A Note on the Complexity of the Reachability Problem for Tournaments

Till Tantau∗
Technische Universität Berlin
Fakultät für Elektrotechnik und Informatik
10623 Berlin, Germany
tantau@cs.tu-berlin.de

October 25, 2001

Abstract

Deciding whether a vertex in a graph is reachable from another vertex has been studied intensively in complexity theory and is well understood. For common types of graphs like directed graphs, undirected graphs, dags or trees it takes a (possibly nondeterministic) logspace machine to decide the reachability problem, and the succinct versions of these problems (which often arise in hardware design) are all PSPACE-complete. In this paper we study tournaments, which are directed graphs with exactly one edge between any two vertices. We show that the tournament reachability problem is first order definable and that its succinct version is \( \Pi^P_2 \)-complete.

Keywords: Descriptive complexity, algorithms, tournaments, reachability, succinct representations.

Introduction

A group of knights have gathered to hold a tournament that consists of a series of jousts between every pair of the knights. After the tournament Sir Lancelot and Sir Galahad meet and Sir Lancelot says, “I liked your style. It is only fair you won our joust.” Sir Galahad answers, “I am not so sure. I think you won a joust against someone who won against someone who won against someone, and so forth, who won against me. Is that true?” The two knights ponder on this, but it seems difficult to answer as there were so many jousts. So they go to Merlin, the magician who moves backwards in time, and pose their problem. Merlin broods on the problem for a while.

∗Work done while visiting the University of Rochester, New York. Supported by the TU Berlin Erwin-Stephan-prize grant.
and finally proclaims: “If some of the jousts in the tournament ended in a
draw, your question is perhaps difficult to answer. But I read a paper in
the far future that presented an extremely efficient algorithm to solve your
problem, if none of them did.” It is this paper Merlin has read.

For the reachability problem we are asked to decide whether there exists a
path from a given source vertex $s$ to a given target vertex $t$ in some graph $G$.
If we restrict the type of graphs for which we try to solve this problem, the
complexity of the problem changes, as the following well known results show:

**Fact 1** ([10, 11]). The reachability problems for directed graphs as well as
for directed acyclic graphs are $\text{NL}$-complete.

**Fact 2** ([13]). The reachability problem for undirected graphs is $\text{SL}$-com-
plete.

**Fact 3** ([3]). The reachability problems for directed forests, undirected for-
tests, directed trees as well as graphs where all nodes have out-degree at most
one are $\text{L}$-complete.

In this paper we study the reachability problem for tournaments [15]
and show that it is first order definable. That means we present a first order
formula $\phi_{\text{TRP}}(s,t)$ that is satisfied by a graph, iff the graph is a tournament
in which $t$ is reachable from $s$. The formula will neither use an ordering
on the universe nor the bit predicate, see [9] for an introduction. A key
ingredient of our proof will be the so-called king lemma, see Fact 9.

Languages whose descriptive complexity is first order are known to be
very simple from a computational point of view. In particular, they are
known [14] to be decidable by a family of circuits of constant depth, but
unbounded fan-in. As it is also known that $\text{L}$-hard sets cannot be first order
definable [1, 6], we conclude the reachability problem for, say, forests is
(unconditionally) harder to solve than the tournament reachability problem.

In hardware design, one is often concerned with graphs that are not given
explicitly via, say, an adjacency list, but only implicitly via a program or
circuit that generates the graph. One way to formalise this is the following:
a succinct representation of a graph $G = (V,E)$ with $V = \{0,1\}^n$ is a
circuit $C_G$ with $2n$ input gates such that $(u,v) \in E$ iff $C_G(uv) = 1$. This
formalisation is due to Galperin and Wigderson [7], but others are also
possible [18, 8]. Succinct representations of graphs allow one to code an
exponentially large graph into a small circuit. This makes the reachability
problems for succinctly coded graphs provably harder.

**Fact 4** ([16, 17, 18]). The reachability problems for succinctly represented
graphs are all $\text{PSPACE}$-complete for the following kinds of graphs: directed
graphs, undirected graphs, dags, directed trees, directed forests, undirected
forests and graphs where each node has out-degree at most one.
We show that the succinct reachability problem for tournaments is $\Pi^P_2$-complete, and is hence presumably easier to solve than for other types of graphs. The proof is based on the ideas used in the construction of the first order description of the ordinary tournament reachability problem, but differs at one crucial point: instead of the king lemma we use the observation that every tournament has a dominating set of logarithmic size. This allows us to trade off the number quantifier alternations against the number of quantified variables.

