Hardness Results for Tournament Isomorphism and Automorphism

Fabian Wagner *
Institut für Theoretische Informatik,
Universität Ulm, 89073 Ulm, Germany
fabian.wagner@uni-ulm.de

September 3, 2007

Abstract

A tournament is a graph in which each pair of distinct vertices is connected by exactly one directed edge. Tournaments are an important graph class, for which isomorphism testing seems to be easier to compute than for the isomorphism problem of general graphs. We show that tournament isomorphism and tournament automorphism is hard under DLOGTIME uniform AC^0 many-one reductions for the complexity classes NL, $C_=L$, PL (probabilistic logarithmic space), for logarithmic space modular counting classes MOD_kL with odd $k \geq 3$ and for DET, the class of problems, NC^1 reducible to the determinant. These lower bounds have been proven for graph isomorphism, see [21].

1 Introduction

The graph isomorphism problem (GI) consists in determining whether there is a bijection between the vertices of two graphs, preserving the edge-relations. Until today, it is open whether GI is contained in P or complete for NP. A proof of the NP-completeness for GI would cause a collapse of the polynomial time hierarchy to its second level, see [7],[20]. Concerning lower bounds, DET $\leq_m^{AC^0}$ GI [21].

For many graph classes, polynomial time algorithms for isomorphism testing are known, e.g. for graphs of bounded degree [16], or planar graphs [13]. Even fast parallel algorithms for isomorphism testing have been developed, e.g. for planar graphs [18], trees [15], [9] or graphs with bounded color-class size [17].

*Supported by DFG grant TO 200/2-1

A tournament is a directed graph with exactly one arc between every pair of distinct vertices. Tournaments comprise a large and important class of directed graphs and can be found in many applications, see e.g. [12]. The tournament isomorphism problem (TI) is GI restricted to tournaments. The best known algorithm for TI takes $n^{log(n)}$ time [6] and for GI takes $\exp(\sqrt{cn\log(n)})$ time (Luks, Zemlyachenko, cf. [6]). Arvind et al. [1] reduced TI onto Mod_2GA which is an intermediate problem between GA and GI and contains the class of graphs with an even number of automorphisms. This follows, because the automorphism group of any tournament is of odd size [14], which in turn implies, that two tournaments are isomorphic, iff the automorphism group of their disjoint union contains an order two permutation (which must switch both graphs). Thus TI seems to be an easier problem than GI. Since the relation between GI and TI is not clear, we contribute to analyze the complexity status of tournament isomorphism. We show that TI and the tournament automorphism problem (TA) are hard for NC^1, L, NL, $\mathrm{Mod_kL}$ with $k \geq 3$ odd integer, #L and DET under AC^0 many-one reductions. For proving that GI is hard for $\mathrm{Mod_kL}$, we need a graph gadget with subgraphs, having orbits of size k to encode an integer in $\mathbb{Z}_k$. Since the order of automorphism groups of tournaments always are *odd* [14], we cannot directly encode an integer in $\mathbb{Z}_k$ with even $k \geq 2$. In order to encode Boolean values, we need another graph gadget. We encode value 0 as the identical mapping and value 1 as the switching of two subgraphs. which again leads to orbits of even size.

Since TA $\leq_m^{\mathrm{AC}^0}$ TI (see Corollary 2.1, TA is prefix-TA without prefix) and the converse direction is unknown, proving DET $\leq_m^{\mathrm{AC}^0}$ TA is a stronger result.

Due to space reasons some proofs are missing and would be included in the final version. We refer to the authors home page for the complete proofs.

The layout of this paper is as follows: In Section 2 we denote complexity classes and graph isomorphism problems. Section 3 contains hardness results for TI, that is DET $\leq_m^{\mathrm{AC}^0}$ TI. In Section 4, we prove these results for TA.

2 Preliminaries

Complexity classes. We assume familiarity with basic notions of complexity theory such as can be found in standard textbooks. NL is the class of languages accepted by nondeterministic Turing machines using a work tape bounded by logarithmic space. #L [4] is the class of functions $f : \Sigma^* \to \mathbb{N}$ that counts the number of accepting paths of a NL machine on a input. The complexity classes PL (probabilistic logspace, [11] [19]), $\mathrm{C_=L}$ (exact counting in logspace, [3]), and $\mathrm{Mod_kL}$ (modular counting in log space, $k \geq 2$, [8]) can be defined in terms of #L functions:

$$\mathrm{PL} = \{A : \exists p \in Poly, f \in \#\,\mathrm{L}\ ,\ \ \forall x \in \Sigma^*\ x \in A \Leftrightarrow f(x) \geq 2^{p(|x|)}\}$$

$$\mathrm{C_{=}L} = \{A : \exists p \in Poly, f \in \#\,\mathrm{L}\ ,\ \ \forall x \in \Sigma^*\ x \in A \Leftrightarrow f(x) = 2^{p(|x|)}\}$$
$$\mathrm{MOD_kL} = \{A : \exists f \in \#\,\mathrm{L}\ ,\ \ \forall x \in \Sigma^*\ x \in A \Leftrightarrow f(x) \equiv 1 \mod k\}$$

$\mathrm{MOD_k}$ circuits ($k \geq 2$) are circuits with input variables over $\mathbb{Z}_k$ and gates computing addition in $\mathbb{Z}_k$. The evaluation problem for such circuits (given fixed values for the inputs, testing if the output is 1) is complete for $\mathrm{MOD_kL}$ under $\mathrm{AC^0}$ many-one reductions.

