On the Complexity of Kings

Edith Hemaspaandra¹
Department of Computer Science
Rochester Institute of Technology
Rochester, NY 14623 USA

Lane A. Hemaspaandra²
Department of Computer Science
University of Rochester
Rochester, NY 14627 USA

Till Tantau³
Institut für Theoretische Informatik
Universität zu Lübeck
D-23538 Lübeck, Germany

Osamu Watanabe⁴
Dept. of Mathematical &
Computing Sciences
Tokyo Institute of Technology
Tokyo, 152-8552, Japan

December 1, 2006; revised November 15, 2007

¹www.cs.rit.edu/~eh/. Supported in part by grants NSF-CCR-0311021 and NSF-IIS-0713061 and by a TransCoop grant. Work done in part while on sabbatical at the University of Rochester and while visiting the Tokyo Institute of Technology.

²www.cs.rochester.edu/u/lane/. Supported in part by grant NSF-CCF-0426761, JSPS Invitation Fellowship S-05022, a TransCoop grant, and a Friedrich Wilhelm Bessel Research Award. Work done in part while visiting the Tokyo Institute of Technology.

³www.tcs.uni-luebeck.de/mitarbeiter/tantau/.

Abstract

A king in a directed graph is a vertex from which each vertex in the graph can be reached via paths of length at most two. There is a broad literature on tournaments (completely oriented digraphs), and it has been known for more than half a century that all tournaments have at least one king [Lan53]. Recently, kings have proven useful in theoretical computer science, in particular in the study of the complexity of reachability problems [NT05] and semifeasible sets [HNP98, HT06, HOZZ06].

In this paper, we study the complexity of recognizing kings. For each succinctly specified family of tournaments, the king problem is already known to belong to $\Pi^p_2$ [HOZZ06]. We prove that the complexity of kingship problems is a rich enough vocabulary to pinpoint every nontrivial many-one degree in $\Pi^p_2$. That is, we show that every set in $\Pi^p_2$ other than $\emptyset$ and $\Sigma^*$ is equivalent to a king problem under $\leq^m_p$-reductions. Indeed, we show that the equivalence can even be instantiated via relatively simple padding, and holds even if the notion of kings is redefined to refer to $k$-kings (for any fixed $k \geq 2$)—vertices from which all vertices can be reached via paths of length at most $k$. In contrast, we prove that recognizing whether a given vertex is a source (i.e., there exists a $k$ such that it is a $k$-king) yields languages that also fall within $\Pi^p_2$, yet cannot be $\Pi^p_2$-complete—or even NP-hard—unless $P = NP$.

Using these and related techniques, we obtain a broad range of additional results about the complexity of king problems, diameter problems, and radius problems. It follows easily from our proof approach that the problem of testing kingship in succinctly specified graphs (which need not be tournaments) is $\Pi^p_2$-complete. We show that the radius problem for arbitrary succinctly represented graphs is $\Sigma^p_3$-complete, but that in contrast the diameter problem for arbitrary succinctly represented graphs (or even tournaments) is $\Pi^p_2$-complete.
1 Introduction

1.1 Problem Statement and Main Result

In this paper, we study the complexity of recognizing kings and \( k \)-kings in graphs. For each \( k \geq 0 \), a \( k \)-king of a graph is a vertex such that every vertex can be reached from it via a path of length at most \( k \). By convention, a vertex of a graph is said to be a king if every vertex of the graph can be reached from it via a path of length at most two. So “king” and “2-king”—and “kingship” and “2-kingship”—are synonymous in this paper, as they also are in the general literature on kings.

In the \( k \)-kingship problem we are given a graph and a vertex as inputs and would like to tell whether the vertex is a \( k \)-king. We can vary the problem by allowing different ways of encoding graphs (the more succinctly, the harder the problem) and by allowing different kinds of input graphs (the more restricted, the easier the problem).

Much is known about the existence of kings in graphs. For example, in the 1950s Landau [Lan53] discovered the simple but lovely result that every tournament has a king. A tournament is a directed graph \( G \) such that for each pair \( u \) and \( v \) of distinct vertices exactly one of the directed edges \( u \to v \) or \( v \to u \) is present in the graph and such that there are no loops. A well-known (see [Wes01]) way to easily see that Landau’s result holds is to note that every vertex with maximum degree must be a king. More recently, similar results were proven for generalizations of tournaments, such as multipartite tournaments ([Gut86, PT91], see also [BG98] and the references therein).

When graphs are specified explicitly in the natural way (say, via an adjacency matrix), it is not hard to see that for each \( k \geq 0 \) the \( k \)-kingship problem is first-order definable and thus very simple from a computational point of view. However, when we specify graphs succinctly, for each \( k \geq 1 \) the complexity of \( k \)-kingship problems jumps from having an upper bound of “first-order definable” to having a lower bound of “coNP-hard” (or worse). There are different ways of specifying graphs succinctly, ranging from the general Galperin–Wigderson model to the polynomial-time uniform tournament family specifiers that arise in the study of semifeasible sets.

In the Galperin–Wigderson model, input graphs are specified as follows: A directed graph \( G \) with a vertex set \( \{0, 1\}^n \) is specified using a circuit \( C \) with \( 2n \) input gates and one output gate. For any two vertices \( x, y \in \{0, 1\}^n \) there is an edge \( x \to y \) in \( G \) if and only if \( C(xy) = 1 \). (This definition does allow the possibility of self-loops.) Note that a circuit whose size is polynomial in \( n \) can encode a graph whose vertex set has size \( 2^n \), which is exponential in \( n \). For this model, the \( k \)-kingship problem can be formalized as follows:

\[
\text{SUCCINCT-} k \text{-KINGS} = \{ (\text{code}(C), x) \mid C \text{ specifies (in the manner specified above) a graph } G \\
\text{in which } x \text{ is a } k \text{-king} \}.
\]

In the above definition, we used \( \text{code}(C) \) to denote a standard binary encoding of the circuit \( C \). Furthermore, \( (\text{code}(C), x) \) is a standard binary encoding of the circuit \( C \) paired with a bitstring \( x \). The exact definition of the pairing function \( (.,.) \) is detailed in the preliminaries section.
In the tournament family specifier model, input graphs are specified using polynomial-time computable, commutative selector functions. A selector function $f$ gets two words $u$ and $v$ as inputs and outputs one of them, thereby telling us where the edge between $u$ and $v$ heads. More formally, a selector function $f: \{0, 1\}^* \times \{0, 1\}^* \to \{0, 1\}^*$ defines an (infinite) graph with the vertex set $\{0, 1\}^*$ where there is an edge from $u$ to $v$ if and only if $f(u, v) = v$ and $u \neq v$. The graph will be a tournament if $f$ is commutative, that is, if for each $u$ and $v$ it holds that $f(u, v) = f(v, u)$. Polynomial-time selector functions were originally introduced by Selman [Sel79, Sel81, Sel82] in his study of so-called P-selective sets: P-selective set can be defined as the sets of vertices in tournaments specified by polynomial-time computable commutative selector functions that are closed under reachability.

Instead of using a selector function to describe a single infinite tournament, we can also use them to describe one tournament per word length: Given a commutative selector function $f$ and a word length $n$, we say that $f$ describes the length-$n$ tournament whose vertex set is $\{0, 1\}^n$ and whose edge set is defined as above: There is an edge from $u \in \{0, 1\}^n$ to $v \in \{0, 1\}^n$ if $f(u, v) = v$ and $u \neq v$. In this model we can also consider the $k$-kingship problem:

$$k\text{-Kings}_f = \{x \in \{0, 1\}^* \mid x \text{ is a } k\text{-king in the length-}|x| \text{ tournament specified by } f\}.$$  

One can view the $k\text{-Kings}_f$ problems (one for each $f$) as very restricted cases of the more general succinct- $k$-kings problem: For the $k\text{-Kings}_f$ we must check $k$-kingship for a single tournament per word length.

Note that in complexity terms, this tremendous uniformity of specification—a polynomial-time computable function specifying for us a single tournament at each length—will naturally tend to tie our hands in terms of showing hardness for higher levels of the polynomial hierarchy. Nonetheless, what we will actually show in this paper is that we can free our hands from those cords. In particular, Corollary 4.2 shows that there is such a selector-specified kings problem that is $\Pi^p_2$-complete for some fixed selector function. Since the analogous problem, even when specified on the fly by circuits that are part of the input (and thus can change the tournament with each input, even within a given length), is clearly always in $\Pi^p_2$, this shows that being trapped to just having one structure per length to work on does not exact a price in terms of the level of complexity that holds. In particular, and this is the core of this paper, we ensure that the one-tournament-per-length has such potential structural richness that asking about different vertices within that structure allows us to achieve the desired hardness level for the underlying $\Pi^p_2$ problem that the tournament is seeking to reframe as a kings problem.

A language $L \subseteq \{0, 1\}^*$ for which there exists a commutative polynomial-time selector function $f$ such that $L = k\text{-Kings}_f$ will be called a P-$k$-king language. Our main interest in this paper is to study which languages are P-$k$-king languages. Our main result is that, for each $k \geq 2$, every language in $\Pi^p_2 - \{\emptyset, \Sigma^*\}$ is many-one equivalent to a P-$k$-king language. Informally put, this shows that $k$-kings languages are comprehensively descriptive in terms of naming the complexity of the nontrivial $\Pi^p_2$ many-one degrees.

We obtain this main result via an even stronger main tool, which shows something
about the uniformity and simplicity of a set of reductions that can instantiate the above equivalences. Namely, we show that, for every $k \geq 2$, a language $L$ is in $\Pi^p_2$ if and only if $\text{pad}^j_{\ell}(L)$ is a $P$-$k$-king language for some $j$. Here, pad$^j_{\ell}$ is a padding operator whose exact definition will be given later.

1.2 Motivations for Studying P-$k$-King Languages

Our study of P-$k$-king languages is motivated from several contrasting directions.

Relationship to the radius problem. Kings (recall: as is standard, by this we mean 2-kings) and $k$-kings are closely related to the radius problem for graphs. A ball of radius $r$ around a vertex $v$ is the set of vertices that can be reached from $v$ within $r$ steps. The radius of a graph is the smallest radius of a ball that covers the whole graph. This means that the radius of a graph is at most $r$ if and only if there exists an $r$-king in the graph. (Note, in contrast, that the $k$-king problems focus on whether a given vertex, which is explicitly stated as part of the input, is a $k$-king.)

We use our results on P-$k$-king languages to give a short proof that radius problems for succinctly specified graphs (using the Galperin–Wigderson model) are complete for the third level of the polynomial hierarchy [MS72, Sto76], i.e., are complete for $\Sigma^p_3 = \text{NP}^{\text{NP}}$. This result is interesting in its own right. While for the first level of the polynomial hierarchy (NP) countless natural complete problems are known, for higher levels the collection of such problems is less extensive (see also Section 1.3’s comments on complete sets for such classes). The succinct radius problem is a new and fairly natural problem that is complete for $\Sigma^p_3$.

Relationship to the diameter problem. Kings and $k$-kings are also closely related to the diameter problem for graphs. The diameter of a graph is the maximum over all ordered vertex pairs of the shortest distance (via a directed path) from the first vertex of the pair to the second vertex of the pair. (If the second vertex isn’t reachable from the first, this distance is $\infty$.) This means that a graph has diameter at most $d$ if and only if every vertex of the graph is a $d$-king of the graph.

Based on this relationship we show that diameter problems for succinctly represented graphs are complete for $\Pi^p_2 = \text{coNP}^{\text{NP}}$.

Relationship to P-selective sets. P-2-king languages are closely related to P-selective languages. For a P-selective language $A$, for each $n$, within the length-$n$ graph specified by the selector function it always holds that the reachability closure of the length $n$ words of $A$ is precisely the length $n$ words of $A$. For a P-2-king language, the words in the language of length $n$ are the kings in the length-$n$ tournament specified by the selector function. This means that for a P-selective set the kings of the tournaments induced by a selector are (speaking very informally) the “least likely” words to be contained in the language. More precisely, unless all words of a given word length are in the language, none of the kings
of the tournament specified by the selector for this word length is in the language. This observation can be used to show that P-selective sets cannot be \( \Pi_2^p/1 \)-immune [HT06].

**Relationship to the second level of the polynomial hierarchy.** Despite the close relationship of P-2-king languages and P-selective languages, there are fundamental differences. For example, it is easy to see that all P-\( k \)-king languages are in \( \Pi_2^p \), see [HOZZ06] for a detailed proof, but it is well known that P-selective languages can be “arbitrarily complex” in a sense that can be crisply formalized (for example, Selman’s seminal papers on P-selectivity established that for every tally language \( A \) there is a P-selective set that is \( \leq^p_{\text{many}} \)-equivalent to \( A \)). This encourages us to investigate which languages are P-\( k \)-king languages. Many languages in \( \Pi_2^p \) are not P-\( k \)-king languages—for example, since every tournament has a king, a P-2-king language contains at least one word for every word length. However, the tool underpinning our main result shows that for every \( k \geq 2 \) and every language \( L \in \Pi_2^p \) a certain padded version of \( L \) is a P-\( k \)-king language. Thus, although not every language in \( \Pi_2^p \) is a P-\( k \)-king language, for every such language a closely related language is a P-\( k \)-king language. And from this we have our main result, which is that every \( \Pi_2^p \) (many-one) degree, except those of \( \emptyset \) and \( \Sigma^* \), contains a P-\( k \)-king language. In fact, something even stronger than many-one degrees holds, due to the the precise proof we use to show this. We in fact establish that for every \( k \geq 2 \) every language in \( \Pi_2^p \) except \( \emptyset \) and \( \Sigma^* \) is equivalent to a P-\( k \)-king language even under first-order reductions. In particular, for every \( k \geq 2 \) there exist P-\( k \)-king languages that are complete for \( \Pi_2^p \) with respect to first-order reductions.