This paper is organised as follows. In Section 1 we give two algorithms for deciding tournament reachability and prove their correctness. The algorithms are build on two different key properties of tournaments. Interestingly, the first algorithm will only be useful for proving that the tournament reachability problem is first order definable, while the second algorithm will only be useful for proving that the succinct version is in $\Pi^P_2$. In Section 2 we switch from an algorithmic view to first order formulas and show how the first algorithm can be turned into a first order formula. In Section 3 we study the succinct version of the tournament reachability problem and prove its completeness for $\Pi^P_2$, using the second algorithm.

1 Two Algorithms for Tournament Reachability

This section presents two efficient algorithms for deciding tournament reachability. The algorithms are similar, but based on a different key property of tournaments. Roughly spoken, the algorithms trade quantifier alternations against the total number of quantified variables. For first orderness the number of quantified variables must be bound by a constant. Here we use the first algorithm, where this number is minimalised. For showing inclusion of the succinct version in $\Pi^P_2$, we can are not so sensitive about the total number of variables, but rather about the number of alternations. Here we use the second algorithm, which minimises this number.

From now on, “graphs” will always be pairs $G = (V, E)$ consisting of a finite set $V$ of vertices and a edge relation $E \subseteq V \times V$. Instead of $(x, y) \in E$ we will often write $x \rightarrow y$. As usual, we say that a vertex $t$ is reachable from a vertex $s$ if $s = t$, or $s \rightarrow t$, or if there exist vertices $z_1, \ldots, z_k \in V$ with $s \rightarrow v_1 \rightarrow \cdots \rightarrow v_k \rightarrow t$. A graph is strongly connected if every vertex is reachable from every other vertex.

**Definition 5.** A set $I \subseteq V$ is closed in a graph $G = (V, E)$ if for all vertices $v \in I$ all vertices reachable from $v$ are also in $I$.

**Observation 6.** A vertex $t$ is not reachable from a vertex $s$ in a graph $G$, iff there exists a closed set $I \subseteq V$ with $s \in I$ and $t \notin I$.

**Definition 7.** A tournament is a graph $G = (V, E)$ such that between any two vertices there is exactly one edge. It is called strong if it is strongly
connected. The language TRP contains all triples \((G,s,t)\) such that \(t\) is reachable from \(s\) in \(G\). The language STRONG-TOURNAMENT contains all strong tournaments.

Note that by this definition tournaments have self-loops at every vertex. This is not crucial at all, but will simplify our arguments.

**Observation 8.** A set \(I\) is closed in a tournament \(G\) iff for all vertices \(u \in V \setminus I\) and all vertices \(v \in I\) we have \(u \rightarrow v\).

**Key Properties of Tournaments.** The following two facts are key properties of tournaments. Fact 9 was first noticed in the study of animal societies, see [12], where the dominance relations on prides of lions form tournaments, as for each pair of lions one dominates the other.

**Fact 9 (King Lemma [12]).** Let \(G = (V,E)\) be a nonempty tournament. Then there exists a king \(x \in V\). It has the property that for all vertices \(y \in V\) there exists a \(z \in V\) with \(x \rightarrow z \rightarrow y\).

**Fact 10 ([15]).** Let \(G = (V,E)\) be a tournament. Then there exists a dominating set \(D \subseteq V\) of size at most \(\lceil \log |V| \rceil\). It has the property that for all \(y \in V\) there exists an \(x \in D\) with \(x \rightarrow y\).

**Description of the Algorithms.** Algorithms 1 and 2 both solve the tournament reachability problem. The first will be used to show that the problem is first order definable, the second to show that its succinct version is in \(\Pi^P_2\).

**Lemma 11.** Let \(G = (V,E)\) be a tournament and let \(s,t \in V\). Then Algorithms 1 and 2 will output “reachable” on input \((G,s,t)\) iff \(t\) is reachable from \(s\).

**Proof.** First assume that the either algorithm outputs “unreachable”. This answer is correct by Observation 6 as this is output only if there exists a set \(I\) closed in \(G\) with \(s \in I\) and \(t \notin I\).

