Walk logic as a framework for path query languages on graph databases

Walk Logic as a framework for path query languages on graph databases

Jelle Hellings, Bart Kuijpers, Jan Van den Bussche, and Xiaowang Zhang
Hasselt University and transnational University of Limburg

ABSTRACT

Motivated by the current interest in languages for expressing path queries to graph databases, this paper proposes to investigate Walk Logic (WL): the extension of first-order logic on finite graphs with the possibility to explicitly quantify over walks. WL can serve as a unifying framework for path query languages. To support this claim, WL is compared in expressive power with various established query languages for graphs, such as first-order logic extended with reachability; the monadic second-order logic of graphs; hybrid computation tree logic; and regular path queries. WL also serves as a framework to investigate the following natural questions: Is quantifying over walks more powerful than quantifying over paths (walks without repeating nodes) only? Is quantifying over infinite walks more powerful than quantifying over finite walks only? WL model checking is decidable, but determining the precise complexity remains an open problem.

Categories and Subject Descriptors

H.2.3 [**Database Management**]: Languages—*Query languages*; F.4.1 [**Mathematical Logic and Formal Languages**]: Mathematical Logic—*Model theory*

General Terms

Theory

1. INTRODUCTION

Graph databases have been investigated at least since the 1980s [2] and are receiving renewed attention recently [38], due to their wide variety of applications such as social sciences, bioinformatics, the Semantic Web, and GIS data. Of course, graphs are relational structures, so that we can use the basic relational query language of first-order logic (FO or relational algebra and calculus) to query graph databases as well. Many applications, however, require queries involving reachability by paths, which are not expressible in FO. Since path and reachability queries can be expressed using recursion, one may employ extensions of FO with recursion, such as Datalog or fixpoint logic, or other powerful query languages that can express reachability, such as transitive-closure logic or second-order logic. Powerful query languages have been extensively studied in database theory and finite model theory [1, 20, 27, 29, 25].

It remains interesting, however, to understand what is obtained when FO is extended with path querying primitives but not much else. Concretely we propose to study what we call *Walk Logic (WL)*: the extension of FO with explicit quantifiers over the walks in a graph.[1] The exact definition of such a logic is not entirely trivial; we have taken inspiration from the path logic on spatial data defined by Benedikt et al. [8]. There is also a connection with first-order logic on *data words* [12, 9], because a walk is seen in our logic as a data word where the letters are the labels of the nodes in the walk and the data values are the identifiers of the nodes.

WL is not intended as a user-friendly language, although a variety of queries can be expressed in a quite natural manner.

Example 1. The following formula expresses that there is a walk W from a node labeled 'office' to a node labeled 'cafeteria', and a different walk W' back from the cafeteria to the office; different in the sense that at least one node on W' is not on W.

$$\exists W \exists W' \exists t_1^W \exists t_2^W \exists t_1'^{W'} \exists t_2'^{W'} \exists t_3'^{W'} \\ (\text{office}(t_1) \wedge t_1 < t_2 \wedge \text{cafeteria}(t_2) \wedge t_1' < t_3' < t_2' \\ \wedge t_1' \sim t_2 \wedge t_2' \sim t_1 \wedge \forall t_3^W (t_1 < t_3 < t_2 \rightarrow t_3 \not\sim t_3')).$$

Here, t_1 and t_2 are variables that range over the positions of walk W, and t_1', t_2', and t_3' range over the positions of walk W'. The predicate $t_1' \sim t_2$ signifies that the node at position t_1' in W' is the same as the node at position t_2 in W. □

Rather, WL is intended as a yardstick to understand the expressiveness of path queries on graphs. Indeed, we will compare the expressiveness of WL with a number of established graph query languages, and will see that doing this

[1] A *walk* is a sequence of nodes such that consecutive nodes are linked by an edge. Walks are very often called just *paths*, but in contemporary graph terminology [19], a *path* is a walk without repeating nodes (what is often called a *simple path*). In this paper, we use the contemporary terminology.

EDBT/ICDT '13 , March 18 - 22 2013, Genoa, Italy

exercise raises various natural questions about the expressive power of such languages.

Expressing properties of walks in graphs is the bread and butter of logics used in the field of verification [14, 5]. There, the graphs are transition systems that are abstractions of computer systems, and walks are runs of these systems. A powerful such logic with explicit walk quantifiers is CTL*. A feature of logics used in verification, however, is that they are invariant under bisimulation, which means that they view a graph essentially as a tree. This makes them unsuitable for general graph querying, where we would like to ask queries such as "can I reach the cafeteria from my office and walk back by a different route?" which are typically not bisimulation-invariant. This weakness can be remedied, however, by moving to "hybrid" logics which can explicitly quantify nodes in the graph [3, 22, 37, 11].

An important caveat in the comparison of WL to verification logics is that the latter logics typically quantify over *infinite* (or at least maximal) walks, whereas from a database querying perspective it is equally natural to quantify over finite walks only. Thus we have two variants of WL: the basic one WL quantifies over finite walks, while WL$^\infty$ can quantify over infinite walks as well. We obtain that hybrid CTL* (with finite-walk semantics) is subsumed by WL$^\infty$ (WL), and the subsumptions are strict due to known limitations on the expressive power of hybrid modal logics [3]. We leave as an interesting open problem the question whether WL$^\infty$ is strictly more powerful than WL; the analogous question could also be asked for hybrid CTL*.

In the nonhybrid case, a combination of well-known techniques from model checking for LTL and the theory of counter-free automata [31, 18] can be used to show that infinite quantification does *not* yield more expressiveness. We give a proof in the framework of *linear WL*: a fragment of WL that is equivalent to boolean combinations of LTL formulas. For CTL, a similar but not quite the same result has been claimed [39]; for CTL* proper, the difference in expressiveness between the finite and the infinite semantics seems open.

In restriction to trees, WL was already studied under the name of "Path Logic" where its expressive power is again closely related to that of CTL* [26, 32]. On trees, WL is a fragment of MSO: monadic second-order logic. Hence, when moving to graphs, it is a natural question to understand how WL relates to MSO on graphs. Indeed, MSO on graphs must be one of the most deeply investigated graph query languages over the past two decades [16]. There are actually two versions of MSO on graphs: MS$_1$ can quantify only over sets of nodes, and MS$_2$ can quantify over sets of edges as well. We will show that, in contrast to the case of trees, over graphs, WL is not subsumed by MSO, not even by MS$_2$.

This shows the power of *quantifying over walks as opposed to paths;* indeed, when WL is restricted to quantification over paths only, the resulting Path Logic (PL) is clearly subsumed by MS$_2$. As a consequence, PL is strictly weaker than WL. The expressive power of WL (PL) will turn out to be incomparable to that of MS$_2$ (MS$_1$). Recently there has been some commotion about walk- versus path-based semantics in the context of regular path queries [4, 30]. Clear differences in complexity have been shown, but the consequences on the expressive power have not yet been fully investigated. We will report some initial results.

Regular path queries (which we call regular *walk* queries since their default semantics is walk-based) indeed form a graph query mechanism that has been popular at least since the 1990s [15]. It has been studied intensively [13] and is receiving renewed attention recently [6, 7]. To compare regular walk queries to WL, we naturally define RWL and ERWL: regular walk logic and extended regular walk logic, which are defined so that they have the known languages CRPQ and ECRPQ as their conjunctive-query fragments. We will show that there are queries in PTIME expressible in WL but not in ERWL. This result is interesting, in view of the reported scarcity of techniques for showing nonexpressibility even by ECRPQs [23]. Our proof is an adaptation of de Rougemonts's Ehrenfeucht-Fraïssé game argument about Hamiltonicity [17]. In the other direction there are CRPQs not expressible in WL, but the fragment of RWL that uses only star-free regular expressions is again subsumed by WL. For ERWL, even for ECRPQ, however, the restriction to star-freeness does not help to stay within WL.