**Relationship to quantifier characterizations.** The quantifier characterization of the polynomial hierarchy states that a language \( L \) is in \( \Pi_2^p \) if and only if there exist a polynomial \( p \) and a ternary polynomial-time decidable relation \( R \) such that \( x \in L \iff (\forall y \in \{0, 1\}^{p(|x|)})(\exists z \in \{0, 1\}^{p(|x|)})[R(x, y, z)] \). For P-2-king languages a more restrictive characterization is possible: A language \( L \) is a P-2-king language if and only if there exists a binary polynomial-time decidable relation \( S \) such that \( x \in L \iff (\forall y \in \{0, 1\}^{|x|})(\exists z \in \{0, 1\}^{|x|})[S(x, y) \land S(y, z)] \) and such that for all distinct \( x, y \in \{0, 1\}^* \) we have \( S(x, y) \iff \neg S(y, x) \).

**Relationship to the top Toda equivalence classes of tournaments.** Kings and \( k \)-kings can be used in the study of the top Toda equivalence classes of tournaments. Given a commutative P-selector \( f \) and a word length \( n \), two elements of the length-\( n \) tournament induced by \( f \) are said to be \textit{Toda equivalent} if there is path from the first to the second element and also a path from the second to the first element. For each word length there exists what is called a \textit{top Toda equivalence class} [HOZZ06], by which we mean the unique equivalence class (in that tournament) whose members are not pointed to by members of any other equivalence class of that tournament. Note that a vertex \( v \) will belong to the top Toda equivalence class precisely if all vertices in the tournament are reachable from \( v \). So asking whether a vertex belongs to the top Toda equivalence class is the same as asking...
whether the vertex has the property that there exists a $k$ for which the vertex is a $k$-king. Analogously to P-$k$-king languages, we can define top-Toda languages for commutative P-selectors $f$:

$$\text{Top-Toda}_f = \{ x \in \{0, 1\}^* \mid \text{there exists a } k \text{ such that } x \text{ is a } k\text{-king in the length-}|x| \text{ tournament specified by } f \}.$$ 

At first sight, these languages might appear to be more difficult than P-$k$-king languages: Instead of checking whether there is a path of length at most $k$ from the input to each vertex in the tournament, we must ask the same question for the richer universe of all paths (i.e., without an upper bound of $k$ being imposed on their lengths). However, we prove that all these languages are also in $\Pi^p_2$. Moreover, we prove that none of these languages can be $\Pi^p_2$-complete—or even NP-hard—unless P = NP. Comparing this to P-$k$-king languages, which a corollary to our main result shows are sometimes $\Pi^p_2$-complete, we see that all top-Toda languages are seemingly easier than some P-$k$-king languages.

### 1.3 Related Work

The work most closely related to that of this paper is the work of Nickelsen and Tantau on the complexity of reachability problems [NT05], the path-breaking modeling and complexity work of Galperin and Wigderson [GW96], and the existing work on the complexity of kings and in particular their use in the study of the semifeasible sets [HNP98, HT06, HOZZ06].

It is well worth mentioning that without the seminal work of Landau [Lan53], which showed that kings always exist in tournaments, it is unlikely that the notion of kings would even be available for study. And Landau’s work has led to a rich (though, naturally, not complexity-theoretic) body of work on the existence of kings or $k$-kings in a variety graph-theoretic structures (for example, for the case of multipartite tournaments see [Gut86, PT91, BG98] and the references therein).

For reasons of focus and coherence, all tournaments in this paper follow the typical notion of a tournament. However, we mention in passing that one central result of this paper, our $\Pi^p_2$-completeness result for the $k$-kings problem, has been studied for the case of $j$-partite tournaments (though in a more circuit-focused model) in the June 2005 technical report version of this paper (available at arXiv.org), where a dichotomy theorem is given that completely characterizes what happens in that case, namely, for the boundary case of “1-kingship” one gets P algorithms and for all other cases $\Pi^p_2$-completeness holds. We also mention that though the results of this paper are fundamentally complexity-theoretic in nature, one can also studying their recursion-theoretic siblings, and can obtain Kleene-hierarchy versions of the polynomial-hierarchy results presented here.

In this paper, we will show problems to be complete for classes at the second and third levels of the polynomial hierarchy. These levels have nothing resembling the range and number of known, natural complete problems that NP has (see, for example, the famous compendium of Garey and Johnson [GJ79]). Nonetheless, these levels do have a larger range and number of known, natural complete problems than many people realize. Schaefer
and Umans have provided a very nice “Garey and Johnson” for classes at levels of the polynomial hierarchy beyond the first [SU02a, SU02b].

1.4 Organization of This Paper

Section 2 provides notations, definitions, and some important lemmas that we will need in the paper’s result sections. Section 3 studies the complexity of the diameter problem, and shows that it is complete for the $\Pi^p_2$ level of the polynomial hierarchy. That section also introduces tools that will be used in subsequent proofs in the paper. Section 4 proves our main result, namely, that $k$-kings problems have the descriptive flexibility to name every nontrivial $\Pi^p_2$ degree. We do so by showing that for each $k \geq 2$ it holds that via a certain family of padding functions each $\Pi^p_3$ language can be turned into a P-$k$-king language. Section 5 studies the complexity of the radius problem, and shows that it is complete for the $\Sigma^p_3$ level of the polynomial hierarchy, which provides an interesting contrast with Section 3’s $\Pi^p_2$-completeness result for the diameter problem. Section 6 studies top-Toda languages and shows that although such languages are always in $\Pi^p_2$, they cannot be $\Pi^p_2$-complete (or even NP-hard) unless $P = NP$. Section 7 is our conclusion.

2 Basic Definitions and Tools

In this section we introduce the notation and terminology used in the rest of the paper. We also prove lemmas on key concepts, which will be used later in the paper.

2.1 Alphabets and Reductions

Throughout this paper $\Sigma = \{0, 1\}$.

In this paper we use two kinds of reductions. When we need a restrictive reduction, we will use first-order reductions [Imm98], written $\leq_{fo}$, where ordering and the bit predicate are available. First-order reductions are the same as DLOGTIME-uniform many-one AC$^0$-reductions, see Barrington et al. [BIS90]. When we need a less restrictive reduction (or when we simply are trying to use the type of reductions that are most commonly used and understood in the field), we use the more widely known polynomial-time many-one reduction, written $\leq_{m}^p$. Note that $A \leq_{fo} B$ implies $A \leq_{m}^p B$.

2.2 Bitstrings, Alphabets, and Padding

We refer to elements of $\{0, 1\}^* = \Sigma^*$ as bitstrings. The length of a bitstring $b$ is denoted $|b|$. We define a pairing function $\langle ., . \rangle : \Sigma^* \times \Sigma^* \rightarrow \Sigma^*$ as follows (unlike some other papers, we will not require our pairing function to be a surjective function): For every two bitstrings $x, y \in \Sigma^*$ where the individual bits of $x$ are $x_1$ to $x_n$, let $\langle x_1, x_2 \ldots x_n, y \rangle = 0x_10x_2 \ldots 0x_n1y$. This function, which clearly is injective, has a number of useful properties. Those that will be used in this paper are listed in the following lemma, whose straightforward proof is omitted.
Lemma 2.1. The pairing function $\langle.,.\rangle: \Sigma^* \times \Sigma^* \rightarrow \Sigma^*$ has the following properties:

1. It is polynomial-time computable.

2. It is polynomial-time invertible (its range is in P and there exist two polynomial-time computable functions $\sigma_1$ and $\sigma_2$ such that, given a string $z$ in the range of $\langle.,.\rangle$, it holds that $\langle\sigma_1(z),\sigma_2(z)\rangle = z$).

3. For all words $x,y,x',y' \in \Sigma^*$ with $|x| = |x'|$ and $|y| = |y'|$ we have $|\langle x,y \rangle| = |\langle x',y' \rangle|$.

4. The range of the function does not include any word from $\{0\}^*$ (no word pair is mapped to an element of $\{0\}^*$).

For a tuple $(x_1,\ldots,x_n)$ of words, $n \geq 1$, let $\langle x_1,\ldots,x_n \rangle = \langle x_1,\langle x_2,\ldots,\langle x_{n-1},x_n \rangle \rangle \cdots \rangle$.

For a positive integer $j$, we define a padding function $\text{pad}_j: \Sigma^* \rightarrow \Sigma^*$ by

$$\text{pad}_j(x) = x0^{|x|+j+3}.$$ 

Thus we add $|x|+j+3$ zeros after $x$. The reason for the slightly startling “+ 3” will become clear later on. Note that for every word $y \in \Sigma^*$ there can be at most one word $x$ such that $\text{pad}_j(x) = y$ and, if such an $x$ exists, it is easy to compute.

We define two padded versions of languages. The “usual” way to define a padded version of a language $L$ is to consider the image of $L$ under the padding function $\text{pad}_j$, that is, for a given language $L \subseteq \Sigma^*$ let $\text{pad}_j(L) = \{\text{pad}_j(x) \mid x \in L\}$. The “interesting” words in a padded language are those in $\text{pad}_j(\Sigma^*)$. Words outside $\text{pad}_j(\Sigma^*)$ are not in $\text{pad}_j(L)$. For the second padded version of $L$ we change this latter property: The membership of the words in $\text{pad}_j(\Sigma^*)$ is the same, but (almost) all other words are in the second padded version. Formally, for a language $L \subseteq \Sigma^*$ we define

$$\text{pad}_j'(L) = \text{pad}_j(L) \cup (\Sigma^* - \text{pad}_j(\Sigma^*) - \{1,11\}).$$

Once more, there is a startling part of the definition, namely the “$- \{1,11\}$” and, once more, this will be explained later on. The important properties of the padded versions of a language $L$ are listed in the next lemma.

Lemma 2.2. Let $L \subseteq \Sigma^*$ and let $j$ be a positive integer. The language $L' = \text{pad}_j'(L)$ has the following properties:

1. For all words $x \in \Sigma^*$: $x \in L$ if and only if $\text{pad}_j(x) \in L'$.

2. $L'$ contains one word of length 0, one word of length 1 and three words of length 2.

3. For all word lengths $n \geq 3$ that are not of the form $n = m + mj + j + 3$ for some nonnegative integer $m$, all words of length $n$ are in $L'$.

4. For every word length $n$ that is of the form $n = m + mj + j + 3$ for some nonnegative integer $m$, all words of length $n$ that do not end with $mj + j + 3$ zeros are in $L'$. 

7
5. If \( L \neq \emptyset \) and \( L \neq \Sigma^* \), then \( L \) and \( L' \) are \( \leq_f \)-equivalent.

Proof. For the first claim, just note that \( \text{pad}_j'(L) \) is the same as \( \text{pad}_j(L) \) when we consider only words in the range of \( \text{pad}_j \).

For the second claim, first note that \( \text{pad}_j \) does not map any word to a word of length less than 3. (This is why we added 3 in the definition of the padding function.) But then the words in \( L' \) of lengths 0, 1, and 2 are the empty word, the word 0, and the three words 00, 01, and 10.

The third and fourth claims follow directly from the definition.

The first-order reduction from \( L \) to \( L' \) is given by the function \( \text{pad}_j \), which is clearly very easy to compute. For the reduction in the other direction we first check whether the input is 1 or 11. In either case we map this input to a fixed word known not to be in \( L \). Otherwise we check whether the word has length \( m + m^j + j + 3 \) for some \( m \) and whether it ends with \( m^j + j + 3 \) zeros. If so, we map the word to its first \( m \) bits, otherwise to some arbitrary fixed word that is known to be in \( L \).

\[ \square \]

2.3 Graphs and Tournaments

A directed graph is a pair \( (V, E) \) consisting of a nonempty vertex set \( V \) together with an edge set \( E \subseteq V \times V \). An undirected graph is a directed graph with a symmetric edge relation, that is, \( (u, v) \in E \) holds if and only if \( (v, u) \in E \) holds. Another common way of formalizing undirected graphs is to use size-two sets for the edges, but we will use symmetric edge relations. Instead of \( (u, v) \in E \) we also write \( u \rightarrow_v E v \) or just \( u \rightarrow v \) when \( E \) is clear from context. The out-degree of a vertex \( v \) in a graph \( (V, E) \) is the number of vertices \( v \) for which \( u \rightarrow v \) holds. The in-degree is defined analogously.

A path of length \( l \) in a graph is sequence \( v_0, v_1, \ldots, v_l \) of vertices such that \( v_{i-1} \rightarrow v_i \) holds for all \( i \in \{1, \ldots, l\} \). For integers \( k \geq 0 \), a \( k \)-king of a graph is a vertex \( v \) such that there is a path of length at most \( k \) from \( v \) to every other vertex. (Note that the only graph that has a 0-king is the graph consisting of a single vertex.) A 2-king is also just called a king. The diameter of a graph is the smallest number \( d \) such that for every pair \( u, v \in V \) of vertices there is a path from \( u \) to \( v \) of length at most \( d \). Note that the diameter of a graph is exactly the smallest number \( k \) such that every vertex of the graph is a \( k \)-king. If a graph has more than one strongly connected component, its diameter is \( \infty \). The radius of a graph is the smallest number \( r \) such that there exists a vertex \( v \) from which there are paths of length at most \( r \) to all other vertices. Note that the radius of a graph is exactly the smallest number \( k \) such that there exists a \( k \)-king in the graph. It is possible for a graph (though, as we will see, not for a tournament) to have a radius of \( \infty \).

A tournament is a directed graph such that (a) there are no self-loops, i.e., the edge relation \( E \) is irreflexive and (b) for every pair \( u, v \in V \) of distinct vertices we have either \( (u, v) \in E \) or \( (v, u) \in E \), but not both. It is well known that any vertex of maximal out-degree in a tournament is a king of the tournament. In particular, every tournament has a king.
Except for the tournament consisting of a single vertex or of no vertices, a tournament obviously cannot have a diameter strictly less than 2. In the following, we will often need to construct tournaments that have diameter exactly 2. The following lemmas show when and how this can be done, but first we need a definition.