Next, assume that \(t\) is not reachable from \(s\). We argue that the first algorithm will then output “unreachable”. Consider the set \(I\) of all vertices reachable from \(s\). This is a non-empty set and the graph \(G\) induces a tournament on it. By the King Lemma, there exists a king \(v \in I\) and it holds \(I = I_v = \{ x \in V \mid (\exists z)[v \rightarrow z \rightarrow x]\}\). Clearly, \(I_v\) is closed, \(s \in I_v\) and \(t \notin I_v\). For the second algorithm, consider the set \(I\) once more. This time by Fact 10 there exists a set \(D = \{d_1, \ldots, d_k\}\) such that \(\{ x \in V \mid (\exists i)[d_i \rightarrow x]\}\) is exactly \(I\). Once more, \(I\) is closed, \(s \in I\) and \(t \notin I\). \(\square\)
Algorithm 1 Used to show TRP ∈ FO.

```
input (G, s, t)
if G is no tournament then output “no tournament”; halt
forall v ∈ V do
  let I_v := { x ∈ V | (∃ z)[v → z → x] }  
  if I_v is closed in G and s ∈ I_v and t ∉ I_v then
    output “unreachable”; halt
output “reachable”
```

Algorithm 2 Used to show SUCCINCT-TRP ∈ Π^P_2.

```
input (G, s, t)
if G is no tournament then output “no tournament”; halt
let k := ⌈ log |V| ⌉
forall d_1, ..., d_k ∈ V do
  let I := { x ∈ V | (∃ i)[d_i → x] } 
  if I is closed in G and s ∈ I and t ∉ I then
    output “unreachable”; halt
output “reachable”
```

2 First Orderness of Tournament Reachability

In this section we give a first order formula describing tournament reachability based on Algorithm 1. The existence of this formula implies [14] that TRP can be recognised by a family of AC^0-circuits, i.e., bounded depth, unbounded fan-in circuits. Another consequence is that it can be recognised by an alternating Turing machine making a constant number of alternations in logarithmic time. Finally, it can also be recognised in constant parallel time, see [9].

We will be using the standard vocabulary (also called signature) for graphs, namely τ = 〈E^2〉. The set STRUC[τ] of finite τ-structures, i.e., the structures with vocabulary τ, consist of all tuples G = 〈|G|, E^G〉, where |G| is the finite nonempty universe of G and E^G is a binary relation on |G|.

Clearly, the class STRUC[τ] is exactly the class of finite graphs. We say that a graph property P ⊆ STRUC[τ] is first order definable, if there exists a first order formula φ over the vocabulary τ such that for all G ∈ STRUC[τ] we have G |= φ if G ∈ P.

**Theorem 12.** The language TRP is first order definable. That is, there exists a first order formula φ_{TRP} such that for all graphs G = (V, E) and all vertices s, t ∈ V we have

\[(V, E) |= φ_{TRP}(s, t) \iff ((V, E), s, t) ∈ \text{TRP}.
\]

**Proof.** We give a stepwise construction of φ_{TRP}. Firstly, we show how to
express the property “G is a tournament”.

\[ \phi_{\text{is-tournament}} \equiv (\forall x)[E(x,x)] \land \\
(\forall x,y)[x \neq y \rightarrow (E(x,y) \leftrightarrow \neg E(y,x))]. \]

Next, we define a predicate that “implements” the sets \( I_v \) from Algorithm 1 as follows:

\[ \phi_I(v,x) \equiv (\exists z)[E(v,z) \land E(z,x)]. \]

Clearly, \( \phi_I(v,x) \) holds iff \( x \in I_v = \{ x \in V \mid (\exists z)[v \rightarrow z \rightarrow x] \} \). We next give a formula that holds iff \( I_v \) is closed.

\[ \phi_{\text{closed}}(v) \equiv (\forall x,y)[(\neg \phi_I(v,x) \land \phi_I(v,y)) \rightarrow E(x,y)]. \]

Translated this just says that the set \( I_v \) is closed, if for all pairs \( x,y \in V \) with \( x \not\in I_v \) and \( y \in I_v \) we have \( x \rightarrow y \). By Observation 8 this is just the definition of \( I_v \) being closed in \( G \).