Query evaluation for WL can be reduced to model checking of boolean combinations of LTL formulas, but the reduction has nonelementary complexity and this approach seems to be a gross overkill; the precise data complexity of WL query evaluation remains open. Already PL can express the Hamiltonian path query, so the data complexity is at least NP- and coNP-hard. On the one hand, we have not been able to find a PSPACE-hard query expressible in WL, but on the other hand, we do not see how WL could be subsumed by, say, second-order logic, or the while-language [1].

We will also consider the positive-existential (also known as the union-of-conjunctive-queries) fragment of WL. In that simple setting many logics amount to first-order logic extended with reachability, and the earlier question regarding finite versus infinite walks can be easily answered. Moreover, in the positive-existential setting, PL becomes more powerful than WL.

This paper is further organized as follows. Section 2 formally defines our basic setting. Section 3 discusses the WL query evaluation problem. Section 4 introduces the theme of infinite versus finite walks. Section 5 compares to MSO on graphs and introduces the theme of walks versus paths. Section 6 compares to regular walk queries. Section 7 discusses the positive-existential case. Section 8 concludes.

2. WALK LOGIC

Graphs. We assume some finite vocabulary $\mathcal{A}$ of *atomic propositions*. In this paper we will work with directed, node-labeled graphs of the form $G = (N, E, l)$ where N is a finite set of elements called the *nodes* of the graph; $E \subseteq N \times N$ is the set of *edges*; and $l : N \to 2^{\mathcal{A}}$ assigns to each node a set of atomic propositions which are abstractions of data

or properties pertinent to that node.[2] When $a \in l(v)$ we say that node v has label a. We do not consider graphs with self-loops, i.e., edges of the form (v, v), as these can be easily modeled by labels.

A *walk* in G is a finite nonempty sequence $v_1 \dots v_n$ of nodes such that $(v_i, v_{i+1}) \in E$ for each $1 \leq i < n$, i.e., any two consecutive nodes form an edge. The numbers $1, \dots, n$ are called the *positions* in the walk. An *infinite walk* is similarly defined but is countably infinite; its set of positions is simply defined to be the set of natural numbers. Clearly a (finite) graph has infinite walks if and only if it has a cycle; and a (finite) graph has a cycle if and only if it has arbitrarily long finite walks. A walk is a *path* if no node occurs more than once in it. The *length* of a walk is its length as a sequence, i.e., n in the above notation. Thus, beware that a walk of length 1 is trivial and consists just of a single node.

First-order logic. We briefly recall the basic query language of first-order logic (relational calculus, relational algebra [1]). We can view a node-labeled directed graph $G = (N, E, l)$ as a relational structure $(N, E, a^G)_{a \in \mathcal{A}}$ where the domain is N; we have E as a binary relation, and we have a unary relation a^G for each $a \in \mathcal{A}$ such that $a^G = \{v \in N \mid a \in l(v)\}$. We can then use first-order logic over the relational vocabulary $(E, a)_{a \in \mathcal{A}}$ as a graph query language. Thus, the formula $\exists z(E(x, z) \wedge E(z, y) \wedge \text{bar}(z))$ expresses that there is a node labeled 'bar' with an edge to it from x and an edge from it to y.

Walk Logic. We assume a sufficient supply of *walk variables*, and, for every walk variable W, a sufficient supply of *position variables of sort* W. Position variables of different sorts are different, and to indicate that position variable t is of sort W, we write t^W. The intuition is that if the value of W is some walk w, then the values that t can take are the positions in w.

We now define the syntax of Walk Logic (WL). An atomic formula is one of the following:

- $a(t)$ with a an atomic proposition and t a position variable;
- $t_1 \sim t_2$ with t_1 and t_2 position variables, not necessarily of the same sort;
- $t_1 < t_2$ with t_1 and t_2 position variables that must be of the same sort.

Formulas are now built from atomic formulas using the boolean connectives and existential and universal quantifiers in the usual manner. Quantifiers can be over walk variables as well as over position variables. The notion of *free variable* in a formula is defined in an unusual manner as we do not consider free walk variables. The notion of free position variables in a formula is however defined in the usual manner.

Informally, $a(t)$ means that the node at position t is labeled a; the informal semantics of $t_1 \sim t_2$ is that the same node is at position t_1 and at position t_2 (possibly in different walks); and $t_1 < t_2$ has the obvious meaning. A quantified walk variable ranges over all walks in the graph; a quantified position variable t^W ranges over all positions in the walk that is the value of the free occurrences of walk variable W.

We refer back to Example 1 for an example of a WL formula.

Formally, the semantics of WL formulas is defined as follows. Let us say that a set X of walk and position variables is *legal* if the sort of any position variable in X also belongs to X, i.e., if $t^W \in X$ then $W \in X$. Given a graph G, an *assignment on* G is a mapping α on a finite, legal set X of variables such that walk variables are mapped to walks in G, and each position variable t^W is mapped to a position in $\alpha(W)$. Given a formula φ, an assignment is called *appropriate* for φ if its domain X includes all free variables of φ.

Example 2. Let φ be the atomic formula $a(t^W)$. Then an assignment appropriate for φ must be defined on W as well as on t: on t because t is a free variable of φ, and on W because if $t^W \in X$ and X is legal, then also $W \in X$. □

Now let $G = (N, E, l)$ be a graph, φ be a formula, and α be an assignment on G appropriate for φ. Then we define that G *satisfies* φ under α, denoted by $G, \alpha \models \varphi$, as follows:

- $G, \alpha \models a(t^W)$ if $a \in l(v_i)$, where $i = \alpha(t)$ and $\alpha(W) = v_1 \dots v_n$;
- $G, \alpha \models t_1^{W_1} \sim t_2^{W_2}$ if $x_i = y_j$, where $i = \alpha(t_1)$ and $j = \alpha(t_2)$ and $\alpha(W_1) = x_1 \dots x_n$ and $\alpha(W_2) = y_1 \dots y_m$;
- $G, \alpha \models t_1 < t_2$ if $\alpha(t_1) < \alpha(t_2)$;
- $G, \alpha \models \exists W\, \varphi$, for a walk variable W, if there exists a walk w in G such that $G, \alpha\{W \mapsto w\} \models \varphi$.[3]
- $G, \alpha \models \exists t^W\, \varphi$, for a position variable t, if there exists a position i in $\alpha(W)$ such that $G, \alpha\{t \mapsto i\} \models \varphi$.

The semantics of the boolean connectives is the standard one and we omit that part of the definition.[4]

Clearly, the satisfaction of a formula φ depends only on the values of the variables in the set $\text{vars}(\varphi)$, defined as consisting of all free position variables of φ plus the sorts of these variables. Formally, we have that $G, \alpha \models \varphi$ if and only if $G, \alpha|_{\text{vars}(\varphi)} \models \varphi$. Note that $\text{vars}(\varphi)$ is the smallest (w.r.t. set inclusion) domain of any assignment appropriate for φ. If φ is a sentence (formula without free variables) then $\text{vars}(\varphi)$

[2]Often graph databases are defined as edge-labeled, rather than node-labeled, graphs [2]. Walk logic is more elegantly defined on node-labeled graphs, which is why we have chosen this option. None of our results depend essentially on this choice.

[3]The notation $\alpha\{W \mapsto w\}$ denotes the assignment that is equal to α except that W is mapped to w.

[4]Note also that equality of position variables $t_1 = t_2$ can be expressed as $\neg(t_1 < t_2) \wedge \neg(t_2 < t_1)$.

is empty; otherwise it contains at least one walk variable. As usual, if φ is a sentence, then we can simply talk about $G \models \varphi$.

Node variables. Using sentences we can express yes/no queries; using formulas with free variables we can express queries that return tuples of walks and positions in these walks. How can we express classical queries, i.e., queries that return tuples of nodes of the graph? In practice we can extend WL with *node variables*, which can (for now) only occur free and which can only be used with atomic propositions and in comparisons by $\sim$ with other node variables or position variables.