**Definition 2.3.** Let \( G = (V,E) \) be a graph. Let \( u \) and \( v \) be two new vertices that are not in \( V \). Let \( G'[u,v] = (V',E') \) be the following graph: \( V' = V \cup \{u,v\} \) and \( E' = E \cup \{(u,v)\} \cup \{(x,u) \mid x \in V\} \cup \{(v,x) \mid x \in V\} \).

The definition states that we obtain \( G'[u,v] \) from \( G \) by adding the vertices \( u \) and \( v \) and adding edges from all old vertices to \( u \), adding an edge from \( u \) to \( v \), and adding edges from \( v \) to all old vertices.

**Lemma 2.4.** Let \( T = (V,E) \) be a tournament and let \( u,v \notin V \) be two new vertices. Let \( K \subseteq V \) be the set of kings in \( T \). Then the set of kings of \( T[u,v] \) is \( K \cup \{u,v\} \).

In particular, if \( T \) has diameter 2, then \( T[u,v] \) has diameter 2.

**Proof.** To prove the lemma we need to show that \( u \) and \( v \) are kings of \( T[u,v] \) and, furthermore, that a vertex \( x \in V \) is a king of \( T[u,v] \) if and only if \( x \) is a king in \( T \).

First, \( v \) is a king in \( T[u,v] \) as there are direct edges from \( v \) to all \( x \in V \) and there is a path \( v \rightarrow x \rightarrow u \) in \( T[u,v] \), where \( x \in V \) is an arbitrary vertex.

Second, \( u \) is a king in \( T[u,v] \) as there is an edge \( u \rightarrow v \) and for all \( x \in V \) there is a path \( u \rightarrow v \rightarrow x \).

Third, a king \( k \) of \( T \) is also a king of \( T[u,v] \): For every \( x \in V \) there is a path of length at most 2 from \( k \) to \( x \) since \( k \) is a king in \( T \). There is an edge \( k \rightarrow u \) by construction and there is the path \( k \rightarrow u \rightarrow v \).

Fourth, if \( x \in V \) is not a king of \( T \), it is also not a king of \( T[u,v] \): Let \( y \in V \) be a vertex at a distance (measured in \( T \)) from \( k \) of at least 3. Then the distance from \( x \) to \( y \) in \( T[u,v] \) is 3, but not less: There is a path \( x \rightarrow u \rightarrow v \rightarrow y \), but neither the path \( x \rightarrow u \rightarrow y \) nor the path \( x \rightarrow v \rightarrow y \) exists.

**Lemma 2.5.** There is no 4-vertex tournament of diameter 2.

**Proof.** For the sake of contradiction, assume that a 4-vertex tournament \( T \) with diameter 2 exists. Clearly, no vertex can have out-degree 0 or 3. Then all four vertices have out-degree 1 or 2. The out-degrees must sum up to 6 as there are six edges, and thus two vertices must have out-degree 1 and the other two have out-degree 2. Let \( u \) be a vertex of out-degree 1 and let \( v \) be the single vertex for which there is an edge \( u \rightarrow v \) in \( T \). We must be able to reach the other two vertices, call them \( z \) and \( w \), within two steps from \( u \), and thus there must be edges from \( v \) to both \( z \) and \( w \). Consider the edge between \( w \) and \( z \). If it goes from \( z \) to \( w \), then the shortest path from \( w \) to \( z \) is \((w,u,v,z)\); if it goes from \( w \) to \( z \), then the shortest path from \( z \) to \( w \) is \((z,u,v,w)\). In either case we have a contradiction.

**Definition 2.6.** We define a 6-vertex tournament \( T_6^{\text{diam}=2} \) as follows. Let \( V = C \cup D \) where \( C = \{c_0,c_1,c_2\} \) and \( D = \{d_0,d_1,d_2\} \). We connect the vertices \( c_i \) in a “clockwise” fashion
and the vertices $d_i$ in a “counter-clockwise” fashion, that is, we add edges $c_0 \rightarrow c_1 \rightarrow c_2 \rightarrow c_0$ and $d_2 \rightarrow d_1 \rightarrow d_0 \rightarrow d_2$. We call $C$ and $D$ the two cycles of the tournament. Next, we add an edge $c_i \rightarrow d_i$ for all $i \in \{0, 1, 2\}$. In the other direction we add an edge $d_i \rightarrow c_{(i+1) \pmod{3}}$ for all $i \in \{0, 1, 2\}$. The remaining missing edges are added in any fixed, arbitrary manner.

**Lemma 2.7.** The tournament $T_{diam=2}^6$ has the following properties:

1. Its diameter is 2.
2. For every vertex $c \in C$ there is a vertex in $d \in D$ such that $d \rightarrow c$.
3. For every vertex $d \in D$ there is a vertex in $c \in C$ such that $c \rightarrow d$.

**Proof.** For the first claim, first note that we can reach all vertices from vertex $c_0$ via the following paths: $c_0 \rightarrow c_1$, $c_0 \rightarrow c_1 \rightarrow c_2$, $c_0 \rightarrow d_0$, $c_0 \rightarrow c_1 \rightarrow d_1$, and $c_0 \rightarrow d_0 \rightarrow d_2$. Second, observe that the situation is symmetric for all other vertices—only the numbering is changed. For the second claim, we can reach $c_i$ from $d_{(i+2) \pmod{3}}$; for the third claim, we can reach $d_i$ from $c_i$. □

**Lemma 2.8.** Let $n$ be a positive integer. Then there exists an $n$-vertex tournament of diameter 2 if and only if $n \notin \{1, 2, 4\}$.

**Proof.** Let $\{x_1, \ldots, x_n\}$ be a set of vertices. We show how to construct a diameter-2 tournament $T_{diam=2}^n$ on this set for $n \notin \{1, 2, 4\}$ and show that no such tournament exists for $n \in \{1, 2, 4\}$.

For $n = 1$, the only tournament contains a single vertex and has diameter 0.

For $n = 2$, the only tournament is a single edge, and thus it has an infinite diameter.

For $n = 3$, the cycle $x_1 \rightarrow x_2 \rightarrow x_3 \rightarrow x_1$ is a tournament of diameter 2.

For $n = 4$, no tournament of diameter 2 exists by Lemma 2.5.

For $n = 5$ and all larger odd numbers, we can construct an $n$-vertex tournament of diameter 2 by repeatedly applying Lemma 2.4 to the 3-vertex tournament of diameter 2. When we apply Lemma 2.4 for the first time, we set $u = x_4$ and $v = x_5$. When we apply it for the second time, we set $u = x_6$ and $v = x_7$. More generally, each time we apply the lemma we set $u$ to be $x_i$ for an even $i$ and we set $v$ to be $x_{i+1}$.

For $n = 6$, the tournament $T_{diam=2}^6$ from Definition 2.6 has diameter 2 by Lemma 2.7.

For $n = 8$ and all larger even numbers, we construct an $n$-vertex tournament of diameter 2 by repeatedly applying Lemma 2.4 to the 6-vertex tournament of diameter 2. This time, each time we apply the lemma we set $u$ to be $x_i$ for an odd $i$ and we set $v$ to be $x_{i+1}$. □

**Definition 2.9.** Given a list $(x_1, \ldots, x_n)$ of vertices for $n \geq 5$, let $T_{diam=2}^n(x_1, \ldots, x_n)$ be the tournament constructed in Lemma 2.8.

When the list is clear from context, we just write $T_{diam=2}^n$. 

10
2.4 The Polynomial Hierarchy

The polynomial hierarchy [MS72, Sto76] can be defined in various ways that yield the same classes. However, for the purpose of this paper, it is easiest to use a quantifier-based definition (see [SM73, Wra76]). We say that a binary relation $R \subseteq \Sigma^* \times \Sigma^*$ is polynomial-time decidable if the set \{ $(x, y)$ | $(x, y) \in R$ \} is in P. We define polynomial-time decidable ternary or quaternary relations similarly. With this definition, we can define the two levels of the polynomial hierarchy that will be of interest to us, namely $\Pi^p_2$ and $\Sigma^p_3$, as follows: A language $L$ is in $\Pi^p_2$ if and only if there exist a polynomial $p$ and a polynomial-time decidable relation $R \subseteq \Sigma^* \times \Sigma^* \times \Sigma^*$ such that for all words $x \in \Sigma^*$ we have

$$x \in L \iff (\forall y \in \Sigma^{p(|x|)})(\exists z \in \Sigma^{p(|x|)})[R(x, y, z)].$$

(2.1)

A language $L$ is in $\Sigma^p_3$ if and only if there exist a polynomial $p$ and a polynomial-time decidable relation $R \subseteq \Sigma^* \times \Sigma^* \times \Sigma^* \times \Sigma^*$ such that for all words $x \in \Sigma^*$ we have

$$x \in L \iff (\exists w \in \Sigma^{p(|x|)})(\forall y \in \Sigma^{p(|x|)})(\exists z \in \Sigma^{p(|x|)})[R(x, w, y, z)].$$

(2.2)

2.5 Succinct Representations of General Graphs

By circuit we refer to combinatorial circuits containing input-, output-, negation-, and-, and or-gates. The fan-in of each gate is at most 2. Fan-out is not restricted. For a circuit $C$ with $n$ input gates and $m$ output gates, we in a slight overloading of notation also use $C$ to denote the function computed by the circuit $C$. This function, $C$, maps elements of $\Sigma^n$ to $\Sigma^m$. For a circuit $C$ we use $\text{code}(C)$ to denote a standard binary encoding of the circuit. The exact details of such a coding will not be important, but note that for $n$-input and $m$-output circuits $C$ the coding will have length at least $n + m$.

We use circuits to define graphs succinctly. For positive integers $n$, given an $2n$-input, 1-output circuit $C$, we say that it specifies the graph $G$ whose vertex set is $V = \Sigma^n$ and whose edge set is defined as follows: There is an edge from $x \in V$ to $y \in V$ if and only if $C(xy) = 1$. We say that $C$ is a succinct representation of $G$. Note that a graph $G$ has many succinct representations.

We next state two lemmas, the second of which will be used repeatedly in our proofs. The first lemma is a standard fact from the literature (it is at the heart of the proof that the circuit value problem is P-complete, see for example [Pap94]) and is used in the proof of the second lemma.

**Lemma 2.10.** For every polynomially time-bounded Turing machine $M$ there exists a logspace computable function $f$ that maps every word $w \in \Sigma^*$ to a circuit $C_{|w|}$ that has $|w|$ input gates and one output gate and satisfies this bi-implication: $C_{|w|}(w) = 1$ if and only if $w \in L(M)$. Note that $C_{|w|}$ depends only on the length of $w$ (and, of course, on $M$).

**Lemma 2.11.** Let $R \subseteq \Sigma^* \times \Sigma^* \times \Sigma^*$ be a ternary relation that is decidable in polynomial time and let $p$ be a polynomial. For every word $x \in \Sigma^*$ let $G_x = (V, E)$ with $V = \Sigma^{p(|x|)}$ and $(y, z) \in E$ if and only if $(x, y, z) \in R$. Then there is a logspace-computable function $f$ that maps every word $x \in \Sigma^*$ to the code of a circuit that specifies the graph $G_x$.  

11
Proof. Given input $x$ we must compute, in polynomial time, the code of a circuit $C$ with the property that, for all $y, z \in \Sigma^p(|x|)$: $C(yz) = 1$ if and only if $(x, y, z) \in R$. The circuit must have $2p(|x|)$ input gates, through which $y$ and $z$ are fed into the circuit, and one output gate.

Let $M$ be a polynomially time-bounded Turing machine that decides the set $\{\langle x, y, z \rangle | (x, y, z) \in R \}$. Then there exists another machine $M'$ that decides the set $\{xyz | x, y, z \in \Sigma^* \}, |y| = |z| = p(|x|), (x, y, z) \in R \}$. Let $f$ be the function from Lemma 2.10 for the machine $M'$. On input $x$, we compute $f(x0^p(|x|)0^p(|y|))$, resulting in a circuit $C'$ with $|x| + 2p(|x|)$ input gates.

The desired circuit $C$ is obtained from $C'$ by a syntactic modification of $C'$. We replace the first $|x|$ input gates by 0- and 1-gates according to the following rule: The $i$th input gate is replaced by a constant gate that outputs the $i$th bit of $x$. Clearly, for any reasonable encoding of circuits, this transformation can be done in logarithmic space.

We claim that resulting circuit $C$, which has $2p(|x|)$ input gates, has the desired properties. To see this, consider $y, z \in \Sigma^p(|x|)$. Then $C(yz) = C'(xyz)$. And this value, $C'(yz)$, is 1 if and only if $(x, y, z) \in R$. □

We next formalize the radius, diameter, and $k$-kingship problems for succinctly specified graphs. Let $k$ be a fixed positive integer.

$\text{succinct-}k\text{-radius} = \{ \text{code}(C) \mid \text{the graph specified by } C \text{ has radius at most } k \}.

\text{succinct-}k\text{-diameter} = \{ \text{code}(C) \mid \text{the graph specified by } C \text{ has diameter at most } k \}.

\text{succinct-}k\text{-king} = \{ \langle \text{code}(C), x \rangle \mid x \text{ is a } k\text{-king in the graph specified by } C \}.

2.6 Tournament Family Specifiers

We can use circuits as introduced in the previous section to describe tournaments succinctly (though of course for a circuit describing a truly random tournament, the term “succinct” would be a quite strained). However, there is also a different, more computationally uniform (and less flexible) way of specifying tournaments succinctly.

A tournament family specifier is a function $f: \Sigma^* \times \Sigma^* \rightarrow \Sigma^*$ such that

1. $f$ is a polynomial-time computable function.

2. $f$ is commutative, that is, for all $x, y \in \Sigma^*$ we have $f(x, y) = f(y, x)$.

3. $f$ is a selector, that is, for all $x, y \in \Sigma^*$ we have $f(x, y) \in \{x, y\}$.

