_c(\Omega \cup \{\prec\})$. Assume it is verifiable. Consider α_2 from the proof of Theorem 2 (asserting that there are two isolated points) and let $\beta_2 = \mathsf{wpc}(q, \alpha_2)$. Note that $G_{n,m} \models \beta_2$ iff $n-m=1$ or $m-n=1$.

According to [7, Proposition 3], there exists an infinite set $X \subseteq \mathscr{U}$ and a $FO(\prec)$ sentence γ_2 such that for all graphs G whose sets of nodes are in X, we have $G \models \gamma_2$ iff $G \models \beta_2$. In particular, for any graph of the form $G_{n,m}$ with nodes from X, $G_{n,m} \models \gamma_2$ iff $|n-m|=1$. Now consider an arbitrary graph G of the form $G_{n,m}$ with the set of nodes A. Let $f: A \to X$ be a monotone injective map from A to X, and let $f(G)$ be the image of G under this map. Then, since γ_2 is a $FO(\prec)$ sentence, we get $G \models \gamma_2$ iff $f(G) \models \gamma_2$ iff $|n-m|=1$. Thus, the assumption that $q \in \mathscr{WPC}(\mathrm{FO}_c(\Omega))$ implies that in first-order logic with built-in order relation, we can test if $G_{n,m}$ satisfies $|n-m|=1$.

Now we are going to show that such a test is impossible. Assume that the first-order language contains only the symbol $\prec$ to be interpreted as a linear order. For each x, define

$$E_x(u,v) \equiv [(v \prec u) \wedge ((u \prec x) \vee (u = x)) \wedge (\forall z. \neg(v \prec z) \vee \neg(z \prec u))] \\ \vee [(u \prec v) \wedge ((x \prec u) \vee (u = x)) \wedge (\forall z. \neg(u \prec z) \vee \neg(z \prec v))].$$

That is, E_x defines a relation that is "successor backward" for elements under x and the successor relation (associated with the ordering) for elements above x. Such a relation is isomorphic to $G_{n,m}$ where n is the number of elements under x and m is the number of elements above x. Let $\gamma_2'(x)$ be a first-order formula in the language of $\prec$ obtained from γ_2 by replacing each occurrence of $E(u,v)$ with $E_x(u,v)$. Then a finite linear ordering $(L, <)$ satisfies $\exists x.\gamma_2'(x)$ iff there a "middle" element such that the number of elements above it is n and the number of elements below it is $n-1$ (or vice versa). This happens iff the size of the finite universe is even. However, since a $\mathrm{FO}(\prec)$ sentence of quantifier rank k cannot distinguish two linear orderings of size $> 2^k$ [34], we have a contradiction that shows that q is not verifiable over $\mathrm{FO}_c(\Omega)$.

Finally, we must show that q is not verifiable over monadic Σ_1^1. Assume such a q is verifiable. Let $\mathscr{G} = \{G_{n,n} \mid n \geqslant 1\}$. Then there exists a monadic Σ_1^1 sentence β such that, if G is a tree, then $G \models \beta$ iff $G \in \mathscr{G}$. To see this, note that the first-order sentence α_0 saying that a graph has one root of outdegree 2, two leaves on indegree 1, and that every other node has both in- and outdegree 1, defines a graph one of whose connected components is $G_{n,m}$ for some n, m, and all other connected components are cycles. Let α_1 be the first-order sentence $\exists! x.E(x,x) \wedge (\forall y.E(x,y) \vee E(y,x) \to x = y)$; that is, α_1 states that there exists a unique isolated point. Then, if G is a tree, then $G \models \alpha_0 \wedge \mathsf{wpc}(q, \alpha_1)$ iff $G \in \mathscr{G}$, since for any tree of the form $G_{n,m}$, where $n \neq m$, there exist at least two isolated points in $q(G_{n,m})$.

To complete the proof, we must show that there is no monadic Σ_1^1 sentence β such that, if G is a tree, then $G \models \beta$ iff $G \in \mathscr{G}$. Let *Tree* be the class of all trees. According to [16, Theorem 5.5], it suffices to show that for any positive integers c and k, the duplicator can win the (c, k) Ajtai–Fagin game for $\mathscr{G}$ and $Tree - \mathscr{G}$. The game is played as follows.

Step 1. The duplicator selects a graph $G \in \mathscr{G}$.

Step 2. The spoiler colors the nodes of G with c colors.

Step 3. The duplicator selects a graph $G' \in Tree - \mathscr{G}$ and colors its nodes with c colors.

Step 4. The spoiler and the duplicator play k rounds of the Ehrehfeucht–Fraïssé game on colored G and G'.

The winner is determined as the winner in Step 4. For more details on games, see [15, 16]. To determine a winner in Step 4, we shall use the criterion below, that follows immediately from Theorem 4.3 of [17]. We shall use the notation $G_1 \approx_{d,m} G_2$ if G_1 and G_2 are two colored graphs, and for every isomorphism type of a d-neighborhood of a node, either both graphs have the same number $n \leqslant m$ of nodes that realize this type, or both have at least m nodes that realize this type. Note that in a graph colored with c colors, a neighborhood of a point is structure in the vocabulary $\langle E, U_1, ..., U_c, a \rangle$, where E is the binary edge relation, U_is are unary relations that are interpreted as sets of nodes colored with the ith color, and a is a constant interpreted as a point around which the neighborhood is taken.

Claim 1. Let k be a positive integer. Then there exist positive integers d and m such that, whenever G_1 and G_2 are two colored trees of outdegree at most 2, and $G_1 \approx_{d,m} G_2$, then the duplicator has a winning strategy in the k-round Ehrehfeucht–Fraïssé game on G_1 and G_2.

Claim 1 is just a special case of Theorem 4.3 of [17], when the structures are trees, and the maximum degree is 2.

Before we prove that the duplicator has a winning strategy in the (c, k) Ajtai-Fagin game, we need the following combinatorial lemma.

LEMMA 4. *For every positive integers p and l, there exists a positive integer $N[p, l]$ such that, for any $N > N[p, l]$ and any partition of the set $\{1, ..., N\}$ into l sets, there exist two numbers i_1, i_2 that belong to the same class, such that for any $i_1 \leqslant i \leqslant i_2$, it is the case that i belongs to a partition class that has at least $p + i_2 - i_1$ elements.*

Proof of the lemma. If $l = 1$, the statement is trivial, so we assume $l > 1$. Let $f = \max\{p, l\}$. We claim that $N[p, l]$ can be taken to be $4f^4 + f(f+1) + 1$. There exists a number $N_0 \in [4f^4 + 1, 4f^4 + f(f+1) + 1]$ such that N_0 is divisible by l and $l+1$. Let $k = N_0/l$ and $s = N_0/l + 1$. Consider partitioning of $\{1, ..., N_0\}$ into l classes. Then at least one class, say X, contains k elements, $x_1, ..., x_k$. Let $d_i = x_{i+1} - x_i$, $i = 1, ..., k-1$. Then at most s of d_is are greater than or equal to $l+1$. Hence, at least $m_0 = N_0/(l(l+1)) - 1$ of d_is are less than $l+1$. Let $I = \{i \mid x_{i+1} - x_i \leqslant l\}$; we know that $|I| \geqslant m_0$.

If at least one $d_i = 1$, then we are done, since $k \geqslant p+1$. So assume that $d_i > 1$ for all $i \in I$. Assume that the conclusion of the lemma is false. Then, for every $i \in I$, there exists y_i such that $x_i < y_i < x_{i+1}$ and y_i belongs to a partition class that has fewer than $p + d_i$ elements. In particular, such a class has fewer than $p + l$ elements. Since there are l classes, we obtain that the number of such y_is is at most $l(p+l)$, which implies that the cardinality of I is at most $l(p+1)$, that is, $m_0 \leqslant l(p+l)$. This implies $N_0 \leqslant l^2(l+1)(p+l)+1$, which is easily seen to contradict the assumption that $N_0 \geqslant 4f^4+1$. This contradiction proves the lemma.

Using Lemma 4, we conclude the proof as follows. Given c and k, let d and m be given by Claim 1. Let $l = (2d+1)^c$, and let $N = N[m, l]$. Then in step 1 the duplicator selects $G_{n,n}$ where $n > N + 2(d+1)$. Let us call a node *internal* if it is at the distance at least $d+1$ from the root and the leaves. A d-neighborhood of such a node is a $2d+1$-element chain. The choice of n guarantees that both branches of $G_{n,n}$ have at least $N+1$ internal nodes.

Now the spoiler colors the graph with $G_{n,n}$ with c colors. Let G_1 be this colored graph. Note that there are at most $l = (2d+1)^c$ of isomorphism types of d-neighborhoods of internal nodes in G_1. Thus, coloring corresponds to partitioning the internal nodes into l classes, given by the isomorphism types of their d-neighborhoods. Applying Lemma 4 to $N+1$ internal nodes in one of the branches of $G_{n,n}$, we obtain that there exist two internal nodes a, b in the same branch, such that a and b have isomorphic d-neighborhoods, b is at a distance j from a, and for every node on a path from a to b, the isomorphism type of its d-neighborhood occurs at least $j+m$ times among the internal nodes of this branch.

Then in step 3, the duplicator selects the graph G' obtained from $G_{n,n}$ by collapsing b to a, that is, by removing all the nodes starting from the successor of a up to b. Note that $G' \in \mathit{Tree} - \mathscr{G}$. Then G' is colored by the duplicator, and the coloring in inherited from G_1. We call the resulting colored graph G_2.