Example 3. The following query, using two free node variables x and y, defines all pairs of nodes (x, y) such that there is a walk from x to y that goes through a node labeled a:

$$\exists W \exists t_1^W \exists t_2^W \exists t_3^W (t_1 < t_2 < t_3 \wedge t_1 \sim x \wedge t_3 \sim y \wedge a(t_2)).$$

□

In principle, however, node variables can be simulated using position variables. Formally, for every node variable x we use a separate walk variable W_x and then use x as a position variable of sort W_x. Then for any formula φ using node variables, we have $G, \alpha \models \varphi$ (with α assigning nodes of G to the node variables) if and only if $G, \alpha' \models \varphi$ where α' is any assignment obtained from α by assigning to W_x a walk w that contains node $\alpha(x)$, and assigning to x any position in w where $\alpha(x)$ occurs.

We can even simulate quantification of node variables. Let φ be a formula involving quantified node variables. Define φ' to be the formula obtained from φ by replacing every quantifier $\exists x$ of a node variable by $\exists W_x \exists x$. Then again one can verify that $G, \alpha \models \varphi$ if and only if $G, \alpha' \models \varphi'$. Moreover, we could even allow edge predicates $E(x, y)$ between node variables. This can be simulated by the subformula

$$\exists W \exists t_1^W \exists t_2^W (t_1 \sim x \wedge t_2 \sim y \wedge t_2 = t_1 + 1).$$

In the above formula the successor predicate $t_2 = t_1 + 1$ is of course readily expressed as $t_1 < t_2 \wedge \neg \exists t_3^W\ t_1 < t_3 < t_2$.

By the above discussion, the following is clear:

PROPOSITION 1. *WL is at least as powerful as first-order logic (FO) on graphs.*

Of course WL is much more powerful than FO, as the following exercise shows.

Exercise 1. Express in WL that the graph has a Hamiltonian path. □

Later in the paper we will explore the expressive power of WL further.

Variants of WL. As motivated in the Introduction, we define three natural variants of WL as follows. We omit their formal definition.

- PL is the variant of WL where walk variables are restricted to range over paths only. So syntactically, PL is identical to WL, but the semantics is different.
- WL^∞ is the extension of WL where, in addition to quantifiers $\exists W$ over finite walks, we also allow quantifiers $\exists^\infty W$ over infinite walks.
- Linear WL is the restriction of WL where the predicate $\sim$ is forbidden. Similarly one can consider linear WL^∞ as a restriction of WL^∞.

Example 4. The following PL formula expresses that there is no simple way to go from node x to node y and visit a bar on the way:

$$\forall P((\exists s^P \exists t^P (s < t \wedge s \sim x \wedge t \sim y)) \rightarrow \forall u^P (s < u < t \rightarrow \neg \mathrm{bar}(u))).$$

The semantics of the above formula as a WL formula would say something stronger, namely, that there is no bar reachable from x so that y is reachable from the bar. □

Since one can express in WL that a walk W is a path (see solution of Exercise 1), PL is subsumed by WL. Trivially, linear WL is subsumed by WL, and WL is subsumed by WL^∞. We will investigate strictness of these inclusions later in the paper.

3. QUERY EVALUATION FOR WL

For any WL sentence φ, the *query evaluation problem for* φ is to decide, given a graph G, whether $G \models \varphi$. (Query evaluation of general formulas, rather than just sentences, is discussed afterwards.) This is the "data complexity" setting [36]; in the "combined complexity" setting, one considers the general problem of deciding, given G and φ, whether $G \models \varphi$.

We will first show that if φ is *linear* (does not use the $\sim$ predicate), then query evaluation for φ is decidable in polynomial time (data complexity).

We begin by noting the following which can be proved by straightforward formula manipulation:

LEMMA 1. *Every linear WL sentence φ is equivalent to a boolean combination of linear WL sentences of the form $\exists W\ \psi$ where ψ has no walk quantifiers and all position variables occurring in ψ have sort W.*

In the absence of the predicate $\sim$, walks are viewed as strings over the alphabet $\Sigma = 2^{\mathcal{A}}$, so that sentences ψ as in the above proposition are nothing but sentences in *first-order logic (FO) on strings* over the alphabet Σ [35]. On strings, first-order logic is equivalent to the logic LTL [18]. We are left with the problem to decide, given a graph G, whether there exists a walk w in G such that the string of labels

traced by w satisfies a fixed LTL formula. But this is almost exactly the model checking problem for LTL [5], which is well known to be decidable in polynomial time.[5] We conclude:

PROPOSITION 2. *For each linear WL sentence φ, the query evaluation problem for φ is decidable in polynomial time.*

Note that the *combined* complexity of this method is horrible, since it involves a translation from FO to LTL, for which a non-elementary lower bound is known [34].

Our next step is to observe that query evaluation for WL can be reduced to query evaluation for linear WL, although the reduction is not efficient. The reduction is straightforward. Given a graph $G = (N, E, l)$ labeled using the basic set of atomic propositions $\mathcal{A}$, we construct the *exposed* graph $\breve{G} = (N, E, \breve{l})$ labeled using the extended set of atomic propositions $\mathcal{A} \cup N$ by defining $\breve{l}(v) = l(v) \cup \{v\}$. Thus, each node is labeled additionally with its own identifier. Then given a formula φ, we define the formula $\breve{\varphi}_G$ obtained from φ by replacing each comparison $t_1 \sim t_2$ by $\bigvee_{v \in N}(v(t_1) \wedge v(t_2))$. Note that this is a linear formula. Clearly, we have for all graphs G and all assignments α on G appropriate for φ that $G, \alpha \models \varphi$ if and only if $\breve{G}, \alpha \models \breve{\varphi}_G$. We conclude:

THEOREM 1. *The query evaluation problem for WL sentences is decidable.*

Now the data complexity of the decision procedure described above is horrible. Indeed, the formula $\breve{\varphi}_G$ depends on G, so the *combined* complexity of checking $\breve{G} \models \breve{\varphi}_G$ determines the *data* complexity of the overall procedure. As mentioned in the Introduction, we leave as open:

PROBLEM 1. *Determine the precise data complexity of the query evaluation problem for WL sentences.*

In view of Exercise 1, the data complexity is at least NP-hard, and also coNP-hard since WL is closed under negation.

Formulas with free variables. For a WL formula φ that may have free variables, the query evaluation problem is to decide for a given G and assignment α on G defined on $\mathrm{vars}(\varphi)$, whether $G, \alpha \models \varphi$. The reduction to linear WL works for formulas in general, so let us assume that φ is linear. Lemma 1 can be generalized to formulas so that every linear WL formula is equivalent to a boolean combination of formulas of the form $\exists W\, \psi$ as in Proposition 1, and additionally formulas of the form χ where χ has no walk quantifiers and all position variables are of the same sort. Now each such formula χ, talking about some walk variable $W \in \mathrm{vars}(\varphi)$, can simply be evaluated on the given walk $\alpha(W)$.

The expressive power of linear WL. We note that Lemma 1 actually provides a characterization of the expressive power of linear WL. As a corollary we obtain

PROPOSITION 3. *Linear WL is strictly subsumed by WL.*

PROOF. Consider the four-node, diamond-shaped directed graph $D = (\{1,2,3,4\}, \{(1,2),(1,3),(2,4),(3,4)\})$. The query "the graph has a subgraph isomorphic to D" is readily expressed in WL, but not in linear WL. Indeed, no sentence $\exists W\, \psi$ as in Lemma 1 can distinguish D from the tree-shaped graph $T = (\{1,2,3,4,5\}, \{(1,2),(1,3),(2,4),(3,5)\})$. □

More generally, Lemma 1 implies that linear WL is bisimulation-invariant [24] whereas full WL is clearly not.

4. INFINITE WALKS

Lemma 1 applies to linear WL^∞ as well. Hence, as already hinted upon in Footnote 5, the query evaluation algorithm from the previous section also applies to WL^∞. In fact, the default semantics for LTL on graphs is to consider infinite walks. We are thus led to wonder whether the expressive power of WL is actually strictly weaker than that of WL^∞. We can only show the following. The proof below uses standard arguments, except that it needs the notion of counter-free automaton to apply to infinite strings, which was only recently worked out [18]:

THEOREM 2. *Every linear WL^∞ sentence is equivalent to some WL sentence.*

PROOF. We must show that a sentence of the form $\exists^\infty W\, \psi$, with ψ an FO sentence on strings, is expressible in finite walk logic. We can translate ψ into a Büchi automaton M that is *counter-free* [18]. It is also known that every language of finite words defined by a counter-free finite automaton is also definable in first-order logic. We note that the notion of counter-freeness has nothing to do with the acceptance criterion of the automaton; in particular, when a counter-free Büchi automaton is used alternatively as a finite automaton, it is still counter-free considered as an automaton on finite words. This will be crucial for the correctness of the proof.