We interpret this as specifying, in the following way, a family of tournaments, one per length. At each length $n$, the vertices in the length-$n$ tournament specified by $f$ will be the bitstrings in $\Sigma^n$. For each two distinct vertices among these, $x$ and $y$, the edge between them will be $x \rightarrow y$ if $f(x, y) = y$ and it will be $y \rightarrow x$ if $f(x, y) = x$. There will be no self-loops. Since our function $f$ always chooses one of its inputs and is commutative, this indeed yields a family of tournaments. We call the tournament just described the length-$n$
tournament induced by \( f \). One may note that the constraints in the definition of tournament family specifiers apply even between strings of different lengths, and yet this is never used in our proofs of results about tournament family specifiers since specifiers specify different, separate tournaments at each length. The reason we require the constraints to hold globally is simply because our motivating notion, commutative P-selectors from P-selectivity theory, has these constraints holding globally. This ensures that every tournament family specifier is a P-selector, but is not needed otherwise.

For completeness, we recall from the introduction the definition of the \( k \)-Kings problem for tournament family specifiers \( f \): \( k \)-Kings\(_f\) = \( \{ x \in \Sigma^* \mid x \text{ is a } k \text{-king in the length-}|x| \text{ tournament specified by } f \} \). A language \( L \) is called a P-\( k \)-king language if there exists a tournament family specifier \( f \) with \( k \)-Kings\(_f\) = \( L \). Our main interest in this paper is finding out which languages are P-\( k \)-king languages. The following lemma provides a first example (we will need the construction from the lemma in later proofs). Note that \( \Sigma^* - \{1,11\} = \text{pad}_j((\Sigma^*) \) holds for all positive integers \( j \).

**Lemma 2.12.** For every \( k \geq 2 \), the language \( \Sigma^* - \{1,11\} \) is a P-\( k \)-king language.

**Proof.** We must show that there is a tournament family specifier such that for all word lengths \( n \), except for \( n = 1 \) and \( n = 2 \), all words in \( \Sigma^n \) are \( k \)-kings of the length-\( n \) tournament induced by \( f \). Note that all vertices of a tournament of length \( n \geq 2 \) are \( k \)-kings if and only if the tournament has diameter 2.

For \( n = 0 \), the tournament family specifier is trivial and, indeed, the single word \( \epsilon \) is a \( k \)-king of the tournament whose vertex set is \( \Sigma^0 \).

For \( n = 1 \), the tournament family specifier directs the arrow from 0 to 1. So 0 and only 0 is a \( k \)-king of the tournament \( \Sigma^1 = \Sigma \).

For \( n = 2 \), the tournament family specifier specifies the following tournament on \( \Sigma^2 = \{00,01,10,11\} \): The three vertices 00, 01, and 10 form a cycle and there is an edge from each of them to 11. Then clearly the three vertices 00, 01, and 10 are \( k \)-kings of \( \Sigma^2 \), and 11 is not.

For \( n = 3 \) and all larger \( n \), the tournament family specifier for the tournament \( \Sigma^n \) specifies a diameter-2 tournament on \( 2^n \) vertices. By Lemma 2.8 such a tournament exists (we have \( 2^n \geq 8 \)). The trickier part is specifying such a tournament in polynomial time. We give a rather detailed construction in the following. Similar constructions later on will be stated with fewer details.

Let us introduce names for the words of \( \Sigma^n \): Let \( \sigma_i \) with \( i \in \{1,2,3,\ldots,2^n\} \) denote the \( i \)th lexicographical word in \( \Sigma^n \). The tournament of diameter 2 on these words will be the tournament \( T_{2^n}^{\text{diam}=2}(\sigma_1,\ldots,\sigma_{2^n}) \) constructed in Lemma 2.8. It remains to show how the edge relation of this tournament can be decided in polynomial time.

The general rule resulting from the construction is the following: Suppose we are given two input words \( \sigma_i \) and \( \sigma_j \) for which our machine \( M \) should decide whether there is an edge from \( \sigma_i \) to \( \sigma_j \) or the other way round. We assume \( i < j \), otherwise we exchange the roles of \( \sigma_i \) and \( \sigma_j \) (and for \( i = j \) nothing needs to be done). If \( i \) and \( j \) are both at most 6, then \( M \) can direct the edge according to the hardwired tournament \( T_6^{\text{diam}=2} \). So suppose \( j > 6 \). In this case we distinguish two cases, depending on whether \( j \) is even or odd.
First, assume that \( j \) is odd. Then \( M \) outputs that there is an edge from \( \sigma_i \) to \( \sigma_j \).

Second, assume that \( j \) is even. If \( i = j - 1 \), then \( M \) outputs that there is an edge from \( \sigma_i \) to \( \sigma_j \). If \( i < j - 1 \), then \( M \) outputs that there is an edge from \( \sigma_j \) to \( \sigma_i \).

Clearly, the computation of \( M \) takes only polynomial time. Furthermore, \( M \) specifies exactly the tournament \( T_{2^n}^{\text{diam}=2}(\sigma_1, \ldots, \sigma_{2^n}) \) as can be seen by comparing the construction of \( T_{2^n}^{\text{diam}=2} \) in Lemma 2.8 and the above specification of \( M \)'s output.

### 3 The Complexity of the Diameter Problem

In this section we prove that the succinct diameter problem is complete for \( \Pi^p_2 \). Indeed, we show that the succinct diameter problem restricted to tournaments, defined by

\[
\text{SUCCINCT-}k\text{-DIAMETER-TOURNAMENT} = \{ \text{code}(C) \mid \text{the graph specified by } C \text{ is a tournament of diameter at most } k \},
\]

is already hard for this class. The tools that we introduce for the proof of this result will be important in the following sections.

**Definition 3.1.** An \( \ell \)-layer tournament is a tournament whose vertex set is the disjoint union of \( \ell \) nonempty sets \( L_1, \ldots, L_{\ell} \) such that the following holds: For any vertex \( u \in L_i \) and \( v \in L_j \) with \( i < j - 1 \), there is an edge \( v \to u \).

**Lemma 3.2.** Let \( T \) be an \( \ell \)-layer tournament, let \( u \in L_1 \), and let \( v \in L_{\ell} \). Then the shortest path from \( u \) to \( v \) has length at least \( \ell - 1 \).

**Proof.** Each edge of a path from \( u \) to \( v \) can increase the level by at most 1. \( \square \)

Our next task is the definition of a rather complex tournament that will be used in later proofs. Recall that we want to show that every problem in \( \Pi^p_2 \) reduces to the succinct diameter problem. For this, we construct a tournament in which all vertices are \( k \)-kings, except possibly for one vertex, which will be a \( k \)-king exactly if a certain “for all . . . exists . . . ” property is true.

For the definition of the tournament and also for the definition of even more complicated tournaments later on, we proceed in three steps. First, we describe the structure of the tournament. This means that we explain how many vertices are present, which names we are going to use to abstractly refer to these vertices, and how these vertices are connected by edges. However, in this first step we do not yet fix which words will later on be used as being associated with these vertices. Rather, the description of the structure of the tournament is given only in terms of the names of the vertices. Second, we prove important properties of the tournament, such as the property that all vertices except possibly for one vertex are \( k \)-kings. Here, we still argue in terms of the names of the vertices of the tournament and are not interested which words are represented by the names. Third, we fix which words we are going to use. That is, for each named vertex we present a unique word that is represented by that name.
**Definition 3.3.** Given integers $n \geq 3$ and $k \geq 2$, a relation $R \subseteq \Sigma^n \times \Sigma^n$, and a set $J$ of even cardinality (called the *junk vertices*), we define a tournament $T^k(R, J)$ as follows.

1. **The layers.**

   The tournament is a $(k + 1)$-layered tournament. The layers get the following special names: Layer $L_1$ is the potential k-king layer. Layers $L_2$, \ldots, $L_{k-1}$ are the antenna layers (note that for $k = 2$ there are no antenna layers). Layer $L_k$ is the $z$-layer. Layer $L_{k+1}$ is the $y$-layer.

2. **Vertices in the layers.**

   The potential $k$-king layer $L_1$ contains a single vertex $p$. Each antenna layer $L_i$ for $i \in \{2, \ldots, k - 1\}$ also contains a single vertex $a_i$.

   The $z$-layer $L_k$ is the set $Z \cup Z'$, where $Z' = \{z_1, z_2, z_3\}$ for even $k$ and $Z' = \{z_1, z_2\}$ for odd $k$. The set $Z = \{\beta_z \mid z \in \Sigma^n\}$ contains $2^n$ elements that are indexed by the bitstrings of length $n$.

   The $y$-layer $L_{k+1}$ is the set $Y \cup C \cup D \cup J$, where $Y = \{\alpha_y \mid y \in \Sigma^n\}$ also contains $2^n$ elements that are indexed by the bitstrings of length $n$ and $C = \{c_0, c_1, c_2\}$ and $D = \{d_0, d_1, d_2\}$.

   We assume that all the sets $Y$, $C$, $D$, $Z$, $Z'$, and $J$ are pairwise disjoint. Note that the total size of the tournament is $k + 8 + 2 \cdot 2^n + |J|$ for even $k$ and $k + 7 + 2 \cdot 2^n + |J|$ for odd $k$.

3. **Edges between adjacent layers.**

   We now describe the connections between one layer and the next (the connections between vertices in nonadjacent layers are already fixed by the fact that $T^k(R, J)$ is a layered tournament).

   For layers $L_1$, \ldots, $L_{k-1}$, each of which contains only one vertex, there is an edge to every vertex in the next layer.

   For layers $L_k$ and $L_{k+1}$, we explain for each element of $Y \cup C \cup D \cup J$ of vertices in the $y$-layer which edges there are to or from vertices in the $z$-layer.

   - (a) Let $\alpha_y \in Y$. For every vertex $\beta_z \in Z$ there is (a) an edge $\beta_z \rightarrow \alpha_y$ if and only if $(y, z) \in R$ and (b) an edge $\alpha_y \rightarrow \beta_z$ if and only if $(y, z) \notin R$. For every vertex $z_i \in Z'$ there is an edge $\alpha_y \rightarrow z_i$.

   - (b) Let $y \in C$. For every $z \in Z$ there is an edge $z \rightarrow y$. There is also an edge $z_1 \rightarrow y$. However, there is an edge $y \rightarrow z_2$ and, provided $z_3$ exists, also an edge $y \rightarrow z_3$.

   - (c) Let $y \in D$. For every $z \in Z$ there is also an edge $z \rightarrow y$. However, there is an edge $y \rightarrow z_1$; and there is an edge $z_2 \rightarrow y$ and, provided $z_3$ exists, also an edge $z_3 \rightarrow y$.

   - (d) Let $y \in J$. For every $z \in Z$ there is an edge $z \rightarrow y$. However, for every $z_i \in Z'$ there is an edge $y \rightarrow z_i$. 
4. Edges inside each layer.

For the potential $k$-king layer and the antenna layers, nothing needs to be specified. For the $z$-layer, number the elements of the $z$-layer such that the first $2^n$ vertices are the elements of $Z$ in their lexicographic order, followed by the vertices $z_1$ and $z_2$ as vertices number $2^n + 1$ and $2^n + 2$. For even $k$, the vertex $z_3$ becomes the vertex numbered $2^n + 3$. For this numbering, we connect the vertices inside the $z$-layer such that they form the tournament $T_{2^n+2}^{\text{diam}=2}$ or the tournament $T_{2^n+3}^{\text{diam}=2}$ from Definition 2.9.

For the $y$-layer, we number the elements of the $y$-layer starting with the $2^n$ vertices of $Y$ in lexicographic order, followed by the vertices of $J$ as vertices numbered $2^n + |J|$ through to $2^n + |J|$. Using this numbering, we connect the vertices as follows.

(a) Connect the vertices in $Y \cup J$ such that they form the tournament $T_{2^n+|J|}^{\text{diam}=2}$ from Lemma 2.8 for the given numbering.

(b) Connect the vertices in $C \cup D$ such that they form the tournament $T_6^{\text{diam}=2}$ from Definition 2.6.

(c) Connect every vertex in $u \in Y \cup J$ and every vertex $v \in C \cup D$ by an edge $v \to u$.

This concludes the definition of $T^k(R, J)$.

Note that in the definition of $T^k(R, J)$ we frequently referred to $n$, where $R \subseteq \Sigma^n \times \Sigma^n$. If $R$ happens to be the empty relation, then $n$ is not specified uniquely, and one would have to write something like $T^k(R, J, n)$ to be precise. To keep the notation simple, we write only $T^k(R, J)$.

We will often need to talk about “an arbitrary vertex in some layer $L_i$.” We will generally use the variables $l_i, l'_i$ and so on to denote such vertices.

Lemma 3.4. Let $k \geq 2$, let $n \geq 3$, let $R \subseteq \Sigma^n \times \Sigma^n$ be a relation, and let $J$ be a set of even size. Then the tournament $T^k(R, J)$ has the following properties:

1. The vertex $p$ in the potential $k$-king layer is a $k$-king if and only if for every $y \in \Sigma^n$ there exists a $z \in \Sigma^n$ such that $(y, z) \in R$ holds.

2. All other vertices are $k$-kings of the tournament.

Proof. We start with the proof of the first claim.

For the first direction, assume that for every $y \in \Sigma^n$ there exists a $z \in \Sigma^n$ such that $(y, z) \in R$ holds. Then $p$ is a $k$-king as can be seen as follows: The construction states that, except for layers $L_k$ and $L_{k+1}$, we can go from every layer in one step to all vertices in the next layer. Thus we can reach all vertices in the $z$-layer $L_k$ within $k - 1$ steps. To reach the vertices in the $y$-layer from the $z$-layer in another step, we use the following edges:

1. Let $\alpha_y \in Y$ with $y \in \Sigma^n$. By assumption there exists a $z \in \Sigma^n$ with $(y, z) \in R$. By part 3a of Definition 3.3.3 there is an edge $\beta_z \to \alpha_y$. 

16
2. Let $c \in C$. Then $z_1 \rightarrow c$ by part 3b.