The checks \( s \in I_v \) and \( t \not\in I_v \) can be trivially translated to \( \phi_I(v,s) \) and \( \neg \phi_I(v,t) \). The following predicate will hence be true, iff the main-loop halts (printing “unreachable”).

\[ \phi_{\text{main-loop-halts}}(s,t) \equiv (\exists v)[\phi_{\text{closed}}(v) \land \phi_I(v,s) \land \neg \phi_I(v,t)]. \]

Finally, putting it all together, we arrive at the desired predicate

\[ \phi_{\text{TRP}}(s,t) \equiv \phi_{\text{is-tournament}} \land \neg \phi_{\text{main-loop-halts}}(s,t). \]

\[ \square \]

**Corollary 13.** **STRONG-TOURNAMENT is first order definable.**

For comparison with the results of the next section, it will be crucial to know how many quantifier alternations there are in the formula \( \phi_{\text{TRP}} \). We can easily find this out by expanding \( \phi_{\text{TRP}} \), which yields the formula

\[ (\forall x,y)[E(x,x) \land (x = y \lor (E(x,y) \leftrightarrow \neg E(y,x)))] \land \\
(\forall v)(\exists x,y)[(\forall z)[\neg E(v,z) \lor \neg E(z,x)] \land \\
(\exists z)[E(v,z) \land E(z,y)] \land \neg E(x,y)] \lor \\
(\forall z)[\neg E(v,z) \lor \neg E(z,s)] \lor (\exists z)[E(v,z) \land E(z,t)]. \]

The expanded formula uses three variables apart from \( s \) and \( t \), namely \( x, y \) and \( z \). Note that this number reflects the amount of space needed by a deterministic logspace Turing machine to decide \( \text{TRP} \). The alternating quantifier depth of the formula is three, starting with a universal quantifier.
3 Succinct Tournament Reachability

In this section we study succinctly represented graphs. This study is motivated by the fact that some very large graphs arising in practice, like the graphs of integrated circuits of modern chips, are highly organised. Such graphs are often not given explicitly but rather implicitly via a description in some hardware description language. It is of interest to know whether there exist efficient algorithms for checking, say, planarity of graphs given in such a succinct way. Unfortunately, it is known [16, 18] that most interesting problems for succinctly represented graphs are PSPACE-complete or even NEXP-complete, and in [7] it is shown that even such trivial properties as “G has an edge” are NP-complete for succinctly represented graphs.

**Definition 14.** A succinct representation of a graph $G = (V,E)$ with $V = \{0,1\}^n$ is a circuit $C_G$ with $2^n$ input gates and one output gate, such that for all $u,v \in V$ we have $(u,v) \in E$ iff $C_G(uv) = 1$.

The idea is that the circuit will tell us for any two vertices of the graph, whether there is a directed edge between them or not. We could also encode graphs by Boolean formulas rather than circuits, but it is easily seen that this gives the same completeness results. Totally different encodings are also possible, see [18] for an overview, but we concentrate on circuits.

**Definition 15.** The problem SUCCINCT-TRP consists of all triples $(C,s,t)$, where $C$ is a succinct representation of a tournament $G = (V,E)$ in which $t \in V$ is reachable from $s \in V$.

Having a look at the expanded formula for $\phi_{\text{TRP}}$, it is easily seen that SUCCINCT-TRP is in $\Pi^P_3$. The following theorem shows that the problem is actually even in $\Pi^P_2$ and also is hard for that class.

**Theorem 16.** SUCCINT-TRP is $\Pi^P_2$-complete.

**Proof.** We first show SUCCINT-TRP $\in \Pi^P_2$. Let an input $(C,s,t)$ be given and let $C$ represent a graph $G = (V,E)$. Note that $k = \log |V| = n$. As it is a coNP-complete problem to decide whether $G$ is a tournament, we can easily check this first.

We next check whether Algorithm 2 will output “reachable”. This is the case if the main loop never reaches the inner halt statement. Spelled out this means, that for all $d_1,\ldots,d_n \in \{0,1\}^n$ either $s \not\in I$ or $t \in I$ or $I$ is not closed, i.e., that there exist vertices $u \in I$ and $v \in V \setminus I$ such that $u \rightarrow v$. As testing whether some vertex is in $I = \{x \in V \mid d_1 \rightarrow x \lor \cdots \lor d_n \rightarrow x\}$ can be done in polynomial time, we get the desired $\Pi^P_2$ algorithm.

We now prove hardness. Let $L \in \Pi^P_2$ be any language. Then by the quantifier characterisation of the polynomial hierarchy [19] there exists a polynomial time decidable ternary relation $R$ and a constant $c$ such that

$$L = \{ x \mid (\forall y, |y| \leq |x|^c)(\exists z, |z| \leq |x|^c)(x,y,z) \in R \}.$$
For somewhat technical reasons, see below, if necessary we modify the relation \( R \) such that for \( y_0 = 1^{\lfloor x \rfloor} \) there always exists a witness \( z \) with \((x, y_0, z) \in R\). We can hence ignore \( y_0 \) in the following. We now reduce \( L \) to \textsc{succinct-trp}. Let \( x \) with \( n := |x| \) be an input string. We must construct a circuit \( C \) and two bitstrings \( s, t \) such that \( x \in L \) iff \((C, s, t) \in \textsc{succinct-trp}\).