We want to express that there exists an infinite walk in $G = (N, E, l)$ whose label sequence is accepted by M. Thereto we use a well-known product construction; we define the Büchi automaton $G \times M$ as follows.

- The set of states equals $N \times Q$ with Q the set of states of M;
- The initial states are those of the form (x, q) with q an initial state of M;

[5] *Almost* exactly because of two differences. First, LTL model checking only considers walks starting in a designated initial node. But that is not essential to the model-checking algorithm. Second, LTL model checking normally considers "maximal" walks only: these are infinite walks, and finite walks that end in a terminal node (without outgoing edges). The well-known automata-based LTL model checking algorithm, however, works as well for finite walks (using standard finite automata) as for infinite walks (using Büchi automata) [5].

- The repeating states are those of the form (x, r) with r a repeating state of M;
- The transitions are those of the form $(x, q) \xrightarrow{a} (y, q')$ such that
  1. there is an edge $(x, y) \in E$;
  2. $a = l(x)$ in G;
  3. there is a transition $q \xrightarrow{a} q'$ in M.

Clearly there is an infinite walk in G whose label sequences is accepted by M, if and only if the automaton $G \times M$ accepts some infinite word.

It is well known [5] that a Büchi automaton accepts some infinite word, if and only if some initial state can reach some repeating state that lies on a cycle. Applied to $G \times M$, our task is thus focused on the following two properties that may or may not hold for given states (x, q) and (y, r) of $G \times M$:

1. (y, r) is reachable by a (possibly empty) path of transitions starting from (x, q);
2. (y, r) lies on a cycle of transitions.

We are going to express properties 1 and 2 by walk formulas $\psi_1^{q,r}(x, y)$ and $\psi_2^r(y)$ that work uniformly for all graphs G. Then the final sentence will be

$$\bigvee_q \bigvee_r \exists x \exists y (\psi_1^{q,r} \wedge \psi_2^r) \qquad (*)$$

where the disjunction is over all initial states q of M and all repeating states r of M. Since we have seen that quantification over nodes is expressible in WL, the theorem will be proved.

The formulas will follow from the following two claims. Let $M_{q,r}$ be the finite automaton obtained from M by setting q to be the only initial state and setting r to be the only final state. Let $M_{q,q}$ be defined analogously. We claim:

1. In $G \times M$, state (y, r) is reachable from state (x, q), if and only if there is a walk in G from x to y whose label sequence, with the last letter omitted, is accepted by $M_{q,r}$.
2. In $G \times M$, state (y, r) lies on a cycle of transitions, if and only if there is a walk in G, of length strictly greater than one, whose label sequence, with the last letter omitted, is accepted by $M_{q,q}$.

   Recall that we define a walk in our work as a nonempty sequence of nodes such that there is an edge from any node in the sequence to its successor in the sequence; thus a walk of length one is a trivial walk that does not follow any edge. Such walks are allowed in the previous property but not in the present one.

The equivalent formulations of the two properties given by the above two claims can readily be translated into walk logic, once we realize that M is counter-free, so that the finite string languages defined by $M_{q,r}$ and $M_{r,r}$ are first-order definable. □

Note that we have not shown that every linear WL^∞ sentence is equivalent to some linear WL sentence; the walk logic formula $(*)$ in the above proof is not linear, since we need quantification over nodes x and y which needs to be simulated using $\sim$. It is tempting to conjecture that linear WL^∞ is strictly more expressive than linear WL. Also, the nonlinear case remains open:

Problem 2. *Is WL^∞ strictly more expressive than WL for yes/no queries over graphs?*

Hybrid CTL^*. Our original reason to consider WL^∞ is that logics used in verification have a semantics based on infinite walks. As mentioned in the Introduction, a powerful such logic is hybrid CTL^*, denoted here by HCTL^* [37, 11]. Actually we can define two variants: HCTL^* under the default semantics based on infinite walks, and a finite-walk variant which we denote by HCTL^*_{fin}. A HCTL^* sentence is always evaluated on a graph and an initial node, thus, these sentences always express unary queries.[6] The expressive power compares to the WL variants as follows:[7]

Proposition 4. *In their expressive power of unary queries on graphs, HCTL^* is strictly weaker than WL^∞, and HCTL^*_{fin} is strictly weaker than WL.*

Proof. That the HCTL^* variants are subsumed by the corresponding WL variants is proven by a syntactic translation. That the subsumptions are strict follows from a known limitation on the expressive power of hybrid modal logics, namely, that they are invariant under generated submodels [3]. For instance, the query "there exists a node labeled a" is expressible in WL but neither in HCTL^*_{fin} nor in HCTL^*, since the query is not invariant under generated submodels. □

Remark 1. Areces et al. [3] have shown that the hybrid version of standard modal logic actually captures the fragment of FO invariant under generated submodels. In analogy to the result that, in restriction to trees, CTL^* captures the bisimulation-invariant fragment of Path Logic [32], it would be interesting to know whether HCTL^* captures the fragment of WL invariant under generated submodels. Also, the proof given by Areces et al. does not immediately apply in restriction to finite graphs [33].

Remark 2. Open problem 2 can also be stated for HCTL^*: How do the expressive powers of HCTL^* and HCTL^*_{fin} compare?

[6] With additional free "state variables", HCTL^* formulas can more generally express k-ary queries.

[7] We note that Benevides et al. [11] report results on the expressive power of HCTL^* that are slightly misleading, for example, they claim that Hamiltonian path and Eulerian trail are expressible, but they do not exhibit one formula that expresses the query on all inputs; instead, they need larger and larger formulas for larger and larger input graphs.

Remark 3. One may also wonder about the relationship between linear WL^∞ and standard, non-hybrid CTL^*. From Lemma 1 it follows that every linear WL^∞ sentence expresses a boolean combination of LTL properties about the graph. It is well-known known that LTL is subsumed by CTL^*. We have not investigated the reverse relationship, whether every CTL^* global graph property (property true at all nodes) is also expressible in linear WL^∞.

5. MSO, AND PATHS VERSUS WALKS

In this section we compare the expressive power of WL and its path-based fragment PL to that of monadic second-order logic (MSO) on graphs.

MSO [16] is the extension of first-order logic with set variables. On graphs, MSO comes in two flavors. In MS_1, we simply use MSO on the same relational structure representation of a graph as used in first-order logic on graphs (recall Section 2). In MS_2, we view a graph $G = (N, E, l)$ as a relational structure $(N \cup E, \mathrm{source}^G, \mathrm{target}^G, a^G)_{a\in\mathcal{A}}$ where source^G equals the binary relation $\{((v, v'), v) \mid (v, v') \in E\}$ and target^G equals $\{((v, v'), v') \mid (v, v') \in E\}$. The edge predicate is redundant in this setting. So MS_2 is MSO over the relational vocabulary $(\mathrm{source}, \mathrm{target}, a)_{a\in\mathcal{A}}$. In MS_2, edges are treated as first-class objects; in particular, one can quantify over sets of nodes and edges as opposed to MS_1 where one can quantify only over sets of nodes. It is known that MS_1 is strictly weaker than MS_2: the query "the graph has a Hamiltonian path" is expressible in MS_2 but not in MS_1.

We begin by making two immediate observations:

PROPOSITION 5. *PL is subsumed by MS_2, but not by MS_1.*

PROOF. PL is subsumed by MS_2 since in the latter logic one can express that a set of edges is a path. PL is not subsumed by MS_1 since Hamiltonicity is expressible in PL but known not to be expressible in MS_1 [16]. □

It will turn out that the subsumption of PL by MS_2 is the only comparison that holds between PL and WL on the one hand, and MS_1 and MS_2 on the other hand. We begin by demonstrating a number of global graph properties that are not expressible in WL^∞.