3. Let $d \in D$. Then $z_2 \rightarrow d$ by part 3c.

4. Let $j \in J$. Then $\beta_z \rightarrow j$ for any $\beta_z \in Z$ by part 3d.

For the second direction, assume that $p$ is a $k$-king. Let $y \in \Sigma^*$ be given. We have to argue that there exists a $z \in \Sigma^*$ with $(y, z) \in R$. Consider a path from $p$ to $\alpha_y$ of length at most $k$. By Lemma 3.2, this path has to end with a step along an edge from some vertex $l_k \in L_{k}$ to $\alpha_y$. This implies $l_x \in Z$ since there is no edge from any vertex in $Z' = L_k - Z$ to $\alpha_y$, see part 3a of Definition 3.3.3. This shows $l_k = \beta_z$ for some $z \in \Sigma^n$ and the presence of the edge $\beta_z \rightarrow \alpha_y$ implies $(y, z) \in R$ as claimed, again by part 3a.

We next prove the second claim by going over all other vertices of $T_k(R,J)$ one by one and showing that they are $k$-kings.

1. Consider the vertex $a_i$ in an antenna layer $L_i$. From $a_i$ we can reach all vertices in the $z$-layer within $k - 2$ steps and all vertices in the antenna layers $L_j$ with $i < j$ in even fewer steps. From the $z$-layer, we can reach all vertices in layers $L_j$ with $j < i$ in one step.

Consider a vertex $l_{k+1}$ in the $y$-layer. We show that we can reach it within two steps from some vertex in the $z$-layer. For $l_{k+1} \in Y \cup J$, we can use the path $z_1 \rightarrow c \rightarrow l_{k+1}$, where $c \in C$ is an arbitrary vertex. For $l_{k+1} \in C$, we can use the path $z_1 \rightarrow l_{k+1}$. Finally, for $l_{k+1} \in D$, we can use the path $z_2 \rightarrow l_{k+1}$.

2. Consider a vertex $l_k$ in the $z$-layer $L_k$. We can reach every other vertex of the $z$-layer from $l_k$ within two steps since the vertices of the $z$-layer induce the tournament $T_{2^{n+2}}$ or $T_{2^{n+3}}$. So it remains to argue that we can also reach the vertices in other layers within $k$ steps.

If $l_k \in Z$, we can reach any vertex $l_{k+1}$ in the $y$-layer by the path $l_k \rightarrow c \rightarrow l_{k+1}$, where $c \in C$ is arbitrary. We can reach the potential $k$-king and all vertices in the antenna layers via the paths $l_k \rightarrow c \rightarrow p$ and $l_k \rightarrow c \rightarrow a_i$.

If $l_k = z_1$, we can reach the vertices of the $y$-layer as follows:

(a) Let $l_{k+1} \in Y \cup J$. Then $l_k \rightarrow c_1 \rightarrow l_{k+1}$ is the desired path.

(b) Let $l_{k+1} \in C$. Then $l_k \rightarrow l_{k+1}$ is the desired path.

(c) Let $l_{k+1} \in D$. By Lemma 2.7 part 3, there exists a $c \in C$ such that $c \rightarrow l_{k+1}$. Then $l_k \rightarrow c \rightarrow l_{k+1}$ is the desired path.

If $l_k \in \{z_2, z_3\}$, we can reach the vertices of the $y$-layer as follows:

(a) Let $l_{k+1} \in Y \cup J$. Then $l_k \rightarrow d_1 \rightarrow l_{k+1}$ is the desired path.

(b) Let $l_{k+1} \in C$. By Lemma 2.7 part 2, there exists a $d \in D$ such that $d \rightarrow l_{k+1}$. Then $l_k \rightarrow d \rightarrow l_{k+1}$ is the desired path.
Definition 3.5. Let $m$ set $\Sigma$ be the size of the tournament $T$ of diameter $2$. There is a path of length at most $2$ from $l \rightarrow y$, see parts 3a and 3d of Definition 3.3.3 for $l \rightarrow z_1$ and part 3b for $z_1 \rightarrow y$. Finally, consider $l'_{k+1} \in D$. Then $l \rightarrow z_2 \rightarrow y$ is the desired path.

(b) Let $l_{k+1} \in C \cup D$. Then all $l'_{k+1} \in Y \cup J$ can be reached in one step. All $l'_{k+1} \in C \cup D$ can be reached within two steps since the vertices in $C \cup D$ form a subtournament of $T$ of diameter $2$.

□

In the definition of the tournament $T^k(R, J)$ we left open which vertices are to be used as vertices $p$ or $a_i$ or $a_y$. We needed only that all these vertices are different. Since our aim is to prove something about succinctly specified tournaments, we will have to use the set $\Sigma^m$ for some $m$ as the set of vertices. This means that we have to use bitstrings for the vertices $p$, $a_i$, and so on. The following definition sets which bitstrings we are going to use.

**Definition 3.5.** Let $k \geq 2$ and $n \geq 3$ be integers, let $R \subseteq \Sigma^n \times \Sigma^n$, and let $m$ be an integer such that $2^m > k + 8 + 2 \cdot 2^n$. We define a tournament $T^k_{\Sigma^m}(R)$ as follows. Its vertex set is $\Sigma^m$. Let $r$ be the size of the tournament $T^k(R, \emptyset)$ from Definition 3.3. Then $r = k + 8 + 2 \cdot 2^n$ when $k$ is even and $r = k + 7 + 2 \cdot 2^n$ when $k$ is odd.

Let $\sigma_i$ denote the lexicographically $i$th element of the set $\Sigma^m$, starting with $i = 1$. Thus $\sigma_1$ is the all-0 bitstring of length $m$, and $\sigma_{2^m}$ is the all-1 bitstrings of length $m$. Let $J = \{\sigma_i \mid r - 5 \leq i \leq 2^m - 6\}$ and note that $J$ has even cardinality.

The tournament $T^k_{\Sigma^m}(R)$ will be the tournament $T^k(R, J)$ with the named vertices assigned to bitstrings in the manner described in the following.

1. The vertex $p$ is mapped to $\sigma_1$.
2. The vertices $a_2, \ldots, a_{k-2}$ in the antenna layers are mapped to $\sigma_2, \ldots, \sigma_{k-2}$, respectively.
3. The vertices in the $z$-layer are mapped to the vertices $\sigma_{k-1}, \ldots, \sigma_{k+2n}$ for even $k$ and to $\sigma_{k-1}, \ldots, \sigma_{k+2n+1}$ for odd $k$. The ordering of the vertices of the $z$-layer is the same as the one described in the Definition 3.3: The vertices $\beta_z \in Z$ come first, in their lexicographic ordering, followed by $z_1, z_2$, and possibly $z_3$. For example, the vertex $\beta_{0^n}$ is mapped to $\sigma_{k-1}$ and $z_2$ is mapped to $\sigma_{k+2n}$.
4. The vertices in the $y$-layer are mapped to the remaining vertices as follows. The vertices $\alpha_y$ are mapped, in lexicographical order, to the vertices $\sigma_{r-2^m-6}$, $\ldots$, $\sigma_{r-6}$. The junk vertices inside the $y$-layer are simply mapped to themselves, namely to $\sigma_{r-5}$ to $\sigma_{2^m-6}$. The vertices in $C \cup D$ are mapped to $\sigma_{2^m-5}$ through $\sigma_{2^m}$. Note that $r - 2^m - 6 = k + 2^m + 1$ for even $k$ and $r - 2^m - 6 = k + 2^m + 2$ for odd $k$.

The above establishes a one-to-one correspondence between the vertices mentioned in the definition of $T^k(R, I)$ and the elements of $\Sigma^m$. This concludes the description of the tournament $T^k_{\Sigma^m}(R)$.

**Theorem 3.6.** Let $k \geq 2$. Then succinct-$k$-diameter-tournament is $\leq_{np}^p$-complete for $\Pi^p_2$.

**Proof.** The problem succinct-$k$-diameter-tournament is clearly a member of $\Pi^p_2$. Given as input a circuit $X$, we must check (a) whether the graph specified by $X$ is a tournament and (b) whether in this graph for all ordered pairs $(s, t)$ of vertices there exists a path of length at most $k$ from $s$ to $t$, that is, whether there exist a list of $k - 1$ (or $k - 2$ or . . . or 0) appropriate intermediate vertices. The first test is a coNP-test, the second is a coNPNP-test.

To prove hardness, let an arbitrary language $L \in \Pi^p_2$ be given. Then there exist a polynomial $p$ and a ternary relation $R \subseteq \Sigma^* \times \Sigma^* \times \Sigma^*$ such that equation (2.1) holds. In the following we describe a polynomial-time reduction from $L$ to succinct-$k$-diameter-tournament.

Let an input $x$ be given. We define a relation $R_x \subseteq \Sigma^p(|x|) \times \Sigma^p(|x|)$ as follows: Let $(y, z) \in R_x$ for $b, c \in \Sigma^p(|x|)$ if and only if $(x, y, z) \in R$. Compute the minimal $m$ such that $2^m > k + 8 + 2 \cdot 2^p(|x|)$. Our aim is to map $x$ to a circuit $X$ that specifies the tournament $T^k_{\Sigma^m}(R_x) = (V, E)$.

Suppose the circuit $X$ does, indeed, specify $T^k_{\Sigma^m}(R_x)$. By Lemma 3.4, all vertices of this tournament are $k$-kings of $T$, except possible for $p = \sigma_1$. This vertex is a $k$-king of $T$ if and only if for all $y \in \Sigma^p(|x|)$ there is a $z \in \Sigma^p(|x|)$ such that $(y, z) \in R_x$. But this means that $p$ is a $k$-king if and only if $x \in L$. Thus $x \in L$ if and only if all vertices of $T^k_{\Sigma^m}(R_x)$ are $k$-kings.

It remains to show that we can construct, in polynomial time, a $2m$-input, 1-output circuit $X$ with the property that for every two vertices $u, v \in V = \Sigma^m$ we have $X(uv) = 1$ if and only if $(u, v) \in E$. To prove this, we employ Lemma 2.11. By this lemma, we can come up with the desired circuit even in logarithmic space, if we can show that there exists a polynomial-time Turing machine $M$ with the following property:

(P1) The machine $M$ gets a tuple $\langle x, u, v \rangle$ as input with $x, u, v \in \Sigma^*$. It accepts this pair if and only if $|u| = |v| = m$, where $m$ is the logarithm of the size of tournament $T^k_{\Sigma^m}(R_x)$, see Definition 3.5, and $(u, v)$ is an edge in the tournament $T^k_{\Sigma^m}(R_x)$.

We now sketch how $M$ works. Suppose an input $w$ is given. The machine $M$ can easily check, see Lemma 2.1, that $w$ is of the form $\langle x, u, v \rangle$ and $M$ can then check whether $|u| = |v| = m$. If any of this is not the case, reject.
Suppose \( u, v \in \Sigma^m \). Let \( u \) be the \( i \)th word in \( \Sigma^m \) in the lexicographical ordering (that is, \( u = \sigma_i \)) and let \( v \) be the \( j \)th word. The task of \( M \) is now to decide whether there is an edge from \( u \) to \( v \) or the other way round.

For this, \( M \) first computes in which level \( u \) and \( v \) lie. All \( M \) has to do is to check in which of the following intervals \( i \) and \( j \) fall (for even \( k \)): \([1,1],[2,2],\ldots,[k-2,k-2],\]

\([k-1,k+2^{|x|}],[k+2^{|x|}+1,2^m] \). For odd \( k \), the last two intervals need to be adjusted. Note that a polynomial-time machine can easily perform this computation.

Our next step is to check how the levels of \( u \) and \( v \) relate. If \( u \) and \( v \) are in different, nonadjacent levels, then \( M \) can output that the edge points from the vertex in the larger level to the smaller one. (To be more precise in this case, \( M \) accepts if and only if \( j < i \).

So assume that \( u \) and \( v \) are either in adjacent layers or in the same layer.

First suppose that \( u \) and \( v \) are in adjacent layers. In this case, \( M \) has to check the conditions described in Definition 3.3. For example, if \( u \) is in the \( z \)-layer and \( v \) is in the \( y \)-layer and \( u \) is the vertex to which a \( \beta_z \) with \( z \in \Sigma^{|x|} \) was mapped (we can easily check whether this is the case and compute \( z \)) and \( v \) is the vertex to which an \( \alpha_y \) with \( y \in \Sigma^{|x|} \) was mapped, then \( M \) must check whether \( (y,z) \in R_x \) holds. By definition, this is the case when \( (x,y,z) \in R \) holds, which can be checked in polynomial time since \( R \) is decidable in polynomial time. To take another example, if \( M \) finds out that \( u \in C \) and \( v \) is in the potential \( k \)-king layer or in one of the antenna layers, then \( M \) will also accept the pair \((u,v)\).

Finally, suppose \( u \) and \( v \) are in the same layer. In this case, \( M \) also needs to correctly discern the direction in which the edge between them is directed. For the \( z \)-layer, the vertices in this layer are connected in such a way that they form the tournament \( T_{|L_k|}^{diam=2} \). The machine \( M \) determines the indices \( i' \) and \( j' \) of, respectively, \( u \) and \( v \) inside their layer, which are just the values of \( i \) and \( j \) minus the index of the beginning of the layer. Then \( M \) must find out whether there is an edge from \( i' \) to \( j' \) in \( T_{|L_k|}^{diam=2} \). We saw already in the proof of Lemma 2.12 how this can be done in polynomial time.