It remains to show that $G_1 \approx_{d,m} G_2$, since this will imply that the duplicator can win the k-round Ehrenfeucht–Fraïssé game on G_1 and G_2 by Claim 1. First notice that there is no d-neighborhood that is present in one graph but is absent in the other. Furthermore, the only neighborhoods that have different number of realizers in G_1 and G_2 are those of the nodes removed in order to construct G_2 from G_1. For each of those neighborhood types, the number of nodes that realized them and that were removed, does not exceed j, and at the same time we know that each of those neighborhood types was realized at least $j+m$ times in G_1. Thus, each of those neighborhood types is realized at least m times in both G_1 and G_2, which proves $G_1 \approx_{d,m} G_2$.

This concludes proving that no $q \in \mathscr{SG}_{\text{tree}}$ is verifiable over monadic Σ^1_1, and completes the proof of the theorem. ∎

Since the sentences not having weakest preconditions in the proof of Theorem 3 are all in FO, we obtain:

COROLLARY 1. *Let $\mathscr{L}$ be either FO_{count}, or $\text{FO}_c(\Omega)$ for an arbitrary Ω, or monadic Σ^1_1. Then neither transitive closure, nor deterministic transitive closure, nor any query from $\mathscr{SG}_{\text{tree}}$ is in $\mathscr{WPC}(\mathscr{L}, \text{FO})$.* ∎

3.3. *The Structure of Transactions with Weakest Preconditions*

Here we prove the result showing that the class of verifiable transactions cannot be captured by a transaction language. (By capturing we mean that a language expresses exactly the transactions from a given set.)

THEOREM 5. *There is no transaction language that captures* $\mathscr{WPC}(\mathrm{FO})$. *Furthermore, for an arbitrary recursive signature* Ω, *there is no transaction language that captures* $\mathscr{WPC}(\mathrm{FO}_c(\Omega))$.

Proof. We first prove that no transaction language captures $\mathscr{WPC}(\mathrm{FO}_c(\Omega))$. We assume without loss of generality that the schema consists of a binary relational symbol $E(\cdot,\cdot)$. The proof below will automatically apply to any nonempty relational schema. Given a transaction language $\mathscr{TL}$, we show that it cannot capture $\mathscr{WPC}(\mathrm{FO}_c(\Omega))$. We will assume that the transactions in $\mathscr{TL}$ are enumerated as $\langle T_i\rangle_{i>0}$. This is possible because the syntax of any language is a recursive set. We let $\langle G_i\rangle_{i>0}$ enumerate the set of all graphs. All sentences of $\mathrm{FO}_c(\Omega)$ will be enumerated as $\phi_0, \phi_1, \ldots$, and we define the equivalence relation $=_n$ on graphs by $G =_n G'$ iff $G \models \phi_i \Leftrightarrow G' \models \phi_i$ for all $i \leqslant n$.

The idea of the proof is a diagonal argument. We will be building a transaction T, and for each m we will find a graph G such $T(G) \neq T_m(G)$. To ensure that the transaction T we build is in $\mathscr{WPC}(\mathrm{FO}_c(\Omega))$, we will ensure that for each positive integer n, there is an integer $P(n)$ such that for all $i > P(n)$, $T(G_i) =_n G_i$. That is, for each n and G, eventually T does not change the $=_n$ class. To see that this last suffices, we note the following.

LEMMA 6. *Suppose* T *is a computable transaction and there is a recursive function* P *on the integers such that for each positive integer* n *and for all* $i > P(n)$, $T(G_i) =_n G_i$. *Then* T *is in* $\mathscr{WPC}(\mathrm{FO}_c(\Omega))$

Proof of the lemma. We construct a weakest precondition algorithm for T as follows. Given a sentence ϕ, find an n such that $\phi = \phi_n$. Then apply the hypothesis to find an m $(= P(n))$ such that $T(G_i) =_n G_i$ for each $i > m$. Determine by testing which elements of $\{G_i : i \leqslant m\}$ have the property that $T(G_i) \models \phi$, and generate a sentence χ of $\mathrm{FO}_c(\Omega)$ that defines this finite set. Let ψ be a sentence that describes the set of all G_i with $i \leqslant m$. Now output the sentence $\Phi = \chi \vee (\neg\psi \wedge \phi)$.

We claim that Φ is a weakest precondition for ϕ. If we have G_i with $i \leqslant m$, then G_i satisfies Φ if and only if it satisfies χ, and by the definition of χ this is true if and only if $T(G_i) \models \phi$. If we have G_i with $i \geqslant m$, then G_i satisfies Φ if and only if it satisfies ϕ. Since $T(G_i) =_n G_i$ for $i \geqslant m$ and $\phi = \phi_n$, we get that for each $i \geqslant m$, $T(G_i) \models \phi \Leftrightarrow G_i \models \phi$. Combining the previous two sentences we get that for $i \geqslant m$, $T(G_i) \models \phi \Leftrightarrow G_i \models \Phi$, which completes the proof that Φ is a weakest precondition for ϕ. Lemma is proved. ∎

The construction, then, is intuitively as follows: start enumerating graphs until arriving at a graph G_j whose $=_1$ class has many elements in it. Let T be the identity on $G_1 \cdots G_{j-1}$, and then let $T(G_j)$ be an element $=_1$ equivalent to G_j but not equal to $T_1(G_j)$. Then enumerate all graphs above G_j until arriving at a G_k whose $=_2$ class has many elements in it. Let T be the identity on $G_{j+1} \cdots G_{k-1}$, and then let

$T(G_k)$ be an element $=_2$ equivalent to G_k but not equal to $T_2(G_k)$. Continue this process so as to diagonalize each T_i, while preserving the $=_n$ class for progressively larger n.

We now start the formal construction of T. We define a function $H(m, n)$ by letting $H(m, n)$ be lexicographically least pair $\langle i, j\rangle$ such that $m < i < j$ and $G_j =_n G_i$ and $G_j \neq G_i$.

Then let $P(n)$ and $Q(n)$ be defined inductively by $P(0) = Q(0) = 1$, and by letting $P(n+1)$ be the first component of $H(P(n), n)$, and $Q(n+1)$ be the second component of $H(P(n), n)$.

Finally, define the following transaction T. If $i = 1$ or i is not in the range of P, then $T(G_i) = G_i$. If $i > 1$ is in the range of P, consider the graph $G' = T_{P^{-1}(i)}(G_i)$. Since $G_{P(k)} \neq G_{Q(k)}$ for every $k > 0$, we know that for $j = Q(P^{-1}(i))$ it is the case that $G_i \neq G_j$. Now we define $T(G_i)$ in this case to be the one of G_i, G_j that is not equal to G', and if both are unequal to G', then we define $T(G_i)$ to be $G_{\min(i, j)}$.

The theorem now follows from the following three claims: the transaction T is a total recursive function, it is in $\mathscr{WPC}(\mathrm{FO}_c(\Omega))$, and it is different from each T_i in $\mathscr{TL}$.

To prove the first claim, notice that $=_n$ is an equivalence relation with only finitely many equivalence classes. We now show that $H(m, n)$ is a total function. That is, for every m and n there are $\langle i, j\rangle$ such that $m < i < j$ and $G_j =_n G_i$ and $G_j \neq G_i$. If the above were not the case, then fix a counterexample m and n. Then all G_is with $i > m$ are pairwise nonequivalent with respect to $=_n$. But this gives us infinitely many $=_n$ classes. From this, we see that $H(m, n)$ is recursive, since we can check each pair $\langle i, j\rangle$ in turn to see if $i \leqslant j$ and $G_j =_n G_i$ and $G_j \neq G_i$ until we find a pair for which this is true.

From this we see easily that P and Q are total recursive. Next we note that since both components of $H(m, n)$ are above m, the functions $P(m)$ and $Q(m)$ are (strictly) monotonically increasing. From this it follows that we can test recursively whether an i is in the range of P or not, that $P^{-1}(n)$ is well-defined for every n in the range, and that the range of P is infinite.

Note also that $G_{P(n+1)} =_n G_{Q(n+1)}$ and $P(n+1) \neq Q(n+1)$, using the definition of $H(m, n)$.

Since $P(n)$ and $Q(n)$ are distinct for every n, we get that for each m and i and each n it is true that one of $G_{P(n)}, G_{Q(n)}$ must be unequal to $T_m(G_i)$. Since P is monotonic, we can calculate the unique m such that $P(m) = i$ whenever i happens to be in the range of P. Together the last two sentences show that T is total and recursive.

We now turn to showing the second claim: that T admits a weakest precondition algorithm. We do this by verifying that for each positive integer n and for each $i > P(n)$, $T(G_i) =_n G_i$, and then citing Lemma 6 above.

Consider $T(G_i)$ for $i > P(n)$. If i is not in the range of P or $i = 1$, then $T(G_i) = G_i$ and so clearly $T(G_i) =_n G_i$. If i is in the range of P, then let $j = P^{-1}(i)$. Then $T(G_i)$ is one of either $G_{Q(j)}$ or $G_{P(j)}$, so in either case, $T(G_i) =_j G_{P(j)}$, since, as commented above $G_{Q(j)} =_j G_{P(j)}$. But then $T(G_i) =_j G_i$, since $P(j) = i$. Since P is monotonic and $i > P(n)$, we have $P^{-1}(i) > n$. This says that $j > n$, and hence $T(G_i) =_n G_i$, which

completes the proof of the second hypothesis of Lemma 6, and hence the proof that T is in $\mathscr{WPC}(\mathrm{FO}_c(\Omega))$

Finally we show that for each $m>0$, $T\neq T_m$. Let $i=P(m)$ (so $m=P^{-1}(i)$). Clearly, i is in the range of P (and the strict monotonicity of P guarantees $i>1$), so by definition, $T(G_i)$ has the property that $T(G_i)\neq T_{P^{-1}(i)}(G_i)$. But this means $T(G_i)\neq T_m(G_i)$, and this completes the proof that no transaction language captures $\mathscr{WPC}(\mathrm{FO}_c(\Omega))$.