THEOREM 3. *The following yes/no queries are not expressible in WL^∞: weak connectivity; planarity; and bipartiteness.*[8]

PROOF. The proof is based on the observation that WL^∞ can only work with directed walks. Hence, on classes of graphs with a constant bound on the length of directed walks, WL^∞ collapses to FO. Hence it suffices to give, for each property, a class C of directed graphs of bounded walk length and show that the property is not expressible in FO even in restriction to the class C. □

[8]These are properties of undirected graphs whereas we are working with directed graphs. We resolve this mismatch consistently in this paper by agreeing that a directed graph G satisfies a property of undirected graphs if the undirected graph underlying G satisfies the property.

Since weak connectivity (even on directed graphs), as well as bipartiteness, is expressible in MS_1, we obtain:

COROLLARY 1. *MS_1 is not subsumed by WL^∞.*

Remark 4. In contrast to weak connectivity, *strong (directed) connectivity* is obviously expressible in WL. Thus, if we restrict attention to undirected graphs, which can be modeled in our framework as directed graphs with a symmetric edge relation, connectivity becomes expressible in WL, even in PL. Furthermore, also *planarity* restricted to undirected graphs is expressible in PL, using the well-known Kuratowski characterization of planar graphs. We leave open whether *bipartiteness* is expressible in WL on undirected graphs. Indeed we leave open whether Corollary 1 still holds in restriction to undirected graphs, although we conjecture this to be the case. □

The next theorem will imply the converse to Corollary 1 for WL and MS_2. Recall that an *Eulerian trail* in a directed graph $G = (N, E)$ is a walk $v_1 \dots v_n$ containing each edge precisely once, so formally, $E = \{(v_i, v_{i+1}) \mid 1 \le i < n\}$ and $(v_i, v_{i+1}) = (v_j, v_{j+1})$ implies $i = j$.

THEOREM 4. *The existence of an Eulerian trail is not expressible in MS_2.*

PROOF. For natural numbers i and j, define the graph $G(i, j) = (N, E)$ by $N = \{x_1, \dots, x_i, v_1, v_2, y_1, \dots, y_j\}$ and $E = \{(x_k, v_1) \mid 1 \le k \le i\} \cup \{(v_1, y_k) \mid 1 \le k \le j\} \cup \{(y_k, v_2) \mid 1 \le k \le j\} \cup \{(v_2, x_k) \mid 1 \le k \le i\}$. Then $G(i, j)$ has an Eulerian trail if and only if $|i - j| \le 1$.

We will show that there is no MS_2 sentence that expresses existence of Eulerian trail even when restricted to the class of all $G(i, j)$ graphs. Since this class is "uniformly 2-sparse" [16], MS_2 is equivalent to MS_1 on this class, so it suffices to show that there is no MS_1 sentence.

We now claim the following reduction from MS_1 on graphs to MSO on naked sets:[9] *For each i and each MSO sentence φ over graphs, there exists an MSO sentence $\varphi'_{(i)}$ over naked sets such that for each j, we have $G(i, j) \models \varphi \Leftrightarrow [j] \models \varphi'_{(i)}$, where $[j]$ denotes the set $\{1, \dots, j\}$. Moreover, $\varphi'_{(i)}$ has the same quantifier rank as φ.*

Now assume, for the sake of contradiction, that there exists an MSO sentence φ expressing existence of Eulererian trail. Let k be the quantifier rank of φ. Then for each j, we have $G(j, j+1) \models \varphi$ but $G(j, j+2) \not\models \varphi$. As a consequence of the reduction to naked sets, we have for each j that $[j + 1] \models \varphi'_{(j)}$ but $[j + 2] \not\models \varphi'_{(j)}$. However, it is known [29, Proof of Proposition 7.12] that for any j' and j'' at least 2^k, the

[9]On naked sets, the vocabulary is empty, and the only relation symbol that can be used in formulas is the equality predicate.

sets $[j']$ and $[j'']$ are indistinguishable by MSO sentences of quantifier rank k. Since $\varphi'_{(j)}$ has quantifier rank k, we have arrived at the desired contradiction. □

Remark 5. The above result is for directed Eulerian trails on directed graphs, but the analogous inexpressibility result holds for the classical Eulerian property of *undirected* graphs (say, the undirected graph underlying the given directed graph). Indeed in the above proof we can make the graphs $G(i,j)$ symmetrical; the resulting class of graphs is uniformly 4-sparse and the same argument applies. □

Since the existence of an Eulerian trail is readily expressed in WL, we obtain from Theorem 4:

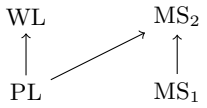
COROLLARY 2. *WL is not subsumed by MS_2.*

Moreover, since PL *is* subsumed by MS_2, we obtain:

COROLLARY 3. *WL is strictly stronger in expressive power than PL.*

Remark 6. While Eulerian trail is easily expressed in WL, one can show that the query "the underlying undirected graph is Eulerian" is *not* expressible, not even in WL^∞. This can be shown using the same counterexample graphs used in the proof of Theorem 3, again exploiting the weakness WL is limited to following directed walks. Of course, when the given graph is symmetric, the Eulerian property is expressible. □

To conclude this section, the picture that emerges is the following, where arrows indicate strict subsumption and absence of arrows indicates incomparable expressive power:



This picture seems satisfying, since PL and WL are really meant for walk-based querying whereas the MSO variants are meant for global graph properties as well.

6. REGULAR WALK LOGIC

In this section we compare the expressive power of WL with the much-studied regular path queries. One of the more powerful formalisms in the literature is that of ECRPQs [6]. However, ECRPQs are a positive-existential logic, so we first introduce *extended regular walk logic (ERWL)* as the natural extension of ECRPQs closed under negation. In the literature this variant is also called $ECRPQ^\neg$.

Recall that in a graph each node v is labeled with a set $l(v) \subseteq \mathcal{A}$ of atomic propositions. For regular walk logic it is more convenient to think of each such set as a global label and thus work with the alphabet $\Sigma = 2^{\mathcal{A}}$ of labels.

For the syntax of ERWL we assume sufficient supplies of *node variables* and *walk variables*. The atomic formulas are the following:

- $a(x)$ and $x = y$ with $a \in \Sigma$ and x and y node variables;
- $\text{from}(W, x)$ and $\text{to}(W, x)$, with W a walk variable and x a node variable;
- $R(W_1, \ldots, W_k)$, with $W_1, \ldots, W_k$ walk variables and R a regular expression over the alphabet $\Sigma_\perp^k$, where $\Sigma_\perp = \Sigma \cup \{\perp\}$ and $\Sigma = 2^{\mathcal{A}}$.

The formulas of ERWL are built from the atomic formulas using quantification over nodes, quantification over walks, and the boolean connectives in the usual manner. Examples of formulas will be seen in the proofs below.

An ERWL formula φ is evaluated on a graph $G = (N, E, l)$ and an assignment α defined on the free variables of φ, as follows: (we only give the nonobvious rules)

- $G, \alpha \models \text{from}(W, x)$ if $\alpha(x)$ is the first node of walk $\alpha(W)$.
- $G, \alpha \models \text{to}(W, c)$ if $\alpha(x)$ is the last node of walk $\alpha(W)$.
- $G, \alpha \models R(W_1, \ldots, W_k)$ if the tuple of traces $(l(\alpha(W_1)), \ldots, l(\alpha(W_k)))$ satisfies the regular relation R.[10] Here, for a walk $w = v_1 \ldots v_n$ we naturally define the string $l(w) = l(v_1) \ldots l(v_n)$ over Σ; this string is called the *trace* of w.
- $G, \alpha \models \exists W\, \varphi$, for walk variable W, if there exists a walk w in G such that $G, \alpha\{W \mapsto w\} \models \varphi$.

Note that the atomic predicates $a(x)$ in principle are redundant as they can be expressed as $\exists W(\text{from}(W, x) \wedge a(W))$.