For the \( y \)-layer, the machine finds out for both \( u \) and \( v \) whether they fall inside \( Y \cup J \) or inside \( C \cup D \). In either case, the position of \( u \) and \( v \) inside \( Y \cup J \) or \( C \cup D \) is determined (once more, by subtracting the beginning of \( Y \cup J \) or of \( C \cup D \) from the vertex’s lexicographic index). If \( u, v \in Y \cup J \), we check whether there is an edge in the tournament \( T_{|Y \cup J|}^{diam=2} \) at the corresponding position. Again, this check can be done in polynomial time. If \( u, v \in C \cup D \), we output whether there is an edge in \( T_{|C \cup D|}^{diam=2} \) at the corresponding position. If exactly one of \( u \) and \( v \) is in \( Y \cup J \) and the other is in \( C \cup D \), there is an edge from the vertex in \( C \cup D \) to the other vertex. \( \Box \)

It is easy to see that the proof of Theorem 3.6 also proves the following corollary. (Note that the lower bound does not follow from Theorem 3.6, since determining whether the graph specified by \( X \) is a tournament may not be in P.)

**Corollary 3.7.** Let \( k \geq 2 \). Then \( \text{succinct-}k\text{-diameter} \) is \( \leq_m \text{-complete for } \Pi^p_2 \).

Note that \( \text{succinct-1-diameter} \) is easily seen to be \( \leq_m \text{-complete for coNP} \) and the language \( \text{succinct-1-tournament-diameter} \) is the empty set (since in our model all
graphs specified by a circuit have size at least 2, and so in the case of tournaments, cannot have diameter 1, since no tournament on 2 or more vertices has diameter 1).

4 The Complexity of P-k-King Languages

In this section we establish the following result, which is the main result of this paper.

**Theorem 4.1.** Let $k \geq 2$. Each language in $\Pi^p_2 - \{\emptyset, \Sigma^*\}$ is $\leq_m^p$-equivalent to a P-$k$-king language.

This result says that, excluding from our attention the singleton degrees of the empty set and of $\Sigma^*$, for every $k \geq 2$ every many-one degree can be named by a $k$-kings problem. That is, $k$-kings problems are so flexible that they take on every possible $\Pi^p_2$ complexity level.

Note in particular that the theorem applies to the complete $\Pi^p_2$ degree. Thus we have the following corollary, which shows that the result of [HOZZ06] that P-2-king languages are all in $\Pi^p_2$ is optimal.

**Corollary 4.2.** For each $k \geq 2$, there is a $\Pi^p_2$-complete P-$k$-king language.

We will prove Theorem 4.1 via showing a result, Theorem 4.3, that in effect is even stronger.

**Theorem 4.3.** Let $L$ be a language and let $k \geq 2$. Then $L \in \Pi^p_2$ if and only if there exists a positive integer $j$ such that $\text{pad}^j_2(L)$ is a P-$k$-king language.

Theorem 4.3 says that each set in $\Pi^p_2$ has a padded version of itself that is a P-$k$-king language. By part 5 of Lemma 2.2, this implies Theorem 4.1.

One might naturally ask why it is worth proving Theorem 4.3 rather than merely proving Theorem 4.1. After all, proving Theorem 4.1 directly is slightly simpler; $\text{pad}^j_2$ is not the most intuitively natural padding family in the world; and there are costs to our approach such as the fact that the proof of Theorem 4.3 has much machinery and is somewhat fragile. However, it turns out that Theorem 4.3 is “what we really need” in certain applications: Our proof that the radius problem for succinctly specified graphs is $\Sigma^p_3$-complete relies on the stronger claim of Theorem 4.3. Theorem 4.3 also is worthwhile simply for what it says: It shows that a relatively simple, restricted family of reductions—namely, those of the family $\text{pad}^j_2$—suffices to map each set in $\Pi^p_2 - \{\emptyset, \Sigma^*\}$ to a P-$k$-king language it is equivalent to. Just to make the power of this observation utterly explicit, we mention that, in light of Lemma 2.2 and Theorem 4.3, the $\leq_m^p$-equivalence in Theorem 4.1 and the (implicit)

---

1The case of the degree of the empty set is hopeless, since the only set $\leq_m$-equivalent to the empty set is the empty set, and the empty set can never be a kings problem. The case of the degree of $\Sigma^*$ depends on such things as the alphabet cardinality. However, note that when $\Sigma = \{0, 1\}$, as it does throughout this paper, the two length-1 strings already ensure that no kings set can be $\Sigma^*$, since at least one of those two strings is not a king in the length-1 tournament. So since the only set $\leq_m$-equivalent to $\Sigma^*$ is $\Sigma^*$, this means that the degree of $\Sigma^*$ cannot contain a king set.
\(\leq^p_m\)-completeness in Corollary 4.2 can even be respectively replaced by \(\leq_{fo}\)-equivalence and \(\leq_{fo}\)-completeness.

To prove Theorem 4.3, we introduce new tournaments and, as in the previous section, we do so in three steps: First, we explain how these tournaments are structured, but do not fix which words we are going to use as vertices. Second, we prove that the tournament has desirable properties. Third, we explain which words we are going to use for the vertices.

The core idea behind the definition of the tournaments is to “weave together” multiple \(T^k_{\Sigma^m}(R)\) tournaments.

**Definition 4.4.** Let \(k \geq 2\), \(n \geq 1\), and \(n' \geq 3\) be integers, let \(R \subseteq \Sigma^n \times \Sigma^{n'} \times \Sigma^{n'}\) be a ternary relation, let \(m\) be an integer with \(2^m > k + 8 + 2 \cdot 2^n\), and let \(F\) (called the fill-up vertices) be a set of vertices whose cardinality is not 0, 2, or 4. The woven tournament \(W^k(R, F, m)\) is defined as follows.

1. **The tournaments that are woven together.**
   The vertex set of the woven tournament is the disjoint union of all \(T^k_{\Sigma^m}(R_x)\) for \(x \in \Sigma^n\) and of \(F\). Here, \(R_x \subseteq \Sigma^n \times \Sigma^{n'}\) is the relation that contains all pairs \((y, z)\) such that \((x, y, z) \in R\).
   
   Taking the disjoint union means that for every \(x \in \Sigma^n\) and each \(v \in T^k_{\Sigma^m}(R_x)\), the tournament \(W^k(R, F, m)\) contains a tagged element \(v^x\).
   
   Tagged potential \(k\)-kings get a name for later reference: Let the tagged version of the single vertex of the potential \(k\)-king layer of \(T^k_{\Sigma^m}(R_x)\) be denoted \(p^x\) in the following.

2. **Edges inside the tournaments that are woven together.**
   The vertices inside the woven tournament that come from one \(T^k_{\Sigma^m}(R_x)\) are connected in the woven tournament in the same way as they are connected in \(T^k_{\Sigma^m}(R_x)\); formally, there is an edge \(u \rightarrow v\) in \(T^k_{\Sigma^m}(R_x)\) if and only if there is an edge \(u^x \rightarrow v^x\) in \(W^k(R, F, m)\).

3. **The layers.**
   The woven tournament is a \(k + 1\)-layered tournament. The vertices on layer \(L_i\) of the woven tournament are exactly the vertices in the tournaments \(T^k_{\Sigma^m}(R_x)\) in layer \(L_i\).
   In addition, all vertices from \(F\) are part of layer \(L_k\).

4. **Edges between different, nonadjacent layers.**
   The definition of a layered tournament, Definition 3.1, fixes how vertices in nonadjacent layers are connected.

5. **Edges inside each layer.**
   We next describe how the vertices are connected inside each layer. For two vertices in the same \(T^k_{\Sigma^m}(R_x)\), we have already fixed how they are connected. In the other cases we proceed as follows:
(a) For vertices in layers other than layer $L_k$, we add the missing edges arbitrarily.

(b) For vertices in layer $L_k$, we can also connect vertices in different $T_{\Sigma m}(R_x)$ arbitrarily, but for a vertex $u \in T^k_{\Sigma m}(R_x)$ and a vertex $f \in F$, the edge direction is $f \rightarrow u^x$.

(c) Let $f_1, \ldots, f_{|F|}$ be names for the vertices inside $F$. We connect them such that they form the tournament $T^{|\operatorname{diam}|=2}_{|F|}(f_1, \ldots, f_{|F|})$.

6. Edges between adjacent layers.

Our final task is to describe how vertices in different, but adjacent, layers are connected. Once more, the connection between vertices $u^x$ and $v^x$ stemming from the same $T_{\Sigma m}(R_x)$ is already fixed. For all vertex pairs in adjacent layers for which we have not yet assigned a direction, we use the following rules:

(a) The edges between the vertices in layer $L_{k-1}$ and the vertices in $F$ always point to the vertex in $F$.

(b) Let $l^x_k$ be a vertex in layer $L_k$ and let $l^{x'}_{k+1}$ be a vertex in layer $L_{k+1}$ where $x \neq x'$. Then the edge between $l^x_k$ and $l^{x'}_{k+1}$ is $l^x_k \rightarrow l^{x'}_{k+1}$.

(c) Except when otherwise specified by the two just-stated rules, edges point from the vertex in the layer with the larger index to the vertex in the layer with the smaller index.

This concludes the description of the tournament $W^k(R,F,m)$.

Similarly to the definition of $T^k(R,J)$, the definition of $W^k(R,F,m)$ makes references to the numbers $n$ and $n'$, which are not unique if $R$ happens to be the empty relation. To keep the notation simple, we write $W^k(R,F,m)$ nevertheless.

**Lemma 4.5.** Let $k \geq 2$, $n \geq 1$, and $n' \geq 3$ be integers, let $R \subseteq \Sigma^n \times \Sigma^{n'} \times \Sigma^{n'}$ be a ternary relation, let $m$ be an integer with $2^m > k + 8 + 2 \cdot 2^{n'}$, and let $F$ be a set of vertices whose cardinality is not 0, 2, or 4. Then the woven tournament $W^k(R,F,m)$ has the following properties:

1. For every $x \in \Sigma^n$ the vertex $p^x$ is a $k$-king of the woven tournament if and only if for every $y \in \Sigma^{n'}$ there exists a $z \in \Sigma^{n'}$ such that $(x,y,z) \in R$ holds.

2. All other vertices (vertices other than the $p^x$) are $k$-kings of the woven tournament.

**Proof.** We start with a proof of the first claim.

For the first direction, assume that $p^x$ is a $k$-king of the woven tournament. We have to argue that for every $y \in \Sigma^{n'}$ there exists a $z \in \Sigma^{n'}$ such that $(x,y,z) \in R$. Consider the tournament $T^k_{\Sigma m}(R_x)$. Let $V$ be its vertex set and let $V^x = \{v^x \mid v \in V\}$ be the set of tagged versions of the vertices in $V$. Then $V^x$ is a set of vertices in the woven tournament.

We claim that every path of length at most $k$ from $p^x$ to a vertex $v^x \in V^x$ in the $y$-layer has to fall completely inside $V^x$. Since the woven tournament is a layered tournament, by
Lemma 3.2 we know that a path from \( p^x \) to \( v^x \) has to advance one layer in each step. In particular, this path cannot go back a level or stay inside the same level for one step.

Suppose the path from \( p^x \) to \( v^x \) leaves the set \( V^x \) at some point (and returns to it later on, at the latest when it reaches \( v^x \in V^x \)). When the path leaves \( V^x \), it must do so while advancing a level. By the construction of the woven tournament, this is possible only in the following cases: First, we can go from a vertex in layer \( L \) advancing a level. By the construction of the woven tournament, this is possible only in \( \Sigma \) all vertices on layers other than the \( L \) in \( \Sigma \).

For particular, this path cannot go back a level or stay inside the same level for one step. Lemma 3.2 we know that a path from \( p^x \) to \( v^x \) is a path of length at most \( k \) from the potential \( k \)-king of \( T^k_{\Sigma_m}(R_x) \) to every vertex in the \( y \)-layer of \( T^k_{\Sigma_m}(R_x) \). This implies that the potential \( k \)-king of \( T^k_{\Sigma_m}(R_x) \) is a \( k \)-king of \( T^k_{\Sigma_m}(R_x) \) since all vertices on layers other than the \( y \)-layer are easily reachable even within \( k-1 \) steps from the potential \( k \)-king. By the first part of Lemma 3.4, this implies that for every \( y \in \Sigma_n' \) there exists a \( z \in \Sigma_n' \) such that \( (y, z) \in R_x \), which is equivalent to \( (x, y, z) \in R \).

Let us now prove the second direction of the first claim. Assume that for every \( y \in \Sigma_n' \) there exists a \( z \in \Sigma_n' \) such that \( (x, y, z) \in R \) holds and we wish to show that \( p^x \) is a \( k \)-king of the woven tournament.

By Lemma 3.4, the potential \( k \)-king \( p \) of \( T^k_{\Sigma_m}(R_x) \) is a \( k \)-king of \( T^k_{\Sigma_m}(R_x) \). This implies that there is a path in the woven tournament of length at most \( k \) from \( p^x \) to each vertex \( v^x \) for \( v \in V^x \). Again, \( V\overline{x} \) is the tagged version of the vertex set of \( T^k_{\Sigma_m}(R_x) \).

The harder part is arguing that we can reach all vertices of the woven tournament other than those in \( V^x \) from \( p^x \) within \( k \) steps. For this, we have to consider each type of vertex carefully.

1. Consider any vertex \( l'^x_k \) for \( x' \neq x \) in any layer \( L_i \) for \( i \leq k - 1 \). This vertex can be reached as follows: There is a path of length \( k - 1 \) from \( p^x \) to every vertex in \( V^x \) in layer \( L_k \), and there is an edge from every such vertex to \( l'^x_k \).

2. Consider any vertex in \( F \). This vertex can be reached as follows: There is a path of length \( k - 2 \) from \( p^x \) to the single vertex of \( V^x \) in layer \( L_{k-1} \), and there is a vertex from this vertex to every vertex in \( F \). Thus we can reach every vertex of \( F \) within \( k - 1 \) steps from \( p^x \).

3. Consider any vertex \( l'^x_k \) for \( x' \neq x \) in layer \( L_k \). This vertex can be reached from \( p^x \) via the path of length \( k - 1 \) to some vertex \( f \in F \) (note that \( F \) contains at least one vertex) and the edge from \( f \) to \( l'^x_k \).