To prove that no transaction language captures $\mathscr{WPC}(\mathrm{FO})$ we need a slight moldification of the proof above: we need to also ensure that T is generic in order for T to have a chance to be definable in pure first-order logic, while the above construction did not do anything to ensure genericity.

We again make use of an enumeration $\langle C_n\rangle$ of graphs such that no two graphs in the enumeration are isomorphic, and every graph is isomorphic to one of the C_n's. That is, the enumeration contains representatives for every isomorphism class of graphs. We can get such a recursive enumeration by enumerating the first graph G_1, then enumerating graphs until we come upon one nonisomorphic to any previously enumerated graph, etc.

Let I be the range of the enumeration $\langle C_m\rangle_{n\in\omega}$. For any graph G, let $[G]$ denote the (unique) graph C_n that is isomorphic to G. Given any mapping T from I to I, T expands to a mapping $[T]$ from graphs to graphs by letting $[T](G)=T([G])$.

Note that every finite collection of isomorphism classes can be expressed by a sentence of FO. Let $\langle\phi_i\rangle$, as before, enumerate the FO sentences. We can now define a transaction T from I to I exactly as we defined T before, but using C_i in place of G_i.

Now we claim that $[T]$ is in $\mathscr{WPC}(\mathrm{FO})$, but is unequal to any T_n. The proof that $[T]$ is recursive and diagonalizes the T_n's is exactly as before, using the fact that for each n there is an $=_n$ class that contains infinitely many nonisomorphic graphs.

To see that that $[T]$ is in $\mathscr{WPC}(\mathrm{FO})$, show that for each positive integer n and for each $i>P(n)$, $T(C_i)=_n C_i$: this follows exactly as before. A weakest precondition algorithm for $[T]$ is constructed as follows. Given a sentence ϕ, find an n such that $\phi=\phi_n$. Then find an m such that $T(C_i)=_n C_i$ for each $i>m$. Determine by testing which elements of $\{C_i: i\leqslant m\}$ have the property that $T(C_i)\models\phi$, and generate a sentence χ of FO that defines the union of the finite set of isomorphism classes of this set. Let ψ be a sentence of FO that describes the set of isomorphism classes of all C_i with $i\leqslant m$. Then exactly the same proof as above shows that $\chi\vee(\neg\psi\wedge\phi)$ is a weakest precondition for ϕ. This completes the proof of Theorem 5. ∎

As the last result of this section, we show that one cannot find a condition on degrees of nodes in graphs that describes the transactions with weakest preconditions. We are motivated by the following property of first-order queries, established in [27]. For a graph G, its *degree count* $dc(G)$ is the number of different in- and outdegrees of nodes of G. Then, for any first-order query q, $dc(q(G))$ is bounded by a number that depends only on q and the maximal possible in- or outdegree in G. For example, if G is an arbitrary binary tree, then an upper bound on $dc(q(G))$ is fully determined by q.

One may ask whether the class $\mathscr{WPC}(\mathrm{FO})$, which includes all first-order definable transactions, satisfies this property. We can show that this is not the case. In fact, a degree-based characterization is impossible for $\mathscr{WPC}(\mathrm{FO})$. Given any function $f: \mathbb{N} \to \mathbb{N}$ such that $f(n) > 0$, let $\mathscr{Q}_f$ be the family of graph queries q such that $dc(q(G)) \leqslant f(dc(G))$ for every G.

COROLLARY 2. *For any f, both $\mathscr{Q}_f - \mathscr{WPC}(\mathrm{FO})$ and $\mathscr{WPC}(\mathrm{FO}) - \mathscr{Q}_f$ are nonempty.*

Proof. Let q return the diagonal if its input graph is connected, and the complete graph on the input's nodes if it is not. Since connectivity is not first-order, $q \notin \mathscr{WPC}(\mathrm{FO})$. On the other hand, $q \in \mathscr{Q}_{\lambda x.1}$. Conversely, $T \in \mathscr{WPC}(\mathrm{FO})$ that will be described later in the proof of Theorem 7 is such that when its input is a chain, it constructs its transitive closure. Hence, T is not in $\mathscr{Q}_f$ for any f. ∎

4. PRECONDITIONS VS PRERELATIONS

The goal of this section is to examine the relationship between $\mathscr{WPC}(\mathrm{FO})$ and $\mathscr{PR}(\mathrm{FO})$. In particular, we find a transaction that separates them. However, we are not interested in just any transaction. Usually transformations expressed by database languages are required to satisfy certain properties. In particular, one is most often interested in transactions and queries computable in polynomial time. In addition, queries and transformations are sometimes required to be *generic*, that is, invariant under any permutation of the universe.

There is a large body of research dealing with languages capable of expressing polynomial-time generic database queries or transformations. Languages in which such transformations can be expressed include datalog with negation [1], languages based on structural recursion [10], loops [27], nondeterministic *reduce* operators [25], *while* queries [1], etc.

Hence, our first goal is to produce a generic polynomial time computable transaction that separates $\mathscr{WPC}(\mathrm{FO})$ from $\mathscr{PR}(\mathrm{FO})$. The transaction exhibited below can be expressed in all languages mentioned in the previous paragraph. The result below provides the proof of Theorem D from the Introduction. After proving Theorem 7 we proceed to show that $\mathscr{PR}(\mathrm{FO}_c(\Omega))$ can be captured by a transaction language, which gives us a separation result for $\mathscr{WPC}(\mathrm{FO}_c(\Omega))$ and $\mathscr{PR}(\mathrm{FO}_c(\Omega))$. We close the section by showing that $\mathscr{WPC}(\mathrm{FO}_c)$ and $\mathscr{PR}(\mathrm{FO}_c)$ coincide on generic transactions.

THEOREM 7. *There exists a polynomial-time computable generic transaction T such that*

$$T \in \mathscr{WPC}(\mathrm{FO}) - \mathscr{PR}(\mathrm{FO}).$$

Proof. Given a C&C-graph G, let $chain(G)$ be the connected component of G which is a chain. We consider the following transaction T on graphs $G = \langle X, E \rangle$:

$$T(G) = \begin{cases} tc(chain(G)) & \text{if} \quad G \models \Psi_{\mathrm{c\&c}} \\ \langle X, \{(x, x) \mid x \in X\} \rangle & \text{if} \quad G \models \neg \Psi_{\mathrm{c\&c}}. \end{cases}$$

It is not hard to see that T is generic and PTIME-computable. Now assume that T is in $\mathscr{P}\mathscr{R}(\mathrm{FO})$. Since Ω is empty, there exists a first-order formula $\beta(x, y)$ in free variables x and y such that (x, y) belongs to the output of $T(G)$ if $G \models \beta(x, y)$ when x and y are interpreted as elements of X. In particular, $T(G)$ can be calculated as the result of a first-order query. Notice that if G is a chain, then $T(G)$ is its transitive closure. However, according to the bounded degree property of first-order logic [27], there is no first-order definable query on graphs which computes transitive closure if its input happens to be a chain. Hence, $T \notin \mathscr{P}\mathscr{R}(\mathrm{FO})$.

The proof of $T \in \mathscr{W}\mathscr{P}\mathscr{C}(\mathrm{FO})$ is more involved. Recall the following definitions and results from [18]. In a graph, an r-neighborhood of a node x, denoted by $N_r(x)$, is the set of nodes of that graph that can be reached from x by a nonoriented path of length $\leqslant r$. By $\psi^{(r)}(x)$ we denote a formula in which x is the only free variable and all quantifiers are of form $\forall y \in N_k(x)$ or $\exists y \in N_k(x)$ where $k \leqslant r$. Let $d(x, y) > i$ denote that the distance between x and y is bigger than i. For every constant i, $d(x, y) > i$ can be expressed in first-order logic. It is known that every first-order sentence is equivalent to a Boolean combination of Gaifman sentences of the following form [18]:

$$\alpha \equiv \exists x_1 \cdots \exists x_s . \left(\psi^{(r)}(x_1) \wedge \cdots \wedge \psi^{(r)}(x_s) \wedge \bigwedge_{i, j = 1, \ldots, s, i \neq j} d(x_i, x_j) > 2r \right). \quad (1)$$

Hence, it is enough to show that with respect to T, every α of form (1) has a weakest precondition. Without loss of generality, assume $s \geqslant 1$ (otherwise α is constant and the proof is trivial).

Now observe that $T(G)$ has the following property. It is either a diagonal graph (having loops on all its nodes and nothing else), in which case $N_r(x) = \{x\}$ for any x, or it is a finite nonreflexive linear order (transitive closure of a chain) in which case $N_1(x)$ contains all nodes for any x.