We identify the following fragments of ERWL:

- RWL is the fragment where atomic formulas of the form $R(W_1, \ldots, W_k)$ are only allowed for $k = 1$.
- ECRPQ is the fragment where only existential quantifiers are allowed and the only allowed boolean connective is $\wedge$.
- CRPQ is the syntactic intersection of RWL and ECRPQ.

[10] Given a tuple $\vec{s} = (s_1, \ldots, s_k)$ of strings over Σ, we can form a string $[\vec{s}]$ over $\Sigma_\perp^k$ as follows. Fist, if necessary, we pad strings to the right with $\perp$ symbols so that the all strings become of the same length. Then, the tuple of first letters is taken as the first letter of $[\vec{s}]$; the tuple of second letters as the second letter of $[\vec{s}]$; and so on. Now we say that $\vec{s}$ satisfies the regular relation R if $[\vec{s}]$ satisfies R [6, 10].

- For each variant we can also consider the star-free fragment, where only star-free regular expressions are allowed.[11]

Since star-free regular expressions are first-order definable [35], we immediately obtain:

PROPOSITION 6. *Star-free RWL is subsumed by WL.*

When either the star-free restriction is lifted, or regular relations are allowed, however, we are no longer within WL:

PROPOSITION 7. *Neither CRPQ nor star-free ECRPQ is subsumed by WL.*

PROOF. We use here the popular conjunctive-query syntax for (E)CRPQs. For a CRPQ not expressible in WL, we use the even-length walk query

$$Q_1 \leftarrow a(x), b(y), (cc)^*(x, y).$$

Strictly in our syntax the above query would be written $\exists W \exists x \exists y (a(x) \wedge b(y) \wedge \mathrm{from}(W, x) \wedge \mathrm{to}(W, y) \wedge (cc)^*(W))$. For a star-free ECRPQ not expressible in WL, we use the existence of two walks of different length:

$$Q_2 \leftarrow \mathrm{from}(W_1, x_1), \mathrm{to}(W_1, y_1), \mathrm{from}(W_2, x_2), \mathrm{to}(W_2, y_2), \\ a(x_1), b(y_1), a(x_2), b(y_2), \mathrm{dl}(W_1, W_2).$$

Here, the different-length predicate dl can be expressed by the star-free regular expression (writing Λ for $\Sigma^2_\perp$)

$$\bigcup_{a \in \Sigma} (\Lambda^* \left[\begin{smallmatrix} \perp \\ a \end{smallmatrix}\right] \Lambda^* \cup \Lambda^* \left[\begin{smallmatrix} a \\ \perp \end{smallmatrix}\right] \Lambda^*).$$

To prove that these two queries are not expressible in WL, we use the following:

LEMMA 2. *Over graphs that are disjoint unions of chains, WL collapses to FO(Reach).*

Here, FO(Reach) is the extension of FO on graphs that provides the reflexive-transitive closure of the edge relation in the form of an extra predicate $\mathrm{Reach}(x, y)$.

By the Lemma, it suffices to show that Q_1 and Q_2 are not expressible in FO(Reach) on disjoint unions of chains, which can be shown using well-known inexpressibility arguments about FO on linear orders [29].

To prove the Lemma, we use that on disjoint unions of chains there is at most one walk between any two nodes. Thus we can simulate a walk variable W by a pair of node variables x_W and y_W satisfying $\mathrm{Reach}(x_W, y_W)$. A position variable t^W is then simulated by a node variable t satisfying $\mathrm{Reach}(x_W, t) \wedge \mathrm{Reach}(t, y_W)$. A comparison $t_1 \sim t_2$ is expressed as $t_1 = t_2$ and a comparison $t_1 < t_2$ is expressed as $t_1 \neq t_2 \wedge \mathrm{Reach}(t_1, t_2)$. □

[11] The star-free regular expressions (sfre) over an alphabet Σ are defined as follows. Each $a \in \Sigma$ is a sfre; the expression Σ^* is a sfre; and if e_1 and e_2 are sfres, then so are $e_1 \cdot e_2$, $e_1 \cup e_2$, and $e_1 - e_2$.

Remark 7. The queries given in the above proof use node labels, but we can also separate the logics using queries over unlabeled graphs. To separate CRPQ from WL one can use "there exist two nodes x and y such that there is both an even-length and an odd-length walk from x to y". To separate star-free ECRPQ from WL one can use "there exist two nodes x and y such that there are two walks from x to y of different lengths". The proof can be adapted so that graphs consisting of two disjoint chains joined at their endpoints are considered. By a slightly more involved simulation we still have that WL collapses to FO(Reach) on such graphs. □

As a counterweight to the previous proposition, we have inexpressibility on the ERWL side:

THEOREM 5. *The yes/no queries "there exists a Hamiltonian path" and "there exists an Eulerian trail" are not expressible in ERWL.*

PROOF. We adapt de Rougemont's Ehrenfeucht-Fraïssé (EF) game argument that Hamiltonicity is not expressible in infinitary logic [17, 28]. Actually, we only need the winning strategy for the first-order logic game; it will however be extended so that the spoiler and the duplicator can choose walks as well as nodes.

The considered family of graphs is that of all graphs $\overline{\mathbf{K}}_m \times \mathbf{C}_n$ that are the product graph of a totally disconnected m-node graph $\overline{\mathbf{K}}_m$ with the n-node undirected cycle $\mathbf{C}_n$, with $1 < m$ and $1 < n$. We thus have nodes $v_0, \ldots, v_{m-1}$ from $\overline{\mathbf{K}}_m$, nodes $w_0, \ldots, w_{n-1}$ from $\mathbf{C}_n$, undirected edges (v_i, w_j) for $0 \leq i < m$ and $0 \leq j < n$, and undirected edges $(w_j, w_{j+1 \mod m})$ for $0 \leq j < m$. The edges are symmetric, so the pairs just described as well as their converses belong to the graph.

It can be verified that $\overline{\mathbf{K}}_m \times \mathbf{C}_n$ is Hamiltonian if and only if $m \leq n + 1$. However, the duplicator has a winning strategy in the n-round EF game on $\overline{\mathbf{K}}_m \times \mathbf{C}_n$ and $\overline{\mathbf{K}}_{m'} \times \mathbf{C}_n$ for any $m, m' \geq n$, showing that Hamiltonicity is not expressible in FO on graphs.

We now observe that the winning strategy can be expanded to the case where we extend the universe of each structure with the set of all walks on that graph, and expand the vocabulary with the predicates from, to, and R for all regular relations R, in accordance with the semantics of ERWL. Clearly, ERWL boils down to FO on these extended and expanded structures.

The reason why the strategy extends to the walk-extended structures is the following. Suppose the spoiler chooses a walk w from node v_1 to node v_2 in one of the structures. Let v_1' and v_2' be the nodes in the other structure that the duplicator would choose in response to the point moves v_1 and v_2. Now there always exists a walk w' from v_1' to v_2' in the other structure of exactly the same length as w, and this is the walk move that the duplicator will respond with. This follows from the following claim: *Let v_1 and v_2 be two nodes in $\overline{\mathbf{K}}_n \times \mathbf{C}_m$. Then for every length $\ell \geq 3$ there exists a walk from v_1 to v_2 of length ℓ.* We only have to worry

about lengths, since our graphs are unlabeled ($l(v) = \emptyset$ for each node v), so the alphabet Σ has only a single letter. Hence walks of the same length have identical traces. Note also that walks of length one and two (equality and edge) are already taken care of by the winning condition on point moves.

The argument for Eulerian trail is very similar, but now we use the undirected versions of the $G(i, j)$ graphs from the proof of Theorem 4. □

Since Hamiltonicity is expressible in PL and Eulerian trail is expressible in WL, we obtain from Theorem 5:

COROLLARY 4. *Neither PL nor WL is subsumed by ERWL.*

We note though the following:

PROPOSITION 8. *Linear WL is subsumed by RWL.*

Indeed this follows immediately from Lemma 1 and the expressibility of first-order logic on strings by star-free regular expressions.