4. Consider any vertex \( l'^x_{k+1} \) for \( x' \neq x \) in layer \( L_{k+1} \). This vertex can be reached as follows: There is a path of length \( k - 1 \) from \( p^x \) to each vertex \( l'^x_k \in V^x \) in layer \( L_k \) and, since \( x' \neq x \), there is an edge from \( l'^x_k \) to \( l'^x_{k+1} \).

We next prove the second claim. For this claim we show that all vertices that are not in the potential \( k \)-king layer are always \( k \)-kings of the woven tournament.
1. Let \( a_i^x \) be a vertex in an antenna layer \( L_i \). From \( a_i^x \), we can reach all \( u^x \in V^x \) within \( k \) steps by Lemma 3.4 and the fact that the woven tournament inherits the edges inside \( V^x \) from \( T_{2m}^x(R_x) \). Now consider a vertex \( v \) of the woven tournament outside \( V^x \). Above, we argued that we can reach \( v \) from \( p^x \) within \( k \) steps. Furthermore, the path from \( p^x \) to \( v \) goes through \( a_i^x \). Thus we can also reach \( v \) from \( a_i^x \) within \( k \) steps.

2. Let \( l_k^x \) be a vertex in layer \( L_k \) (but not a vertex from \( F \)).
   (a) Again, we can reach all other vertices in \( V^x \) within \( k \) steps by Lemma 3.4.
   (b) Consider a vertex \( l_i \notin V^x \) in a layer \( L_i \) for \( i < k \). Then there is a direct edge \( l_k^x \rightarrow l_i \).
   (c) Consider a vertex \( l_k^x \) for \( x' \neq x \) in layer \( L_k \). It can be reached within \( k \) steps from \( l_k^x \) as follows: There is an edge form \( l_k^x \) to \( p^{x'} \) and there is a path of length at most \( k - 1 \) from \( p^{x'} \) to \( l_k^x \).
   (d) Consider a vertex \( f \in F \). It can be reached within \( k \) steps from \( l_k^x \) by investing one step to go to any \( p^{x'} \) with \( x' \neq x \) and \( k - 1 \) steps to reach \( f \).
   (e) Every vertex \( l_{k+1}^x \) for \( x' \neq x \) in layer \( L_{k+1} \) can be reached in one step.

3. Let \( f \in F \). From this vertex, we can reach every vertex \( l_k^x \) in layer \( L_k \) in one step. We can reach every vertex \( l_k^x \) in any different layer in two steps as follows: Use one step to go to some \( (l_k')^{x'} \) in layer \( L_k \) with \( x' \neq x \) and use another step to go to \( l_k^x \). Note that this works both for the antenna layers and for the \( y \)-layer. Finally, we can reach all other vertices in \( F \) within two steps since the vertices in \( F \) are connected according to the edge relation of \( T_{\mathsf{diam}=2}^{|F|} \).

4. Consider a vertex \( l_{k+1}^x \) in layer \( L_{k+1} \).
   (a) As before, we can reach all vertices \( v^x \in V^x \) within \( k \) steps by Lemma 3.4.
   (b) We can reach all vertices \( l_k^{x'} \) with \( x' \neq x \) in any layer \( L_i \) for \( i \leq k \) as follows: There is an edge \( l_k^{x'} \rightarrow p^{x'} \) in \( L_i \) and a path of length at most \( k - 1 \) from \( p^{x'} \) to \( l_k^{x'} \).
   (c) All \( f \in F \) can be reached in one step from \( l_{k+1}^x \).
   (d) Let \( (l_{k+1}')^{x'} \) for \( x' \neq x \) be a vertex in layer \( L_{k+1} \). To reach it, we invest one step to reach either the vertex \( z_1^x \) or the vertex \( z_2^x \). By Definition 3.3 part 3 at least one such edge always exists. Then we can reach \( (l_{k+1}')^{x'} \) from either \( z_1^x \) or \( z_2^x \) in one step.

\[ \square \]

Our next step is to fix how the vertices of the tournament \( W^k(R, F, m) \) are coded.

**Definition 4.6.** Let \( k \geq 2 \), \( n \geq 0 \), and \( n' \geq 3 \) be integers, let \( R \subseteq \Sigma^n \times \Sigma^{n'} \times \Sigma^{n'} \) be a ternary relation, let \( m \) be an integer such that \( 2^m > k + 8 + 2 \cdot 2^n' \), and let \( l = n + m + 3 \). The tournament \( W_{\Sigma^l}(R) \) is defined as follows. For \( n \geq 1 \), its vertex set is \( V = \Sigma^l \). The set
\[ F \] is the set of all elements of \( V \) that do not end with 000. Note that this set does not have size 0, 2, or 4. We number the vertices in \( F \) lexicographically. The vertices of the different \( T_{\Sigma_m}^k(R_x) \) are encoded as follows: A vertex \( u^x \) is mapped to the bitstring \( xu000 \). Thus we prefix the vertices of \( T_{\Sigma_n}^k(R_x) \) with \( x \) and add 000 at the end. For \( n = 0 \), we also define a tournament \( W_{\Sigma}^k(R) \), but differently (because \( W^k(R,F) \) is not defined for \( n = 0 \)). We set \( W^k(R,m) \) to be the tournament \( T_{\Sigma_m+3}^k(R_e) \).

**Lemma 4.7.** Let \( k \geq 2, n \geq 0, \) and \( n' \geq 3 \) be integers and let \( R \subseteq \Sigma^n \times \Sigma^{n'} \times \Sigma^{n'} \) be a ternary relation. Then the woven tournament \( W_{\Sigma}^k(R) \) has the following properties:

1. For every \( x \in \Sigma^n \), the bitstring \( x0^{l-n} \) is a \( k \)-king of the woven tournament if and only if for every \( y \in \Sigma^{n'} \) there exists a \( z \in \Sigma^{n'} \) such that \((x,y,z) \in R \) holds.

2. All bitstrings that do not end with \( 0^{l-n} \) are \( k \)-kings of the woven tournament.

**Proof.** For \( n \geq 1 \) observe that the potential \( k \)-kings \( p^x \) inside the woven tournament are exactly the words \( x0^{l-n} \). For \( n = 0 \), note that \( 0^l \) is the potential king of \( T_{\Sigma_m+3}^k(R_e) \) and all other bitstrings are \( k \)-kings of \( T_{\Sigma_m+3}^k(R_e) \).

We are now ready to prove the main result.

**Proof of Theorem 4.3.** We start with the easier direction. Suppose \( \text{pad}_j^k(L) \) is a P-\( k \)-king language. Every P-\( k \)-king language is in \( \Pi_{n+2}^p \) as we have only to check on input \( x \) whether for all \( y \) of the same length there exist an integer \( k' \in \{0, \ldots, k\} \) and bitstrings \( w_0, w_1, \ldots, w_{k'} \) of length \( |x| \) such that \( x = w_0, y = w_{k'} \), and for each \( \ell \in \{0, \ldots, k' - 1\} \), the selector on input \( (w_{\ell}, w_{\ell+1}) \) picks \( w_{\ell+1} \). Next, if \( \text{pad}_j^k(L) \in \Pi_{n+2}^p \), then we clearly also have \( L \in \Pi_{n+2}^p \).

For the other direction, let \( L \in \Pi_{n+2}^p \) be given. By the quantifier characterization of the polynomial hierarchy, see equation (2.1), there exist a polynomial \( p \) and a polynomial-time decidable relation \( R \subseteq \Sigma^* \times \Sigma^* \times \Sigma^* \) such that for all words \( x \in \Sigma^* \) we have

\[
 x \in L \iff (\forall y \in \Sigma^{p(|x|)})(\exists z \in \Sigma^{p(|x|)})[R(x,y,z)].
\]

Choose \( j \geq 3 \) such that the function \( m(n) = n^j + j \) has the property that \( 2^{m(n)} > k + 8 + 2 \cdot 2^{p(n)} \). We claim that \( \text{pad}_j^k(L) \) is a P-\( k \)-king language.

For each nonnegative integer \( n \) we define a relation \( S_n \) as follows: \( S_n \subseteq \Sigma^n \times \Sigma^{p(n)} \times \Sigma^{p(n)} \) and a tuple \((x,y,z) \in \Sigma^n \times \Sigma^{p(n)} \times \Sigma^{p(n)} \) is in \( S_n \) if and only if \((x,y,z) \in R \).

We must now argue that there is a tournament family specifier whose \( k \)-king sets for the different word levels are exactly the words in \( \text{pad}_j^k(L) \). Observe that \( \text{pad}_j^k(L) \subseteq \text{pad}_j^k(\Sigma^*) = \Sigma^* - \{1, 11\} \) and that \( \text{pad}_j^k(L) \) differs from \( \text{pad}_j^k(\Sigma^*) \) only for words of lengths \( l \) of the form \( l = n + n^j + j + 3 \) for some \( n \).

We first explain which tournaments \( T_l \) for \( l \geq 0 \) we are going to use for each word length \( l \). Then we explain how a polynomial-time machine \( M \) can decide the edge relation of these tournaments. We distinguish two different kinds of word length \( l \).
1. For a word length $l$ that is not of the form $l = n + n^2 + j + 3$, let $T_l$ be the tournament specified for the word length $l$ by the selector $f'$ constructed in the proof of Lemma 2.12. In that lemma it is shown that pad$'_j(\Sigma^*)$ is a P-$k$-king language.

2. For a word length $l$ of the form $l = n + n^2 + j + 3$ for some $n$ let $T_l$ be the tournament $W^k_{\Sigma^*}(S_n)$. Note that, indeed, the vertex set of this tournament is $\Sigma^l = \Sigma^{n+m(n)+3}$.

We claim that the $k$-king sets of the $T_l$ form the language pad$'_j(L)$.

1. For word lengths $l$ that are not of the form $n + n^2 + j + 3$ we have pad$'_j(L) \cap \Sigma^l = \text{pad}'_j(\Sigma^*) \cap \Sigma^l$ and the $k$-king set of the tournament $T_l$ is exactly pad$'_j(\Sigma^*) \cap \Sigma^l$.

2. For word lengths $l$ of the form $n + n^2 + j + 3$ we have $T_l = W^k_{\Sigma^*}(S_n)$. By Lemma 4.7, the $k$-king set of $T_l$ contains all bitstrings, except possibly for those that end with $0^{l-n}$. A bitstring $x0^{l-n}$ of this form is in the $k$-king set if and only if $x \in L$. This, in turn, is the case if and only if pad$'_j(x) \in \text{pad}'_j(L)$. This shows that the $k$-king set of $T_l$ is exactly pad$'_j(L) \cap \Sigma^l$.

It remains to argue that we can decide the edge relations of the $T_l$ in polynomial time using a machine $M$. When our input is the two words $x$ and $y$, machine $M$ first compares the lengths of these words and, if they are different, outputs the lexicographically smaller one. If they have the same length $l$, it checks whether $l = n + n^2 + j + 3$ for some $n$. If this is not the case, $M$ outputs the value $f'(x,y)$. Otherwise, the machine proceeds as follows.

Given two words $a, b \in \Sigma^l$ for a word length $l = n + n^2 + j + 3$, the objective of $M$ is to decide in which direction the edge between $a$ and $b$ should go in the tournament $W^k_{\Sigma^*}(S_n)$. For this, it will use the machine constructed in the proof of Theorem 3.6, called $M'$ in the following, which satisfies property (P1):

(P1) The machine $M'$ gets a coded tuple $\langle x, u, v \rangle$ as input with $x, u, v \in \Sigma^*$. It accepts this pair if and only if $|u| = |v| = m(n)$ and $(u,v)$ is an edge in the tournament $T^k_{\Sigma^m(n)}(R_x)$.

For the case $n = 0$, on input $u,v \in \Sigma^*$ the machine $M$ can simply output $M'(\epsilon, u, v)$.

For $n \geq 1$, to determine the direction of the edge between $a$ and $b$, the machine $M$ first finds out for both $a$ and $b$ whether they are in $F$, see Definition 4.6, or whether they are of the form $u^x$ for some vertex $u$ in a tournament $T^k_{\Sigma^m(n)}(R_x)$. This check is easy to perform since we only have to check the last three bits. Depending on the result of the test, we distinguish three cases.

1. Both $a$ and $b$ are in $F$. Then $M$ computes their lexicographic index inside $F$ and outputs the edge direction according to $T^{diam=2}_{[F]}$. This is correct since by Definition 4.4 part 5.5c, the vertices in $F$ are connected in this way.

2. Exactly one of $a$ or $b$ is outside $F$. Without loss of generality assume that this is the case for $a$. Then $a = u^x$ for some vertex $u$ in a tournament $T^k_{\Sigma^m(n)}(R_x)$ and the machine $M$ can determine $x$ by looking at the first $n$ bits of $a$ and it can determine $u$
by looking at the remaining bits of \(a\), except for the last three (which are 000). Then \(M\) determines the level of \(u\) inside \(T_{\Sigma_m(n)}^k(R_x)\). This is done in exactly the same way as \(M'\) does this, see Theorem 3.6. Once the level \(L_i\) of \(u\) has been determined, \(M\) can decide the direction of the edge between \(a\), which is not in \(F\), and \(b\), which is in \(F\). To be in accordance with the different parts of Definition 4.4, \(M\) must output the following:

(a) For \(i < k - 1\), the edge points \(b \rightarrow a\), see part 4.
(b) For \(i = k - 1\), the edge points \(a \rightarrow b\), see part 6.6a.
(c) For \(i = k\), the edge points \(b \rightarrow a\), see part 5.5b.
(d) For \(i = k + 1\), the edge points \(a \rightarrow b\), see part 6.6c.

3. Both \(a\) and \(b\) are outside \(F\). Let \(a = u^x\) and let \(b = v^{x'}\). If \(x = x'\), we have to check whether there is an edge \(u \rightarrow v\) in the tournament \(T_{\Sigma_m(n)}^k(R_x)\), which can be done using \(M'\).