It is enough to find a sentence β such that $G \models \beta$ iff $T(G) \models \alpha$ provided that G is a graph that satisfies $\Psi_{\mathrm{c\&c}}$. Since all neighborhoods in diagonal graphs consist of one point, we have two cases. If $\psi^{(r)}(x)$ holds in the neighborhood consisting of x only, where the corresponding graph has a loop on x, then α on the diagonal graph reduces to the statement that there exist at least s distinct nodes. Hence, in this case a precondition for α is $(\Psi_{\mathrm{c\&c}} \wedge \beta) \vee (\neg \Psi_{\mathrm{c\&c}} \wedge \mu_s)$ where μ_s is a first-order sentence saying that there are at least s distinct nodes. If $\psi^{(r)}(x)$ fails in the one-point neighborhoods, then $\Psi_{\mathrm{c\&c}} \wedge \beta$ is a precondition.

Now, assuming that G satisfies $\Psi_{\mathrm{c\&c}}$, we construct a precondition for α by considering the following cases.

Case 1: $s > 1$ *and* $r \geqslant 1$. Then α cannot be satisfied on a finite linear order because this would imply the existence of two nodes such that the distance between them is at least 2. Hence, for case 1 we take β to be *false*.

Case 2: $s \geqslant 1$, $r = 0$. Here we consider two subcases. First, suppose that $\psi^{(r)}(x)$ fails in one-point neighborhoods where the only edge is the loop on that point. Then α becomes *false* and hence β is taken to be *false*. Second, assume that $\psi^{(r)}(x)$

is satisfied in such one-point neighborhoods. Then α becomes equivalent to $\exists x_1 \cdots \exists x_s.(\wedge_{i,\, j=1,\, \ldots,\, s,\, i \neq j} d(x_i, x_j) > 0)$. Thus, it will hold in $T(G)$ iff the chain part of G has at least s points. The sentence p_s verifying this property is given below; then β is taken to be p_s.

$$p_s \equiv \exists y_1 \cdots \exists y_s.((\forall z. \neg E(z, y_1)) \wedge E(y_1, y_2) \wedge \cdots E(y_{s-1}, y_s))$$

Case 3: $s = 1$, $r \geqslant 1$. Then α becomes $\exists x.\psi^{(r)}(x)$. Since we only consider α on graphs of the form L_n, a linear order on n nodes, an r-neighborhood of any $x \in L_n$ is the whole of L_n, which means that there exists a (nonlocal) sentence α' that is equivalent to α on all L_n (it is obtained from α by removing range restriction from local quantifiers). Now we need the following claim, proved in [20, 34] by using Ehrenfeucht–Fraissé games.

Claim. For every first-order sentence α', it is possible to find $n \in \mathbb{N}$ such that either for all $m \geqslant n$: $L_m \models \alpha'$, or for all $m \geqslant n$: $L_m \models \neg\alpha'$.

Using this claim, we can find the precondition β as follows. Look at α' and find the n as stated above. Let $N_1 \cup N_2$ be the partition of $\{0, 1, \ldots, n-1\}$ such that for all $i \in N_1$: $L_i \models \alpha'$ and for all $j \in N_2$: $L_j \models \neg\alpha'$. Let p_i^0 be the sentence saying that the chain part of the input has precisely i elements; it can be defined as $p_i \wedge \neg p_{i+1}$. Then β is found as

$$\beta \equiv \left(\bigvee_{i \in N_1} p_i^0 \wedge \bigwedge_{j \in N_2} \neg p_j^0 \right) \vee p_n$$

if for all $m \geqslant n$: $L_m \models \alpha'$, and

$$\beta \equiv \left(\bigvee_{i \in N_1} p_i^0 \wedge \bigwedge_{j \in N_2} \neg p_j^0 \right) \wedge \neg p_n$$

if for all $m \geqslant n$: $L_m \models \neg\alpha'$.

Finally, we observe that the weakest precondition β given above is computable from α. Indeed, it is possible to verify if $\psi^{(r)}(x)$ holds in one-node diagonal graphs, and for case 3 it is known [20] that n can be taken to be 2^k, where k is the quantifier rank of α'. Hence, all that is needed for constructing the precondition in case 3 is verifying if α' holds for L_j, where $j = 0, \ldots, 2^k + 1$. The theorem is proved. ∎

The next result shows that the preconditions constructed by the algorithm in the proof of Theorem 7 are rather complex.

COROLLARY 3. *Let T be the transaction from Theorem* 7. *Then, for any $n > 1$, there exists a sentence α of quantifier rank n, such that if m is the quantifier rank of* $\mathsf{wpc}(T, \alpha)$ *computed by the algorithm of Theorem* 7, *then $m \geqslant 2^n$.*

Proof. We may take α to be already a Gaifman sentence. Then, according to the algorithm of Theorem 7, one of the Boolean components of $\mathsf{wpc}(T, \alpha)$ is p_{2^n} which means that the quantifier rank of $\mathsf{wpc}(T, \alpha)$ is at least 2^n. ∎

We do not know if a better algorithm is possible. In fact, it remains open whether polynomial time weakest precondition algorithms exist for any transaction in $\mathscr{WPC}(\mathrm{FO})$ that is not first-order definable.

The separation result continues to hold for more powerful logics, although it is impossible to find a separating transaction that will have such a nice description as above. First we observe the following.

PROPOSITION 3. *For any Ω, $\mathscr{PR}(\mathrm{FO}_c(\Omega))$ can be captured by a transaction language.*

Proof. Since all functions and predicates in Ω are recursive, the set $\Gamma(D)$ is computable. The transactions in the language capturing $\mathscr{PR}(\mathrm{FO}_c(\Omega))$ are given by tuples $(\Gamma, \alpha_1, ..., \alpha_k)$ where Γ is a finite set of terms, and each α_i has n_i free variables. The generated transaction is defined to be such that the tuple $(\Gamma, \alpha_1, ..., \alpha_k)$ constitutes a prerelation. This fully determines the transaction, and the definition of prerelation implies that such a language captures $\mathscr{PR}(\mathrm{FO}_c(\Omega))$. ∎

Sequential languages that define the same classes of transactions can be found in the literature, see for example [32, 33].

From Proposition 3 and Theorem 5 we obtain:

COROLLARY 4. *For any Ω, there are transactions in $\mathscr{WPC}(\mathrm{FO}_c(\Omega)) - \mathscr{PR}(\mathrm{FO}_c(\Omega))$.*

However, in contrast to pure first-order logic, there are no generic separating transactions if constants are present in the language.

PROPOSITION 4. *There are no generic transactions in $\mathscr{WPC}(\mathrm{FO}_c) - \mathscr{PR}(\mathrm{FO}_c)$.*

Proof. As before, we assume without loss of generality that we are dealing with the language of graphs. Let T be a generic transaction in $\mathscr{WPC}(\mathrm{FO}_c)$. Our goal is to show that T admits prerelations. First notice that for any graph G with the set of nodes X, the set of nodes of $T(G)$, Y, is a subset of X. Indeed, assuming the existence of $a \in Y - X$, pick any $b \in \mathscr{U} - X$, $b \neq a$, and consider a permutation π that permutes a and b and leaves everything else fixed. Since $G = \pi(G)$, by genericity we obtain that b is also a node of $T(G)$. Since $\mathscr{U}$ is infinite, this means that Y is infinite, which is impossible. Thus, $Y \subseteq X$. Consequently, defining prerelations for T, we can take Γ to contain a single variable. In other words, it is enough to find a formula $\beta(x, y)$ such that $G \models \beta(a, b) \Leftrightarrow (a, b) \in T(G)$ for every $a, b \in X$.

Let c and d be arbitrary distinct elements of the universe, and let Ψ be a FO_c-sentence expressing $\mathsf{wpc}(T, E(c, d))$. Let Φ be a sentence expressing $\mathsf{wpc}(T, E(c, c))$. Now we consider formulae $\psi(x, y)$ and $\phi(x)$ in two and one free variables respectively that are obtained from Ψ and Φ by replacing each occurrence of c by x and each occurrence of d by y. Define

$$\gamma(x, y) \equiv (x = y \wedge \phi(x)) \vee (\neg(x = y) \wedge \psi(x, y)).$$

Let C be the set of all constants that occur in γ. Note that $c, d \notin C$.

We now claim that for any graph G whose set of nodes X is disjoint from C, the formula γ defines prerelations. That is, $G \models \gamma(a, b)$ iff $(a, b) \in T(G)$ for every

$a, b \in X$. Consider the case $a \neq b$; the case of $a = b$ is similar. Let π be a permutation of the universe that permutes a and c, b and d, and leaves everything else fixed. Then by genericity $(a, b) \in T(G)$ iff $(c, d) \in T(\pi(G))$. Since $\Psi = \psi(c, d)$ is a weakest precondition, $(c, d) \in T(\pi(G))$ iff $\pi(G) \models \psi(c, d)$, which is equivalent to $G \models \psi(a, b)$, since π leaves everything outside of $\{a, b, c, d\}$ fixed and ψ does not use any of these constants. This proves our claim.

Now consider a new formula $\beta(x, y)$ obtained from $\gamma(x, y)$ as follows: any atomic subformula involving a constant k from C (which is of the form $z = k$ or $E(k, z)$ or $E(z, k)$ where z is either x, or y, or a bound variable) is replaced by *false*. Note that β is now a FO-formula. Then for any graph G with the node set X such that $X \cap C = \varnothing$, we have $G \models \gamma(a, b)$ iff $G \models \beta(a, b)$ for any $a, b \in X$.