Paths versus walks in regular walk logics. Just like we have considered the variant PL of WL that has a path-based rather than a walk-based semantics, we can consider path semantics for ERWL and its fragments. We denote the resulting variant of ERWL by ERPL. For (E)CRPQs, however, which are established names and already have a P in them, we will denote the path-based variants by pCRPQ and pECRPQ.

The apparent increase in complexity of path-based versus walk-based regular walk queries has been reported recently [4, 30]. We complement these results with some initial expressivity results. Much remains open, however.

Recall that PL is subsumed by WL simply because one can express in WL that a walk is a path. This does not go through for ERWL:

THEOREM 6. *ERPL is not subsumed by ERWL.*

PROOF. We work over a one-letter alphabet $\Sigma = \{a\}$ (equivalently, $\mathcal{A} = \emptyset$). The query "the longest path has even length"[12] is expressible in ERPL by $\exists P(\mathit{even}(P) \wedge \neg\exists Q\, \mathit{longer}(Q, P))$, where *even* is the regular expression $(aa)^*$ and *longer* is the regular expression $\left[\begin{smallmatrix} a \\ a \end{smallmatrix}\right]^* \left[\begin{smallmatrix} a \\ \bot \end{smallmatrix}\right]^+$. This query is not expressible in ERWL because it returns true on $\overline{\mathbf{K}}_n \times \mathbf{C}_n$ but false on $\overline{\mathbf{K}}_{n+1} \times \mathbf{C}_n$, using the graphs indistinguishable in ERWL we have already seen in the proof of Theorem 5. □

[12]This query is NP-complete [21], while the data complexity of ERWL is in P [6], but in absence of a proof of P ≠ NP we still have to prove inexpressibility in ERWL.

We leave open whether ERWL is subsumed by ERPL. We can show though that Eulerian trail is still not expressible in ERPL. We omit the proof which is a modification of the proof of Theorem 5.

We also have an analogue of Theorem 6 on the CRPQ level:

THEOREM 7. *pCRPQ is not subsumed by ECRPQ.*

PROOF. Consider the pCRPQ

$$Q \leftarrow (aa)^*(x, y),\ (aa)(y, x)$$

which expresses that there is an even cycle in the graph. To see that Q is not expressible in ECRPQ we consider for any n the cycle C_n of size n. There is a homomorphism h from C_{2n} to C_n which goes once over C_{2n} while going twice over C_n. Moreover, this homomorphism extends to a homomorphism $\bar{h}$ of the structures extended and expanded with all walks, as in the proof of Theorem 5. Specifically, we can always map a walk w from v_1 to v_2 in C_{2n} to a walk $\bar{h}(w)$ from $h(v_1)$ to $h(v_2)$ of exactly the same length, by letting $\bar{h}(w)$ go around in C_n twice as often as w goes around in C_{2n}. As in the proof of Theorem 5, since we are working with a one-letter alphabet, identical lengths guarantee that all regular relations are preserved, so $\bar{h}$ is indeed a homomorphism. Since ECRPQ is subsumed by positive-existential first-order logic on these expanded and extended structures, and positive-existential first-order logic is preserved by homomorphisms, we have that when Q is expressible in ECRPQ and Q is true on C_{2n} then also Q is true on C_n. For odd n this is a contradiction. □

We leave open whether the even cycle query is expressible in the more powerful logic ERWL. Note though that in ERWL one can express that a walk W is a shortest path from node x to node y. In particular, on cycle graphs as used in the above proof, ERWL can express path quantifiers. So at least another proof would be needed.

Infinite versus finite walks for regular walk logics. As with WL^∞, we can add $\exists^\infty$ quantifiers to ERWL and its variants and use appropriate notions of regular relations for infinite words. (Of course, the 'to' predicate will never apply to an infinite walk variable, but the 'from' can.) Then again the question of relative expressiveness can be asked. We leave this question largely untouched; we only note that the proof idea of Theorem 2 readily applies to show the following analogue:

PROPOSITION 9. *CRPQ^∞ is no more powerful than CRPQ.*

7. POSITIVE-EXISTENTIAL WL

Given the popularity of unions of conjunctive queries (equivalent to positive-existential logic [1]), it appears useful to consider the positive-existential fragment of WL, where only existential quantification is allowed and the only boolean connectives are $\wedge$ and $\vee$. A small caveat is that we must add the edge predicate E, or equivalently, the successor predicate $t_2 = t_1 + 1$ on positions, because to express these in core WL we need negation.

Our result is that in this case, there is no difference between infinite versus finite, and the resulting expressive power is a very natural one. Recall that FO(Reach) is the extension of first-order logic on graphs with the reflexive-transitive closure of the edge predicate in a new predicate Reach. In general, FO(Reach) is strictly subsumed by WL and even by PL. We now see that the positive-existential fragments of WL and FO(Reach) coincide, but that paths are now strictly stronger than walks:

THEOREM 8. *The positive-existential fragments of WL and WL^∞ are equivalent and equivalent to the positive-existential fragment of FO(Reach). The positive-existential fragment of PL is strictly more powerful.*

PROOF. In this proof we mostly omit the adjective 'positive-existential' which applies everywhere in the proof. FO(Reach) can be translated in the other logics simply by translating each predicate $\text{Reach}(x, y)$ to $x = y \vee \exists W \exists t_1^W \exists t_2^W (t_1 < t_2 \wedge t_1 \sim x \wedge t_2 \sim y)$.

We can translate WL^∞ to WL as follows. Consider a formula $\exists^\infty W\, \psi$ with ψ a conjunction of atoms (disjunctions can be brought to the top and each disjunct treated separately). We can equivalently express this as $\exists W\, \psi'$ where ψ' is obtained from ψ by adding a position variable l^W that is stated to be strictly larger than all position variables of sort W in ψ, and stating that from l a cycle is reachable:

$$\exists C \exists t_1^C \exists t_2^C \exists t_3^C (t_1 < t_2 < t_3 \wedge l \sim t_1 \wedge t_2 \sim t_3).$$

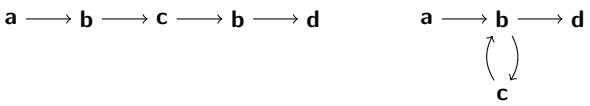
In this way all infinite quantifiers can be eliminated.

We can translate WL to FO(Reach) as follows. Consider a conjunction of atoms ψ in WL. By making a disjunction of all possible completions, we may assume that the conjunction is complete in the sense that all position variables of a same walk variable are totally ordered by the $<$ predicate. We construct a conjunction of atoms ψ' in FO(Reach) by treating each position variable as if it were a node variable; replacing each predicate $t_1 < t_2$ by $\text{Reach}(t_1, t_2)$; and replacing each comparison $t_1 \sim t_2$ by $t_1 = t_2$. We claim that the existential closure $\bar{\exists}\psi$ of ψ is equivalent to $\bar{\exists}\psi'$. To prove the claim, let G be a graph. First consider a WL assignment α on G, appropriate for ψ, such that $G, \alpha \models \psi$. We can turn α into an FO(Reach) assignment α' on G by defining $\alpha'(t^W)$ to be the node in G under position $\alpha(t)$ in walk $\alpha(W)$. Then $G, \alpha' \models \psi'$. Conversely, consider an FO(Reach) assignment α' on G such that $G, \alpha' \models \psi'$. We can define a WL assignment α on G by mapping each walk variable W to a walk that joins all paths implied by predicates $\text{Reach}(t_1^W, t_2^W)$ together in the order consistent with the ordering on W's position variables given in ψ. We then map the position variables accordingly.