The rough idea is as follows. We construct a tournament of exponential size, which is highly structured and can hence be described by a small circuit, namely \( C \). The tournament consists of \( 2^n \) many levels. The \( 2^n - 1 \) many transitions from one level to the next correspond to the different \( y \)'s (and we have made sure there are only \( 2^n - 1 \) many interesting \( y \)'s). The source is any vertex on the first (bottom) level, the target is any vertex on the last (top) level. On each level there are \( 2^n \) many vertices, which correspond to the different \( z \)'s and which are connected in such a way that there is always a path between any two vertices on the same level.

The edges between different levels are generally pointed “downwards”, i.e., from higher levels down to lower levels. The only exception are edges between adjacent levels \( y \) and \( y' \). These edges generally also point downwards, except if the edge is between two vertices corresponding to the same \( z \). In this case the edge points “upwards” if \((x, y, z) \in R\).

We now make this construction more precise. Let \( \ell := n^c \). Our vertex set will be \( V = \{0,1\}^{2\ell} \). Every vertex \( v \in V \) can be split into two parts \( y \in \{0,1\}^\ell \) and \( z \in \{0,1\}^\ell \) with \( yz = v \). For any two distinct vertices \( v = yz \) and \( v' = y'z' \) we put an edge from \( v \) to \( v' \) into the edge set \( E \) if one of the following conditions holds, and an edge from \( v' \) to \( v \) if none of them hold:

1. We have \( y > y' + 1 \).
2. We have \( y = y' + 1 \) and \( z \neq z' \).
3. We have \( y = y' + 1 \) and \( z = z' \) and \((x, y, z) \notin R\).
4. We have \( y = y' \) and \( z > z' + 1 \).

Let \( s := 0^{2\ell} \) and \( t := 1^{2\ell} \). Clearly, there exists a circuit \( C \) with \( 4\ell \) many input gates that evaluates to 1 iff \((v, v') \in E\), as we can use Cook’s construction [2] to turn the predicate \( R \) into a polynomially sized circuit.

Note that for all \( y \) and all pairs \( z, z' \) the vertex \( yz' \) is always reachable from \( yz \) as the vertices on each level form a great “circle”.

From each level \( y \) one can go (at best) only one level higher to the next level \( y' \) as all edges between levels far apart point downwards. To get even one level higher from \( y \) to \( y' \), there must exists a \( z \) such that \((yz, y'z) \in E\). This in turn means \((x, y, z) \in R\). So in order to get from the source \( s \) on the bottom level to the target \( t \) at the top level, for all \( y \) there must exist a string \( z \) such that \((x, y, z) \in R\).

**Corollary 17.** \textsc{succinct-strong-tournament} is \( \Pi_2^P \)-complete.

**Proof.** For the hardness just note that \( s \) is trivially reachable from \( t \).
Conclusion

We have shown that the descriptive complexity of the tournament reachability problem is low. This problem can be described by a first order formula that uses three variables and has three levels of quantifier alternation. Thus the tournament reachability problem has $\text{AC}^0$-circuits. As a corollary we obtain that the problem can also be decided in logarithmic space. As the class of first order properties is a proper subset of the class $\text{L}$, the tournament reachability problem is provably simpler to solve than $\text{L}$-hard problems like tree, forest or dag reachability.

We also showed that the succinct version of the tournament reachability problem has a presumably lower complexity than the succinct version of most other reachability problems. Succinct tournament reachability is $\text{II}_{\text{P}}^2$-complete. The same is true for the succinct strong tournament problem.

The proofs were based on two different algorithms that exploited different key properties of tournaments. While the first algorithm is not useful in the succinct setting as it has one quantifier alternation too much, the second algorithm is not useful in the first order setting as it quantifies over a non-constant number of vertices.

A natural question is arises. Is the first order formula for TRP optimal with respect to its quantifier alternation depth? As we do not need ordering in our first order formula, it seems promising to use an Ehrenfeucht-Fraissé game [4, 5] to show that three levels of quantifier alternations are necessary.

Acknowledgments

I would like to thank Ken Regan, Alan Selman and Leen Torenvliet for helpful discussions.

References