Let G now be an arbitrary graph on X. Consider $a, b \in X$ and any permutation π of $\mathscr{U}$ such that $\pi(X) \cap C = \varnothing$ (it exists because $\mathscr{U}$ is infinite). Since β is a FO-formula (in particular, does not mention any constants), $G \models \beta(a, b)$ iff $\pi(G) \models \beta(\pi(a), \pi(b))$. Since $\pi(X) \cap C = \varnothing$, $\pi(G) \models \beta(\pi(a), \pi(b))$ iff $(\pi(a), \pi(b)) \in T(\pi(G))$ which, by genericity, is equivalent to $(a, b) \in T(G)$. Thus, $G \models \beta(a, b)$ iff $(a, b) \in T(G)$, proving that T admits prerelations. ∎

Since the formula β constructed in the proof above does not mention any constant, we can also conclude that there are no generic transactions in $\mathscr{WPC}(\mathrm{FO}_c) - \mathscr{PR}(\mathrm{FO})$.

5. ROBUSTLY VERIFIABLE TRANSACTIONS

Summarizing the results of the previous sections, we can say that many non-first-order transactions do not have weakest preconditions. Nevertheless, it is possible to find some non-first-order transactions that do have them. However, there is a gap between FO and FO_c. While we can find "nice" transactions that are not first-order definable and have weakest preconditions over FO, no such transactions exist when FO is replaced by FO_c. In particular, the Datalog$^\neg$-definable transaction that provided the separation for $\mathscr{WPC}(\mathrm{FO})$ and $\mathscr{PR}(\mathrm{FO})$ no longer has preconditions if the language is FO_c. While this follows from Proposition 4, we give a direct proof that illustrates the problems caused by the presence of constants.

PROPOSITION 5. *The transaction $T \in \mathscr{WPC}(\mathrm{FO}) - \mathscr{PR}(\mathrm{FO})$ from the proof of Theorem 7 is not in $\mathscr{WPC}(\mathrm{FO}_c)$.*

Proof. Let c be a constant symbol for some element of the universe. Assume that $T \in \mathscr{WPC}(\mathrm{FO}_c)$. Let α be a formula in the language that consists of E and c defined as $(\exists x\, \exists y.E(x, y) \wedge (x \neq y)) \wedge (\forall x. \neg E(x, c) \wedge \neg E(c, x))$; this formula says that the graph has at least one edge which is not a loop, and c is not one of its nodes. Let $\beta = \mathsf{wpc}(T, \alpha)$ and $\gamma \equiv \beta \wedge (\exists x.E(x, c) \vee E(c, x))$. Then $G \models \gamma$ iff $G \models \Psi_{\mathrm{c\&c}}$ (since $T(G)$ has an edge which is not a loop), c is one of its nodes but c is not a node in $T(G)$. That is, G is a C&C-graph but not a chain. In other words, for the graphs in which one node is the interpretation of c, testing for chains would be FO_c-definable. However, this is impossible, and hence $T \notin \mathscr{WPC}(\mathrm{FO}_c)$. ∎

Proposition 5 shows that a transaction verifiable over a language is not necessarily verifiable over even a simple extension of the language. This motivates the following definition. We call a transaction robustly verifiable if it is verifiable in every extension of the language. Below, we show that for FO_c as a specification language (in fact, even for $\mathrm{FO}_c(\Omega)$), a precise characterization of robust transactions is possible—these are exactly the $\mathrm{FO}_c(\Omega)$ definable ones.

Formally, a transaction T is *robustly verifiable* over $\mathrm{FO}_c(\Omega)$ if for every extension of Ω to Ω' with a recursive collection of recursive predicates and functions, it is the case that $T \in \mathscr{WPC}(\mathrm{FO}_c(\Omega'))$. A transaction language $\mathscr{TL}$ is robustly verifiable if all its transactions are.

The following result, together with Proposition 3, gives the proof of Theorem E. In what follows, we use $\mathscr{PR}(\mathrm{FO}_c(\Omega))$ to mean both the class of transactions that admit prerelations and the transaction language that captures this class. The existence of such a language was proved in Proposition 3.

THEOREM 8. *For every Ω, the transaction language $\mathscr{PR}(\mathrm{FO}_c(\Omega))$ is robustly verifiable over $\mathrm{FO}_c(\Omega)$, and every transaction robustly verifiable over $\mathrm{FO}_c(\Omega)$ is equivalent to a transaction in $\mathscr{PR}(\mathrm{FO}_c(\Omega))$.*

Proof. As in the previous results, without loss of generality we will assume that the signature consists of a single binary relation. We first show that transactions in $\mathscr{PR}(\mathrm{FO}_c(\Omega))$ are robustly verifiable. Let T be given by a set of terms Γ and formula $\phi_E(x, y)$ giving the prerelation of $E(x, y)$ with respect to T. Without loss of generality, we can assume all terms in Γ have the same arity n. We give an algorithm WPC transforming every formula $\gamma(\mathbf{z})$ in $\mathrm{FO}_c(\Omega')$ for any extension $\Omega' \supseteq \Omega$ into a $\mathrm{FO}_c(\Omega')$ formula $\mathrm{WPC}[\gamma](\mathbf{z})$ such that $G \models \mathrm{WPC}[\gamma](\mathbf{z})$ iff $T(G) \models \gamma(\mathbf{z})$.

The algorithm is recursive in the logical complexity of γ. If γ is quantifier-free, WPC proceeds by replacing every occurrence of $E(\tau_1, \tau_2)$ where each τ_i is a term, by the formula

$$\left(\exists \mathbf{y}_1 . \bigvee_{\tau \in \Gamma} \tau(\mathbf{y}_1) = \tau_1(\mathbf{z})\right)$$
$$\wedge \left(\exists \mathbf{y}_2 . \bigvee_{\tau \in \Gamma} \tau(\mathbf{y}_2) = \tau_2(\mathbf{z})\right)$$
$$\wedge \phi_E(\tau_1(\mathbf{z}), \tau_2(\mathbf{z})).$$

Then define

$$\mathrm{WPC}[\neg\gamma] = \neg \mathrm{WPC}[\gamma]$$
$$\mathrm{WPC}[\gamma_1 \vee \gamma_2] = \mathrm{WPC}[\gamma_1] \vee \mathrm{WPC}[\gamma_2],$$
$$\mathrm{WPC}[\gamma_1 \wedge \gamma_2] = \mathrm{WPC}[\gamma_1] \wedge \mathrm{WPC}[\gamma_2].$$

If $\gamma \equiv \exists x . \phi(x, \mathbf{z})$, define $\mathrm{WPC}[\gamma]$ to be

$$\exists \mathbf{y}_1 . \left(\bigvee_{\tau \in \Gamma} \phi'(\tau(\mathbf{y}_1), \mathbf{z}) \wedge \exists \mathbf{y}_2 . \left(\bigvee_{\rho \in \Gamma} \phi_E(\rho(\mathbf{y}_2), \tau(\mathbf{y}_1)) \vee \phi_E(\rho(\mathbf{y}_2), \tau(\mathbf{y}_1))\right)\right),$$

where $\phi'(x, \mathbf{z}) = \mathrm{WPC}[\phi]$. First, we must show that the algorithm WPC is correct. We will show that for each T in $\mathscr{PR}(\mathrm{FO}_c(\Omega))$, given by Γ, ϕ_E, and for each formula $\gamma(\mathbf{z})$ in $\mathrm{FO}_c(\Omega')$, and for each graph $G = \langle X, E \rangle$, and a vector $\mathbf{t}$ of elements from $\Gamma(G)$, the following holds:

$$G \models \mathrm{WPC}[\gamma](\mathbf{t}) \Leftrightarrow T(G) \models \gamma(\mathbf{t}).$$

Taking γ to be a sentence, we get as a corollary to this lemma that WPC witnesses the effective verifiability of $\mathscr{PR}(\mathrm{FO}_c(\Omega))$. The proof proceeds by induction on the complexity of γ.

If $\gamma(\mathbf{t})$ is atomic, then $\gamma(\mathbf{t})$ is either of the form $E(t_1, t_2)$, with each t_i an element of $\Gamma(G)$ or contains no occurrences of E. In the first case we get the required equivalence immediately from the definition of $\mathscr{PR}$:

$$T(G) \models E(t_1, t_2) \Leftrightarrow t_1, t_2 \in \Gamma(G)$$
$$\text{and} \qquad G \models \phi_E(t_1, t_2) \Leftrightarrow G \models \mathrm{WPC}[E(t_1, t_2)].$$

In the second case, $\mathrm{WPC}[\gamma] = \gamma$, so the equivalence is trivial.

The induction steps for boolean connectives go through routinely:

$$\begin{aligned}
& T(G) \models \gamma_1 \wedge \gamma_2 \\
\Leftrightarrow\ & T(G) \models \gamma_1 \text{and } T(G) \models \gamma_2 \\
\Leftrightarrow\ & G \models \mathrm{WPC}[\gamma_1] \text{ and } G \models \mathrm{WPC}[\gamma_2] \text{ by induction hypothesis} \\
\Leftrightarrow\ & G \models \mathrm{WPC}[\gamma_1] \wedge \mathrm{WPC}(\gamma_2) \\
\Leftrightarrow\ & G \models \mathrm{WPC}[\gamma_1 \wedge \gamma_2].
\end{aligned}$$

For the quantifier case $\gamma \equiv \exists x.\phi(x, \mathbf{z})$, let $\phi'(x, \mathbf{z}) = \mathrm{WPC}(\phi)$. For each G and for each $\mathbf{t} \subseteq \Gamma(G)$, we get

$$\begin{aligned}
& T(G) \models \exists x.\phi(x, \mathbf{t}) \\
\Leftrightarrow\ & \exists v \in \Gamma(G).\, T(G) \models \phi(v, \mathbf{t}) \\
\Leftrightarrow\ & \exists v \in \Gamma(G).\, G \models \phi'(v, \mathbf{t}) \text{ by induction} \\
\Leftrightarrow\ & \exists v \in \Gamma(G), \exists v' \in \Gamma(G) \text{ such that } G \models \phi'(v, \mathbf{t}) \text{ and} \\
& \qquad ((v, v') \in T(G) \text{ or } (v', v) \in T(G)) \\
\Leftrightarrow\ & G \models \exists \mathbf{v}.\left(\bigvee_{\tau \in \Gamma} \phi'(\tau(\mathbf{v}), \mathbf{t}) \wedge \exists \mathbf{v}'.\, \bigvee_{\rho \in \Gamma} (\phi_E(\tau(\mathbf{v}), \rho(\mathbf{v}')) \vee \phi_E(\rho(\mathbf{v}'), \tau(\mathbf{v}))) \right) \\
\Leftrightarrow\ & G \models \mathrm{WPC}[\gamma](\mathbf{t})
\end{aligned}$$

which completes the proof of correctness of the WPC algorithm.