Finally we must give a query expressible in positive-existential PL that is not expressible in positive-existential FO(Reach). Note that yes/no queries in the latter logic are preserved under homomorphism. Such a query is the following:

$$\exists P \exists t_1^P \exists t_2^P \exists t_3^P (t_1 < t_2 < t_3 \wedge a(t_1) \wedge c(t_2) \wedge d(t_3)).$$

To see the inexpressibility, consider the following two graphs:



There is a homomorphism from the left graph to the right graph, but the left graph satisfies the query while the right graph does not. □

8. CONCLUSION

As a conceptual contribution we have proposed Walk Logic (WL) as a framework for the investigation of the expressive power of path query languages for graph databases. We have identified two themes that challenge our understanding, namely, *finite versus infinite walks*, and *paths versus walks*. We have shown that our theory is workable by showing various results that compare the expressive power of path query languages. At the same time we have identified a number of open problems for further research. We repeat a few here:

- What is the precise data complexity of WL? Is it subsumed by another natural powerful query language?
- Is WL^∞ strictly more expressive than WL? The same question can be asked for HCTL^* and regular walk logics.
- Characterize the expressive power of HCTL^* on finite graphs.

Acknowledgment

Jan Van den Bussche thanks Balder ten Cate for a helpful conversation on the expressive power of hybrid modal logics, and Carlos Areces for answering a question about the state of the art in hybrid modal logics.

9. REFERENCES

[1] S. Abiteboul, R. Hull, and V. Vianu. *Foundations of Databases*. Addison-Wesley, 1995.

[2] R. Angles and C. Gutierrez. Survey of graph database models. *ACM Computing Surveys*, 40(1):article 1, 2008.

[3] C. Areces, P. Blackburn, and M. Marx. Hybrid logics: Characterization, interpolation and complexity. *Journal of Symbolic Logic*, 66(3):977–1010, 2001.

[4] M. Arenas, S. Conca, and J. Pérez. Counting beyond a Yottabyte, or how SPARQL 1.1 property paths will prevent adoption of the standard. In A. Mille et al., editors, *Proceedings 21st World Wide Web Conference*, pages 629–638. ACM, 2012.

[5] C. Baier and J.-P. Katoen. *Principles of Model Checking*. MIT Press, 2008.

[6] P. Barceló, C.A. Hurtado, L. Libkin, and P.T. Wood. Expressive languages for path queries over graph-structured data. In *Proceedings 29th ACM Symposium on Principles of Databases*, pages 3–14. ACM, 2010.

[7] P. Barceló, J. Pérez, and J.L. Reutter. Relative expressiveness of nested regular expressions. In J. Freire and D. Suciu, editors, *Proceedings 6th Alberto Mendelzon International Workshop on Foundations of*

Data Management, volume 866 of *CEUR Workshop Proceedings*, pages 180–195, 2012.

[8] M. Benedikt, M. Grohe, L. Libkin, and L. Segoufin. Reachability and connectivity queries in constraint databases. *Journal of Computer and System Sciences*, 66(1):169–206, 2003.

[9] M. Benedikt, C. Ley, and G. Puppis. Automata vs logics on data words. In A. Dawar and H. Veith, editors, *Computer Science Logic*, volume 6247 of *Lecture Notes in Computer Science*, pages 110–124. Springer, 2010.

[10] M. Benedikt, L. Libkin, T. Schwentick, and L. Segoufin. Definable relations and first-order query languages over strings. *Journal of the ACM*, 50(5):694–751, 2003.

[11] M.R.F. Benevides and L.M. Schechter. Using modal logics to express and check global graph properties. *Logic Journal of the IGPL*, 17(5):559–587, 2009.

[12] M. Bojanczyk, C. David, A. Muscholl, T. Schwentick, and L. Segoufin. Two-variable logic on data words. *ACM Transactions on Computational Logic*, 12(4):article 27, 2011.

[13] D. Calvanese, G. De Giacomo, M. Lenzerini, and M.Y. Vardi. Reasoning on regular path queries. *SIGMOD Record*, 32(4):83–92, 2003.

[14] E.M. Clarke, O. Grumberg, and D.A. Peled. *Model Checking*. MIT Press, 2000.

[15] M. Consens and A. Mendelzon. GraphLog: A visual formalism for real life recursion. In *Proceedings of the Ninth ACM Symposium on Principles of Database Systems*, pages 404–416. ACM Press, 1990.

[16] B. Courcelle and J. Engelfriet. *Graph Structure and Monadic Second-Order Logic*. Cambridge University Press, 2012.

[17] Michel de Rougemont. Second-order and inductive definability on finite structures. *Zeitschr. f. math. Logik und Grundlagen d. Math.*, 33:47–63, 1987.

[18] V. Diekert and P. Gastin. First-order definable languages. In J. Flum, E. Grädel, and T. Wilke, editors, *Logic and Automata, History and Perspectives*, pages 261–306. Amsterdam University Press, 2008.

[19] R. Diestel. *Graph Theory*. Springer, fourth edition, 2010.

[20] H.-D. Ebbinghaus and J. Flum. *Finite Model Theory*. Springer, second edition, 1999.

[21] S. Fortune, J. Hopcroft, and J. Wyllie. The directed subgraph homeomorphism problem. *Theoretical Computer Science*, 10(2):111–121, 1980.

[22] M. Franceschet and M. de Rijke. Model checking hybrid logics, with an application to semistructured data. *Journal of Applied Logic*, 4(3):279–304, 2006.

[23] D.D. Freydenberger and N. Schweikardt. Expressiveness and static analysis of extended conjunctive regular path queries. In P. Barceló and V. Tannen, editors, *Proceedings 5th Alberto Mendelzon International Workshop on Foundations of Data Management*, volume 749 of *CEUR Workshop Proceedings*, 2011.

[24] V. Goranko and M. Otto. Model theory of modal logic. In P. Blackburn, J. van Benthem, and F. Wolter, editors, *Handbook of Modal Logic*, chapter 5. Elsevier, 2007.

[25] E. Grädel, Ph. Kolaitis, L. Libkin, M. Marx, J. Spencer, M. Vardi, Y. Venema, and S. Weinstein. *Finite Model Theory and Its Applications*. Springer, 2007.

[26] T. Hafer and W. Thomas. Computation tree logic and path quantifiers in the monadic theory of the binary tree. In T. Ottmann, editor, *Automata, Languages and Programming, 14th International Colloquium*, volume 267 of *Lecture Notes in Computer Science*, pages 269–279. Springer, 1987.

[27] N. Immerman. *Descriptive Complexity*. Springer, 1999.

[28] Ph.G. Kolaitis. On the expressive power of logics on finite models. In *Finite Model Theory and Its Applications* [25], chapter 2.

[29] L. Libkin. *Elements of Finite Model Theory*. Springer, 2004.

[30] K. Losemann and W. Martens. The complexity of evaluating path expressions in SPARQL. In *Proceedings 31st ACM Symposium on Principles of Databases*, pages 101–112. ACM, 2012.

[31] R. McNaughton and S. Papert. *Counter-Free Automata*. MIT Press, 1971.

[32] F. Moller and A.M. Rabinovich. Counting on CTL*: On the expressive power of monadic path logic. *Information and Computation*, 184(1):147–159, 2003.

[33] M. Otto. Model theoretic methods for fragments of FO and special classes of (finite) structures. In J. Esparza, C. Michaux, and C. Steinhorn, editors, *Finite and Algorithmic Model Theory*, volume 379 of *Lecture Note Series*, chapter 7. London Mathematical Society, 2011.

[34] L.J. Stockmeyer. The complexity of decision problems in automata theory and logic. Technical Report MAC TR-133, MIT Project MAC, Cambridge, MA, 1974.

[35] W. Thomas. Languages, automata, and logic. In G. Rozenberg and A. Salomaa, editors, *Handbook of Formal Languages*, volume 3, chapter 7. Springer, 1997.

[36] M. Vardi. The complexity of relational query languages. In *Proceedings 14th ACM Symposium on the Theory of Computing*, pages 137–146, 1982.

[37] V. Weber. Branching-time logics repeatedly referring to states. *Journal of Logic, Language and Information*, 18(4):593–624, 2009.

[38] P. Wood. Query languages for graph databases. *SIGMOD Record*, 41(1):50–60, March 2012.

[39] W. Zhang. Bounded semantics of CTL and SAT-based verification. In K. Breitman and A. Cavalcanti, editors, *Proceedings 11th International Conference on Formal Engineering Methods*, volume 5885 of *Lecture Notes in Computer Science*, pages 286–305. Springer, 2009.