So suppose that \(x \neq x'\). First, \(M\) determines the levels of \(u^x\) and \(v^{x'}\). It then has to output the correct edge direction according to the different parts of Definition 4.4.

(a) The vertices are in different, nonadjacent layers. Then \(M\) outputs that the edge points from the vertex in the layer with the larger index to the vertex with the smaller index, see part 4.

(b) The vertices are in the same layer. By parts 5.5a and 5.5b we can output any direction for the edge; for example, the edge points to the lexicographically smaller vertex.

(c) The vertices are in different, adjacent layers. By part 6, the edge points from the vertex on the layer with the larger index to the vertex with the smaller index. The only exception, see part 6.6b, is for vertices in layers \(L_k\) and \(L_{k+1}\). Here, since \(x \neq x'\), the edge points from the vertex in layer \(L_k\) to the vertex on layer \(L_{k+1}\).

Clearly, all the described computations can be performed in polynomial time, which proves the claim. \(\square\)

5 The Complexity of the Radius Problem

In this section we apply our results on P-\(k\)-king languages to prove that the succinct radius problem for directed graphs is complete for \(\Sigma_3^p\).

**Theorem 5.1.** Let \(k \geq 2\). Then \textsc{succinct-}k-radius is \(\leq_m^p\)-complete for \(\Sigma_3^p\).

**Proof.** First, \textsc{succinct-}k-radius \(\in \Sigma_3^p\) as can be seen as follows: Given as input the code of a \(2n\)-input circuit \(C\) we have to check whether there exists a length-\(n\) bitstring \(x\) such that for all length-\(n\) bitstrings \(y \neq x\) there exists an integer \(\ell \in \{0, \ldots, k - 1\}\) and there
exist \( \ell \) bitstrings \( z_1, \ldots, z_\ell \) of length \( n \) such that in the graph specified by \( C \) the following is a path: \( x \rightarrow z_1 \rightarrow \cdots \rightarrow z_\ell \rightarrow y \) (for \( \ell = 0 \) the path will simply be \( x \rightarrow y \), of course).

Second, let any language \( L \in \Sigma_k^p \) be given. Then there exists a quaternary relation \( R \subseteq \Sigma^* \times \Sigma^* \times \Sigma^* \times \Sigma^* \) and a polynomial \( p \) such that equation (2.2) holds. We define a language \( L' \) as follows:

\[
L' = \{ \langle x, w \rangle \mid |w| = p(|x|) \land (\forall y \in \Sigma^{|x|}) (\exists z \in \Sigma^{p(|x|)}) [R(x, w, y, z)] \}.
\]

Clearly, \( L' \in \Pi^p_3 \). By Theorem 4.3 there exists a number \( j \) such that \( \text{pad}_j(L') \) is a \( \text{P}-\text{k-king} \) language. Let \( f \) be the tournament family specifier of the padded language.

We now describe a reduction from \( L \) to \textsc{succinct-k-radius}. For an input word \( x \in \Sigma^n \), let \( n' = |\langle x, w \rangle| \) for some word \( w \in \Sigma^{|x|} \) (by Lemma 2.1, for a given \( x \) the number \( n' \) does not vary over the possible values of \( w \), since the length of the pairing function’s arguments is enough to set the length of the pairing function’s output). Consider the tournament \( T \) specified by \( f \) for the word length \( m = n' + (n')^3 + j + 3 \).

By the definition of the padding function, we know that \( \langle x, w \rangle \in L' \) holds if and only if \( \text{pad}_j(\langle x, w \rangle) = (x, w)0^{n'+j+3} \) is a \( k \)-king in \( T \).

We define a graph \( G_x = (V, E) \) as follows: Its vertex set is \( \Sigma^{m+1} \). The edges are added in the following manner: First, for a pair of vertices \( (u0, v0) \), there is an edge \( u0 \rightarrow v0 \) in \( G_x \) if and only if there is an edge \( u \rightarrow v \) in \( T \). Second, for each \( w \in \Sigma^{p(|x|)} \) there is an edge from every vertex of the form \( \text{pad}_j(\langle x, w \rangle)0 \) to \( 0^{m+1} \). Third, let \( \sigma_1, \ldots, \sigma_2w \) be the bitstrings of length \( m + 1 \) that end with a 1 in lexicographic order. Then there is an edge from each \( \sigma_i \) to \( \sigma_{i+1} \) for \( i \in \{1, \ldots, k-2\} \). Fourth, there is an edge from \( \sigma_{k-1} \) to all \( \sigma_i \) with \( i > k - 1 \).

We make the following claims about \( G_x \):

1. \( G_x \) has radius \( k \) if and only if \( x \in L \).

2. We can compute, in polynomial time, the code of a circuit \( C \) that specifies \( G_x \).

For the first claim, note that only vertices of the form \( \text{pad}_j(\langle x, w \rangle)0 \) can be \( k \)-kings of \( G_x \). The reason is that there is exactly one path of length \( k - 1 \) leading to the vertex \( \sigma_k \), namely the path \( \sigma_1 \rightarrow \cdots \rightarrow \sigma_k \). The only vertices from which we can reach \( \sigma_1 \) in turn are the vertices \( \text{pad}_j(\langle x, w \rangle)0 \). Thus only these vertices can be \( k \)-kings of \( G_x \). On the other hand, from each of these vertices we can reach all \( \sigma_i \) with \( i < k \) via the path \( \text{pad}_j(\langle x, w \rangle)0 \rightarrow \sigma_1 \rightarrow \sigma_2 \rightarrow \cdots \rightarrow \sigma_i \) and all \( \sigma_i \) with \( i \geq k \) via the path \( \text{pad}_j(\langle x, w \rangle)0 \rightarrow \sigma_1 \rightarrow \sigma_2 \rightarrow \cdots \rightarrow \sigma_{k-1} \rightarrow \sigma_i \). This implies that we can reach all \( \sigma_i \) in \( G_x \) from \( \text{pad}_j(\langle x, w \rangle)0 \) within \( k \) steps if and only if \( \text{pad}_j(\langle x, w \rangle) \) is already a \( k \)-king of \( T \). Thus \( \text{some} \ \text{pad}_j(\langle x, w \rangle)0 \) is a \( k \)-king of \( G_x \) if and only if there exists a \( w \in \Sigma^{p(|x|)} \) with \( \langle x, w \rangle \in L \). This proves the first claim.

For the second claim, we argue in way similar to that used in the proof of Theorem 3.6. As in that proof, it remains to show that we can construct, in polynomial time, a \( 2(m+1) \)-input, 1-output circuit \( C \) with the property that for any two vertices \( x, y \in V = \Sigma^{m+1} \) we have \( C(xy) = 1 \) if and only if \( (x, y) \in E \). Using Lemma 2.11, it suffices to show that there exists a polynomial-time Turing machine \( M \) that takes tuples \( \langle x, y, z \rangle \) as input with
$x, y, z \in \Sigma^*$ and accepts if and only if $|y| = |z| = m + 1$, where $m = n' + (n')^j + j + 3$ and $n' = |(x, w)|$ for some $w \in \Sigma^{|x|}$, and $(y, z)$ is an edge in the graph $G_x$.

We now sketch how $M$ works. Suppose an input $w$ is given. The machine $M$ checks whether $w$ is of the form $⟨x, y, z⟩$ and whether $x$, $y$, and $z$ have the correct length. Let $y = y'b$ and $z = z'c$ with $b, c \in \{0, 1\}$. If $b = c = 0$, then $M$ applies the selector $f$ to $y'$ and $z'$ and accepts if the selector’s output implies an edge $y' \to z'$ in the tournament $T$. The machine $M$ also accepts in the following cases: First, $b = 0$ and $c = 1$ and $z' = 0^m = \sigma_1$ and $y' = \text{pad}_j(⟨x, w⟩)$ for some $w \in \Sigma^{|x|}$. Second, $b = 1$ and $c = 1$ and $y = \sigma_i$ and $z = \sigma_{i+1}$ for $i < k$. Third, $b = 1$ and $c = 1$ and $y = \sigma_{k-1}$ and $z = \sigma_i$ for $i > k$. In all other cases, $M$ rejects.

From the description it should be clear that $M$ does, indeed, accept an input $⟨x, y, z⟩$ if and only if $(y, z)$ is an edge in the graph $G_x$. Since the machine $M$ works in polynomial time, we get the claim. □

6 The Complexity of Top-Toda Languages

In this section we prove that top-Toda languages are in $\Pi^p_2$, but they cannot be NP-complete unless $P = NP$. Recall the definition of the top-Toda languages: Given a commutative P-selector $f$, let $\text{Top-Toda}_f = \{x \in \{0, 1\}^* \mid x \text{ is a } k\text{-king in the length-}|x| \text{ tournament specified by } f \text{ for some } k\}$. Note that $\text{Top-Toda}_f = \bigcup_k k\text{-Kings}_f$.

**Theorem 6.1.** $\text{Top-Toda}_f \in \Pi^p_2$ for all tournament family specifiers $f$.

**Proof.** Suppose we are given an input $x$, which corresponds to a vertex $v$ of the length-|x| tournament $T$ specified by $f$, and must decide whether every vertex of this tournament is reachable from $v$. For this, we can use the observation from [NT05] that there is a $\Pi^p_2$-algorithm for the following problem: The input is a succinct representation of a tournament, a source, and a target vertex. It accepts if and only if there is a path from the source to the target. For our $\Pi^p_2$-algorithm for Top-Toda$_f$ we have to check whether for all vertices $u$ of the tournament $T$ there is a path from $v$ to $u$. For deciding whether there is such a path, we use the $\Pi^p_2$-algorithm. So the overall complexity is $\forall \cdot \Pi^p_2 = \Pi^p_2$ (we use the “∀” operator here in its standard fashion), thus yielding a $\Pi^p_2$-algorithm for the overall problem. □

**Theorem 6.2.** If $\text{Top-Toda}_f$ is $\leq_m$-hard for NP for some tournament family specifier $f$, then $P = NP$.

**Proof.** Suppose we could show that Top-Toda$_f$ is P-selective. Selman noted already [Sel79] that P-selective sets cannot be $\leq_m$-hard for NP unless $P = NP$. At first sight, the set Top-Toda$_f$ appears to be P-selective: Given any two words $x$ and $y$ of the same length, apply $f$ to them. If $f(x, y) = y$ and $y$ is in the top Toda equivalent class, so is $x$ (since every vertex reachable from $y$ is also reachable from $x$ in one more step). Unfortunately, the argument breaks down when the words $x$ and $y$ have different lengths. In this case, $f(x, y) = x$ does not imply that $x$ must be in the top Toda equivalence class of the length-|x| tournament if $y$ is in the top Toda equivalence class of the length-|y| tournament.
Although we cannot apply Selman’s result directly, with a bit of extra effort we can adapt Selman’s proof so that it also works for our situation. Let us call a language \( A \) \textit{lengthwise P-selective} if the following holds: There is a polynomial-time computable function \( f \) such that for every two words \( x \) and \( y \) of the same length we have \( f(x, y) \in \{x, y\} \) and \( \{x, y\} \cap A \neq \emptyset \) implies \( f(x, y) \in A \). Clearly, \( \text{Top-Toda}_f \) is lengthwise P-selective for every tournament specifier \( f \). Thus all that remains to prove is that if a lengthwise P-selective set is \( \leq^p_m \)-hard for \( \text{NP} \), then \( \text{P} = \text{NP} \).

Suppose \( A \) is lengthwise P-selective and suppose we can \( \leq^p_m \)-reduce the satisfiability problem to \( A \) via some reduction machine \( R \). We present a polynomial-time algorithm for the satisfiability problem: On input of a formula \( \phi \), we keep track of a list of formulas such that \( \phi \) is satisfiable if and only if at least one formula in the list is satisfiable. Initially, the list just contains \( \phi \). We apply two operations to the list repeatedly: In an \textit{expansion step}, we simultaneously replace each formula in the list that contains a variable by the two formulas obtained by substituting the variable once by true and once by false. Note that we do not violate the list invariant during the expansion step. After each expansion step we apply \textit{pruning steps}, where we try to find two different formulas \( \rho \) and \( \psi \) in the list such that \( R(\rho) \) and \( R(\psi) \) have the same length. When we find such a pair, we apply the selector to \( R(\rho) \) and \( R(\psi) \). If the selector picks \( R(\rho) \), we remove \( \psi \) from the list. If the selector picks \( R(\psi) \), we remove \( \rho \) from the list. At the end of a pruning step, we move on to another with an expansion step, and so on.

Note that the pruning operation will never remove the only element from the list that is satisfiable. Thus, at the end, when neither expansion nor pruning is possible any more, the original formula \( \phi \) will be satisfiable if and only if there is a true formula in the list.

It remains to argue that the algorithm runs in polynomial time. In each expansion step, the number of variables in every formula decreases by one, so there can only be as many expansion steps as there are variables in \( \phi \). So it suffices to show that the length of the list is never more than polynomial. For this, observe that all elements in the list are most much longer than the original formula \( \phi \). More precisely, there is a polynomial \( p \) such that the length of all list entries is at most \( p(|\phi|) \) (and indeed, in most natural codings of formulas, each formula in the list would be of size less than or equal to the size of \( \phi \)). Next, \( R \) can be time-bounded by some polynomial \( q \). Then all list elements are mapped by \( R \) to words of length at most \( q(p(|\phi|)) \). This means that whenever the length of the list exceeds \( q(p(|\phi|)) \), two different list elements are mapped to words of the same length and one word is pruned. This shows that the length of the list is at most \( 2 \cdot q(p(|\phi|)) \), namely right after an expansion step.

\[ \Box \]

7 Conclusion

In this paper, we saw that king and \( k \)-king problems have tremendous flexibility, and in fact can be used as a naming scheme for the nontrivial \( \Pi^p_2 \) many-one degrees. Using related techniques, we studied the complexity of radius, diameter, and top-Toda problems. In the case of each of the king, \( k \)-king, radius, and diameter problems, we found that completeness
held for classes at the second or third level of the polynomial hierarchy.

References