For the other direction, assume that T is a robustly verifiable transaction. We will prove that T admits a formula $\phi_E(x, y)$ and a set of terms Γ defining the

prerelation as in the definition of $\mathscr{PR}(\mathrm{FO}_c(\Omega))$. That is, we will show that there are Γ and $\phi_E(x, y)$ such that $T(G)$ has edge set $\{(x, y) \mid x, y \in \Gamma(G), G \models \phi_E(x, y)\}$.

Let $\underline{\mathrm{P}}$ be an additional 4-place predicate symbol. Let L^+ be the set of all first-order formulae in the language with symbols for all constants in the universe, all elements of Ω, and the symbol $\underline{\mathrm{P}}$. Let L be $\mathrm{FO}_c(\Omega)$.

Our proof will proceed by attempting to inductively construct a recursive interpretation P of $\underline{\mathrm{P}}$ such that T does not admit definable weakest preconditions over the extension of L by P. Our attempt to construct such a "bad" P will be a straightforward diagonalization: we will pick a certain sentence *Hasmax* below, and for each possible sentence ϕ we will pick an initial segment of P designed to prevent ϕ from being the precondition of *Hasmax*. Since we know that T is in $\mathscr{WPC}(\mathrm{FO}_c(\Omega))$, this diagonalization cannot succeed; that is, at some point we have constructed a finite initial segment of P that cannot be extended further to prevent a particular sentence ϕ from being the precondition of *Hasmax*. The heart of the argument shows that we can use this initial segment and the sentence ϕ to get a first-order definition of T, thus showing that T is in $\mathscr{PR}(\mathrm{FO}_c(\Omega))$.

The proof is now organized as follows: first, we define the framework for our inductive diagonalization argument. We then show that if this diagonalization were to succeed, we would have a contradiction of robust verifiability for T. Finally, we show that from a partial function at which the diagonalization fails we can construct a prerelation for T.

First, we introduce some new notation. For $\phi(t)$ an L^+ formula, P a subset of $\mathscr{U} \times \mathscr{U} \times \mathscr{U} \times \mathscr{U}$, G a graph (that is, a graph on a finite subset of $\mathscr{U}$), and $t \in \mathscr{U}$, we write $\langle G, P\rangle \models \phi(t)$ if $G \models \phi(t)$, when $\underline{\mathrm{P}}$ is interpreted as P.

Let *P-maximal*(s, t) be the L^+ formula $(\exists x\, \exists y.E(x, y) \wedge \underline{\mathrm{P}}(x, y, s, t)) \wedge (\forall x\, \forall y.(E(x, y) \rightarrow \neg \underline{\mathrm{P}}(s, t, x, y)))$. Let

$$Hasmax \equiv \exists s\, \exists t.E(s, t) \wedge P\text{-}maximal(s, t)$$

That is, *Hasmax* is the L^+-sentence asserting that there is a P-maximal pair.

If p and q are partial functions, then $q \succ p$ means that q extends p as a function. For any set A, let χ_A be its characteristic function. We use this notation in the following definitions.

Suppose p is a finite partial function from $\mathscr{U}^4$ into $\{0, 1\}$. If ϕ is an L^+ sentence, say that:

[1] p is ϕ-good if for all finite P with $\chi_P \succ p$, and for all graphs G,

$$\langle G, P\rangle \models \phi \Leftrightarrow \langle T(G), P\rangle \models Hasmax$$

[2] say p is ϕ-poor if p is not ϕ-good.

[3] p is ϕ-terrible if there is a graph G such that for all finite P with $\chi_P \succ p$,

$$\neg(\langle G, P\rangle \models \phi \Leftrightarrow \langle T(G), P\rangle \models Hasmax). \qquad (*)$$

Intuitively, a finite partial function p used in the definitions above is to be interpreted as an approximation to a predicate P. The idea of the construction that will follow is that we want our final P not to satisfy $(\langle G, P\rangle \models \phi \Leftrightarrow \langle T(G), P\rangle \models \mathit{Hasmax})$ for any L^+ sentence ϕ. Then *Hasmax* will not have a definable weakest precondition in L^+, contradicting the assumption that T is robustly verifiable. We construct our solution by inductively constructing a sequence of approximations p_n that guarantee $(*)$ for each ϕ in turn. We can show that we can continue this construction successfully provided that the p we have constructed thus far is not ϕ-good. We show that if we are "stuck" with a ϕ-good p then T has a $\langle \Gamma, \phi_E(x, y)\rangle$ as required.

The proof of the theorem now falls into two cases.

Deriving a contradiction if the diagonalization succeeds. The diagonalization succeeding corresponds to the following being true:

Case 1. Every finite partial function p is ϕ-poor for each ϕ.

We will now show that this leads to a contradiction of robust verifiability.

PROPOSITION 6. *For every L^+ sentence ϕ and every ϕ-poor p, there exists a partial function p' with a finite domain such that $p' \succ p$ and p' is ϕ-terrible. Furthermore, p' can be found effectively given p.*

Proof. Since p is poor for ϕ, there is finite P with $\chi_P \succ p$, and a graph G such that $(*)$ holds. Since we can test property $(*)$, we can find such G and P effectively by listing out all pairs $\langle G, P\rangle$ until we find one such that $(*)$ holds.

Let p' be a finite partial function such that $p' \succ p$ and

$$\begin{aligned}(\tau_1(\mathbf{s}_1), ..., \tau_4(\mathbf{s}_4)) \in P &\to p'(\tau_1(\mathbf{s}_1), ..., \tau_4(\mathbf{s}_4)) = 1\\ (\tau_1(\mathbf{s}_1), ..., \tau_4(\mathbf{s}_4)) \notin P &\to p'(\tau_1(\mathbf{s}_1), ..., \tau_4(\mathbf{s}_4)) = 0.\end{aligned} \tag{2}$$

Here each $\tau_i(\mathbf{x})$, for $i \in \{1, 2, 3, 4\}$, ranges over terms contained in an atomic subformulae of ϕ or atomic subformulae of *Hasmax*, and $\mathbf{s}_i$ range over the vectors from $\mathscr{U}$ such that the length of $\mathbf{s}_i$ equals the arity of τ_i, and each element of $\mathbf{s}_i$ is a vertex of G or a vertex of $T(G)$.

Then p' will be ϕ-terrible. Condition (2) guarantees that for every P' with $\chi_{P'} \succ p'$, we have

$$\langle T(G), P'\rangle \models \mathit{Hasmax} \Leftrightarrow \langle T(G), P\rangle \models \mathit{Hasmax}$$

and also

$$\langle T(G), P'\rangle \models \phi \Leftrightarrow \langle T(G), P\rangle \models \phi.$$

This finishes the proof of the proposition. ∎

To finish Case 1, we inductively construct a function p defining a characteristic function which is ϕ-terrible for every ϕ. That is, let $\langle \phi_i\rangle$ be an enumeration of all the sentences in L^+, and let p_0 be empty, and p_{n+1} be any finite $p' \succ p_n$ such that

p' is ϕ_n-terrible. The construction of this sequence can be carried out effectively by Proposition 6. We can also ensure that $\bigcup_n p_n$ defines a total function by throwing the nth element of $\mathscr{U}$ into the domain of p_n if it is not there already.

Let P be the unique set such that $\chi_P \succ p_n$ for all n. Clearly, P is recursive, since the construction of p_n is recursive. If we unwind the definition of what it means to be terrible, we see that for P as above, there can be no sentence of L^+ that holds in a graph G exactly when $\langle T(G), P\rangle \models \mathit{Hasmax}$. Hence T does not admit weakest preconditions over L^+, and this contradicts the fact that T is robustly verifiable, completing the proof in Case 1.

Getting a prerelation if the diagonalization is stalled. This corresponds to the negation of Case 1. That is:

Case 2. There is a ϕ and a p that is ϕ-good.

Fix such a p and ϕ. Without loss of generality, we can assume that p is not empty. Let $\mathit{dom}(p)$ be the set of pairs (s, t) such that there exists $\mathbf{t}$ in the domain of p that contains both s and t.

Our goal will be to show that there is a formula $\phi_E(x, y)$ and a set of terms Γ which defines the edges of $T(G)$. Consider the set:

$$\begin{aligned}\mathit{EDEF} \stackrel{\text{def}}{=} \{\langle G, x, y\rangle \in DB \times \mathscr{U} \times \mathscr{U} \mid ((x, y) \in \mathit{dom}(p) \wedge (x, y) \in T(G))\\ \vee (\neg((x, y) \in \mathit{dom}(p)) \wedge \neg(\langle G, p_1 \cup ((\mathit{dom}(p) \cup \{(x, y)\}) \otimes \{(x, y)\})\rangle\\ \models \phi \leftrightarrow \langle G, p_1 \cup (\mathit{dom}(p) \otimes \{(x, y)\})\rangle \models \phi))\}.\end{aligned}$$

In the above, $A \otimes B = \{(x, y, w, z) \in \mathscr{U} \times \mathscr{U} \times \mathscr{U} \times \mathscr{U} \mid (x, y) \in A \wedge (w, z) \in B\}$, $p_1 = p^{-1}(1)$ and DB stands for the set of finite graphs whose nodes are from $\mathscr{U}$.

We will show that EDEF is definable by a formula $\phi_E(x, y)$ and set of terms Γ and that EDEF is exactly the prerelation for T of G. That is, we prove the following.

Claim 1. $\langle G, x, y\rangle \in \mathit{EDEF} \Leftrightarrow (x, y) \in T(G)$

Proof of Claim 1. First suppose $(x, y) \in T(G)$. We will show that $\langle G, x, y\rangle \in \mathit{EDEF}$.

If $(x, y) \in \mathit{dom}(p)$, then by the first disjunct in EDEF, $\langle G, x, y\rangle \in \mathit{EDEF}$, so assume $(x, y) \notin \mathit{dom}(p)$. Since $(x, y) \notin \mathit{dom}(p)$, the characteristic function of $p_1 \cup ((\mathit{dom}(p) \cup \{(x, y)\}) \otimes \{(x, y)\})$ gives an extension of p. Since p is ϕ-good, this means that

$$\begin{aligned}\langle G, p_1 \cup ((\mathit{dom}(p) \cup \{(x, y)\}) \otimes \{(x, y)\})\rangle \models \phi\\ \Leftrightarrow \langle T(G), p_1 \cup ((\mathit{dom}(p) \cup \{(x, y)\}) \otimes \{(x, y)\})\rangle \models \mathit{Hasmax}\end{aligned}$$

and

$$\begin{aligned}\langle G, p_1 \cup (\mathit{dom}(p) \otimes \{(x, y)\})\rangle \models \phi\\ \Leftrightarrow \langle T(G), p_1 \cup (\mathit{dom}(p) \otimes \{(x, y)\})\rangle \models \mathit{Hasmax}.\end{aligned}$$

Note that since $(x, y) \in T(G)$, we have for each $(s, t) \in dom(p)$,

$$\langle T(G), p_1 \cup ((dom(p) \cup \{(x, y)\}) \otimes \{(x, y)\}) \rangle \models \neg P\text{-}maximal(s, t)$$

and

$$\langle T(G), p_1 \cup (dom(p) \otimes \{(x, y)\}) \rangle \models \neg P\text{-}maximal(s, t).$$

Also note that since $(x, y) \notin dom(p)$, and $dom(p) \neq \varnothing$, (x, y) is P-maximal in the model $\langle T(G), p_1 \cup (dom(p) \otimes \{(x, y)\}) \rangle$. Furthermore, (x, y) will not be P-maximal in the model $\langle T(G), p_1 \cup ((dom(p) \cup \{(x, y)\}) \otimes \{(x, y)\}) \rangle$, since here we have $\underline{P}(x, y, x, y)$ holding.

Using the above we can verify that

$$\langle T(G), p_1 \cup (dom(p) \otimes \{(x, y)\}) \rangle \models Hasmax$$

and

$$\langle T(G), p_1 \cup ((dom(p) \cup \{(x, y)\}) \otimes \{(x, y)\}) \rangle \models \neg Hasmax$$

since (x, y) is the only P-maximal element in the first model, and there are no P-maximal elements in the second model. This shows that the second disjunct in *EDEF* holds. Hence, $\langle G, x, y \rangle \in EDEF$.

Now suppose $(x, y) \notin T(G)$. We will show $\langle G, x, y \rangle \notin EDEF$. Clearly $\langle G, x, y \rangle$ does not satisfy the first disjunct in *EDEF*. If $(x, y) \in dom(p)$, then it fails the second disjunct as well, and we are done. So suppose $(x, y) \notin dom(p)$. Once again we have

$$\langle G, p_1 \cup (dom(p) \otimes \{(x, y)\}) \rangle \models \phi$$
$$\Leftrightarrow \langle T(G), p_1 \cup (dom(p) \otimes \{(x, y)\}) \rangle \models Hasmax$$

and

$$\langle G, p_1 \cup ((dom(p) \cup \{(x, y)\}) \otimes \{(x, y)\}) \rangle \models \phi$$
$$\Leftrightarrow \langle T(G), p_1 \cup ((dom(p) \cup \{(x, y)\}) \otimes \{(x, y)\}) \rangle \models Hasmax.$$

To finish the proof, it therefore will suffice to prove that

$$\langle T(G), p_1 \cup (dom(p) \otimes \{(x, y)\}) \rangle \models Hasmax$$
$$\Leftrightarrow \langle T(G), p_1 \cup ((dom(p) \cup \{(x, y)\}) \otimes \{(x, y)\}) \rangle \models Hasmax.$$

But since (x, y) is not an edge of $T(G)$, any pair (s, t) is p-maximal in either of the above models if and only if it is p-maximal in $\langle T(G), p_1 \rangle$. Hence both the above statements are equivalent to $\langle T(G), p_1 \rangle \models Hasmax$. So $\langle G, x, y \rangle$ fails the second disjunct in *EDEF* as well as the first, and the proof of claim 1 is complete. ∎

To complete the proof, we need the following lemma.

LEMMA 9. *EDEF is definable by a formula* $\phi_E(x, y)$ *along with a set of terms* Γ. *That is, there is a formula* $\phi_E(x)$ *and a set of* $\mathrm{FO}_c(\Omega)$*-terms* Γ *such that*

$$\langle G, x, y\rangle \in EDEF \Leftrightarrow x, y \in \Gamma(G) \qquad \text{and} \qquad G \models \phi_E(x, y).$$

Proof of Lemma. $T \in \mathscr{WPC}(\mathrm{FO}_c(\Omega))$ allows us to write out the first disjunct of *EDEF* in L as

$$\bigvee_{(a, b) \in dom(p)} \mathsf{wpc}(T, \exists x\, \exists y.(x = a \wedge y = b)) \wedge (x = a \wedge y = b).$$

The second disjunct in *EDEF* is composed of two conjuncts. The first conjunct is just the negation of $\bigvee_{(a, b) \in dom(p)} (x = a \wedge y = b)$. To handle the second conjunct we first obtain a L-formula $\gamma(x, y)$ such that

$$\langle G, p_1 \cup (dom(p) \otimes \{(x, y)\})\rangle \models \phi \Leftrightarrow G \models \gamma(x, y)$$

by taking ϕ and replacing every atomic formula of the form $\underline{\mathrm{P}}(\tau_1, \tau_2, \tau_3, \tau_4)$ by $(\bigvee_{(a, b, c, d) \in p_1} (\tau_1 = a) \wedge (\tau_2 = b) \wedge (\tau_3 = c) \wedge (\tau_4 = d)) \vee (\bigvee_{(e, f) \in dom(p)} (\tau_1 = e) \wedge (\tau_2 = f) \wedge (\tau_3 = x) \wedge (\tau_4 = y))$.

One can analogously get a L-formula $\kappa(x, y)$ such that

$$\langle G, p_1 \cup ((dom(p) \cup \{(x, y)\}) \otimes \{(x, y)\})\rangle \models \phi \Leftrightarrow G \models \kappa(x, y).$$

Putting these two together we can express the second conjunct in L, which shows that *EDEF* is L-definable by some formula $\beta(x, y)$.

Notice that the above construction yields a $\beta(x, y)$ with the following properties. First, every atomic subformula containing one of the variables x or y is of the form $x = \tau(\mathbf{z})$ or $y = \tau(\mathbf{z})$ for some term τ, where $\mathbf{z}$ contains no variables free in β. Second, $\beta(x, y)$ has only finitely many (x, y) satisfying it for every graph G. Now we will use these properties of β to construct the required set Γ.

Assume *EDEF* is nonempty. We can write $\beta(x, y)$ as

$$Q_1 z_1 \cdots Q_n z_n. \bigvee_{i < m} B_i,$$

where each $Q_k z_k$ is a quantifier binding variable z_k, and each B_i is a conjunction of literals A_{ij} such that any A_{ij} with a variable x free is of the form $x = \tau(\mathbf{z})$ or $x \neq \tau(\mathbf{z})$, where x is not free in $\tau(\mathbf{z})$. Let Γ be all terms that appear in some B_i.

Claim 2. For every graph G and any (x, y) in $\mathscr{U} \times \mathscr{U}$, if $G \models \beta(x, y)$ then x and y are in $\Gamma(G)$.

Proof of Claim 2. If this is not the case, fix G, x_0 and y_0, with (WLOG) y_0 not in $\Gamma(G)$. Let y_1 be any other element of $\mathscr{U} - \Gamma(G)$. Then $G \models \beta(x_0, y_1)$, since for

each $\tau \in \Gamma$ and for any $z_1, \ldots z_k$ nodes of G, $G \models y_0 = \tau(z_1, \ldots, z_k)$ if and only if $G \models y_1 = \tau(z_1, \ldots, z_k)$. But this gives infinitely many edges satisfying $\beta(x, y)$, contradicting the properties of β. This proves Claim 2.

Claim 2 shows that Γ and $\phi_E = \beta$ satisfy the conclusion of Lemma 9. Now Lemma 9 along with Claim 1 complete the proof of the theorem. ▮

COROLLARY 5. *$\mathscr{PR}(\mathrm{FO}_c(\Omega))$ is the maximal robustly verifiable language over $\mathrm{FO}_c(\Omega)$.* ▮

6. CONCLUDING REMARKS

In this paper we have looked at the problem of verifying transaction safety *before* transactions are executed. The main results can be summarized as follows. If integrity constraints are specified in first-order logic, then for first-order transaction languages it is possible to compute both weakest preconditions and prerelations. However, if a transaction language has a mechanism for doing recursion, then such ability is generally lost. There are still some transactions that are not first-order definable but have weakest preconditions. However, if we require that the ability to calculate weakest preconditions be independent of extensions to the language, then we can use only languages that admit prerelations.

The last statement, which is a reformulation of Corollary 5 can be interpreted as follows. If we are interested in designing a "nice" transaction language that is verifiable over FO_c, and the verification algorithm can be extended to encompass additions to the signature, then we cannot hope that the language will be more expressive than the first-order transaction language defined in [32, 33].

In this paper we only discussed whether it is possible to compute preconditions or prerelations, and what are the implications of our ability to compute them. The algorithmic aspects of computing preconditions were left unexplored. In particular, we would like to address in our future work the problem of *efficiently* computing preconditions. We believe it is of practical importance to identify fragments of transaction languages such as those in [3, 4, 32] for which computing preconditions can be done efficiently (say, in polynomial time).

Computing preconditions may depend on a form in which transactions are specified. For simple relational transactions it is possible to reason about their equivalence [26]. Combining this reasoning together with the theorem proving approach of [35] is an interesting direction to pursue. That is, first we may try to find a better analyzable transaction which is equivalent to the original one, and then try to test its safety.

As mentioned in the Introduction, we are interested in transforming a verifiable transaction T into a safe transaction $\underline{\text{if}}\ \mathsf{wpc}(T, \alpha)\ \underline{\text{then}}\ T\ \underline{\text{else}}\ \textit{abort}$ which will maintain α as an invariant. As pointed out in [31, 21, 22, 28, 29], assuming that α is always true, it may be possible to find a Δ, which is much simpler than $\mathsf{wpc}(T, \alpha)$, such that $\alpha \to (\Delta \Leftrightarrow \mathsf{wpc}(T, \alpha))$. Using this we can transform T to $\underline{\text{if}}\ \Delta\ \underline{\text{then}}\ T\ \underline{\text{else}}\ \textit{abort}$ which is more efficient. We are interested in studying classes of transactions for which such a simplification can be efficiently carried out.

ACKNOWLEDGMENTS

We thank Neil Immerman and Scott Weinstein for their helpful comments during earlier stages of this work. We are grateful to the reviewers for a number of valuable suggestions; we especially thank the reviewer who found an error in an earlier version of the proof of Theorem 3. Thanks to Ron Fagin for clarifying the use of the Ajtai-Fagin games and pointing out [16], and to Moshe Vardi for bringing the results of [30] to our attention.

Received July 31, 1996; in revised form February 24, 1998

REFERENCES

1. Abiteboul, S., Hull, R., and Vianu, V. (1994), "Foundations of Databases," Addison–Wesley, Reading, MA.
2. Abiteboul, S., and Vianu, V. (1989), A transaction-based approach to relational database specification, *J. ACM* **36**, 758–789. [Extended abstract in "Proceedings of the 4th Symposium on Principles of Database Systems, 1985," pp. 193–204.]
3. Abiteboul, S., and Vianu, V. (1990), Procedural languages for database queries and updates, *J. Comput. System Sci.* **41**, 181–229.
4. Abiteboul, S., and Vianu, V. (1991), Datalog extensions for database queries and updates, *J. Comput. System Sci.* **43**, 62–124.
5. Ajtai, M., and Fagin, R. (1990), Connectivity is harder for directed than for undirected graphs, *J. Symbolic Logic* **55**, 113–150.
6. Back, R. J. R. (1981), Proving total correctness of nondeterministic programs in infinitary logic, *Acta Informatica* **15**, 233–249.
7. Benedikt, M., Dong, G., Libkin, L., and Wong, L. (1998), Relational expressive power of constraint query languages, *J. ACM* **45**, 1–34. [Extended abstract in "Proceedings of the 15th Symposium on Principles of Database Systems, 1996," pp. 5–16.]
8. Benedikt, M., Griffin, T., and Libkin, L. (1996), Verifiable properties of database transactions, *in* "Proceedings of the 15th Symposium on Principles of Database Systems," pp. 117–127.
9. Berghammer, R., Elbl, B., and Schmerl, U. (1995), Formalizing Dijkstra's predicate transformer wp in weak second-order logic, *Theoret. Comput. Sci.* **146**, 185–197.
10. Breazu-Tannen, V., and Subrahmanyam, R. (1991), Logical and computational aspects of programming with sets/bags/lists, *in* "LNCS 510: Proc. of 18th ICALP, Madrid, Spain, July 1991," pp. 60–75, Springer-Verlag, Berlin.
11. Dijkstra, E. W. (1975), Guarded commands, nondeterminacy and formal derivations of programs, *Comm. ACM* **18**, 453–457.
12. Dijkstra, E. W. (1976), "A Discipline of Programming," Prentice–Hall, Englewood Cliffs, NJ.
13. Etessami, K. (1997), Counting quantifiers, successor relations, and logarithmic space, *J. Comput. System Sci.* **54**, 400–411.
14. Fagin, R. (1993), Finite model theory—a personal perspective, *Theoret. Comput. Sci.* **116**, 3–31.
15. Fagin, R. (1997), Easier ways to win logical games, *in* "DIMACS Series in Discrete Mathematics and Theoretical Computer Science," Vol. 31, pp. 1–32, Amer. Math. Soc., Providence.
16. Fagin, R. (1997), Comparing the power of games on graphs, *Math. Logic Quart.* **43**, 431–455.
17. Fagin, R., Stockmeyer, L., and Vardi, M. (1995), On monadic NP vs. monadic co-NP, *Inform. Comput.* **120**, 78–92.
18. Gaifman, H. (1982), On local and nonlocal properties, *in* "Logic Colloquium '81" (J. Stern, Ed.), pp. 105–135, North-Holland, Amsterdam.

19. Grumbach, S., and Tollu, C. (1995), On the expressive power of counting, *Theoret. Comput. Sci.* **149**, 67–99.
20. Gurevich, Y. (1984), Toward logic tailored for computational complexity, *in* "Proceedings of Computation and Proof Theory," Springer Lecture Notes in Mathematics, Vol. 1104, pp. 175–216.
21. Henschen, L. J., McCune, W. W., and Naqvi, S. A. (1984), Compiling constraint-checking programs from first-order formulas, *in* "Advances in Database Theory" (H. Gallaire, J. Minker, and J. Nicolas, Eds.), pp. 145–170, Plenum Press, New York.
22. Hsu, A., and Imielinski, T. (1985), Integrity checking for multiple updates, *in* "Proceedings of ACM-SIGMOD 1985 International Conference on Management of Data," pp. 152–168.
23. Immerman, N. (1987), Languages that capture complexity classes, *SIAM J. Comput.* **16**, 760–778.
24. Immerman, N., and Lander, E. (1990), Describing graphs: A first-order approach to graph canonization, *in* "Complexity Theory Retrospective" (A. Selman, Ed.), pp. 59–81, Springer-Verlag, Berlin.
25. Immerman, N., Patnaik, S., and Stemple, D. (1996), The expressiveness of a family of finite set languages, *Theoret. Comput. Sci.* **155**, 111–140.
26. Karabeg, D., and Vianu, V. (1991), Simplification rules and complete axiomatization for relational update transactions, *ACM Trans. Database Systems* **16**, 439–475.
27. Libkin, L., and Wong, L. (1997), Query languages for bags and aggregate functions, *J. Comput. System Sci.* **55**, 241–272.
28. McCune, W. W., and Henschen, L. J. (1989), Maintaining state constraints in relational databases: A proof theoretic basis, *J. ACM* **36**, 46–68.
29. Nicolas, J.-M. (1982), Logic for improving integrity checking in relational data bases, *Acta Informatica* **18**, 227–253.
30. Nurmonen, J. (1996), On winning strategies with unary quantifiers, *J. Logic Computation* **6**, 755–778.
31. Qian, X. (1988), An effective method for integrity constraint simplification, *in* "Fourth International Conference on Data Engineering."
32. Qian, X. (1990), An axiom system for database transactions, *Inform. Process. Lett.* **36**, 183–189.
33. Qian, X. (1991), The expressive power of the bounded-iteration construct, *Acta Informatica* **28**, No. 7, 631–656.
34. Rosenstein, J. G. (1982), "Linear Orderings," Academic Press, New York.
35. Sheard, T., and Stemple, D. (1989), Automatic verification of database transaction safety, *ACM Trans. Database Syst.* **14**, 322–368.
36. Schwentick, T. (1996), On winning Ehrenfeucht games and monadic NP, *Ann. Pure Appl. Logic* **79**, 61–92.
37. Stemple, D., Mazumdar, S., and Sheard, T. (1987), On the modes and meaning of feedback to transaction designers, *in* "Proceedings of ACM-SIGMOD 1987 International Conference on Management of Data," pp. 374–386.