DET (also denoted $\mathrm{NC^1}(\#\mathrm{L})$) is the class of functions $\mathrm{NC^1}$ Turing reducible to the determinant. That is the class of problems, solvable by $\mathrm{NC^1}$ circuits with additional oracle gates for computing the determinant of an integer matrix.

The known relations among the considered classes are: $\mathrm{MOD_kL} \subseteq$ DET and NL $\subseteq \mathrm{C_{=}L} \subseteq$ PL $\subseteq$ DET. Therefore, the hardness of GI for DET implies hardness with respect to the other classes. We denote $\mathrm{AC^0}$ many-one reductions by $\leq_m^{\mathrm{AC^0}}$ and logspace Turing reductions by $\leq_T^L$.

Graph Isomorphism Problems. Let $G = (V, E)$ be a *graph* with a set of *vertices* $V = V(G)$ and *edges* $E = E(G)$. A *directed* edge or *arc* is denoted (v_1, v_2) and an *undirected* edge $\{v_1, v_2\}$. $G[X]$ is a subgraph of G *induced* on vertex set X. Let H be a subgraph of G then $G \backslash H = G[V(G) \backslash V(H)]$. Let $E' \subseteq E(G)$ then $G \backslash E' = (V(G), E(G) \backslash E')$.

For shorter notations we write $[k, n] = \{k, \ldots, n\}$ for integers $k < n$ and $[n]$ if $k = 1$. Let $\oplus$ denote the modulo addition in $\mathbb{Z}_n$. The set $Sym(V)$ is the *symmetric group* over a set V and $S_n = Sym([n])$.

An *automorphism* of graph G is a permutation $\phi : V(G) \mapsto V(G)$ preserving adjacency: $(u, v) \in E(G) \Leftrightarrow (\phi(u), \phi(v)) \in E(G)$. The automorphisms except the identity are called *nontrivial*. A *rigid* graph contains no nontrivial automorphisms. The *graph automorphism problem* (GA) decides, whether a graph is rigid or not. The *automorphism group* $Aut(G)$ is the set of automorphisms of G.

An *isomorphism* between graphs G and H is a bijective mapping of vertices in G onto vertices in H that preserves adjacency. Both graphs are *isomorphic* if such an isomorphism exists. The *graph isomorphism problem* (GI) is the problem of deciding, whether two given graphs G, H are isomorphic, write $G \cong H$. Further, define the tuple of graph pairs $PGI = \{\, ((G, H)(I, J)) \mid G \cong H \Leftrightarrow I \ncong J \,\}$ with the promise that exactly one pair is isomorphic [21].

Let H be a subgraph of G. Define $Aut_G(H) \subseteq Aut(H)$ as the set of automorphisms, which can be extended to an automorphism in $Aut(G)$. An automorphism $\phi \in Sym(V)$ *acts cyclically* on a vertex set $V = \{v_0, \ldots, v_{n-1}\}$, if there exists $a \in [0, n-1]$ such that $\phi(v_i) = v_{i \oplus a}$ for all $i \in [0, n-1]$. We further say $\phi \in Sym(V(G))$ acts cyclically on subgraphs $G[V_0], \ldots, G[V_{k-1}]$, if there exists $a \in [0, k-1]$ such that $\phi(v) \in V_{i \oplus a}$ for all $v \in V_i$ and $i \in [0, k-1]$.

Let $S_1, \ldots, S_k \subseteq V(G)$ be a set of distinct vertices of graph G. The set of automorphisms, mapping for all $i \in [k]$ vertices in S_i onto vertices in S_i in any order, are called *setwise stabilizer* of $S_1, \ldots, S_k$. $G_{[S_1,\ldots,S_k]}$ denotes graph G with $S_1, \ldots, S_k$ *setwise stabilized* in automorphism group of G.

Let k be a fixed integer. A *coloring* of a graph G is a function $f : V(G) \mapsto [k]$. For any isomorphism between colored graphs, the color relations have to be preserved. The decision problem is called the *isomorphism problem for colored graphs* (color-GI). Observe that color-GI $\leq_m^{\mathrm{AC}^0}$ GI [14].

Let $\{x_1, \ldots, x_k\}, \{y_1, \ldots, y_k\} \subseteq V(G)$ be vertex sets of graph G. The *prefix automorphism problem* (prefix-GA) as denoted in [14] is to find an automorphism $\phi \in Aut(G)$ such that $\phi(x_i) = y_i \ \forall i \in [k]$. Observe that prefix-GA $\leq_m^{\mathrm{AC}^0}$ GI [21]. If the vertex sets are given as mapping ϕ with $\phi(x_i) = y_i, i \in [k]$ then (G, ϕ) is an instance for prefix-GA.

A *tournament* is a directed graph with one arc between each pair of distinct vertices. A *cyclic tournament* T is a tournament on n vertices $x_0, \ldots, x_{n-1}$ such that $(x_i, x_{i\oplus j}) \in E(T)$ for all $i \in [0, n-1], j \in [1, \lfloor \frac{n}{2} \rfloor]$. The *tournament isomorphism problem* (TI) is the same as GI, if the input-graphs are tournaments. Then we also write color-TI, TA, prefix-TA instead of color-GI, GA, prefix-GA. Arvind et.al. [2] showed that color-TI is polynomial-time many-one reducible to TI without coloring. By verifying the proof, we observe that it is an AC^0 many-one reduction. Adapting the proof of prefix-GA $\leq_m^{\mathrm{AC}^0}$ GI [21], we obtain the following chain of reductions. Thus we prove lower bounds for prefix-TA.

Corollary 2.1 *prefix-TA* $\leq_m^{\mathrm{AC}^0}$ *color-TI* $\leq_m^{\mathrm{AC}^0}$ *TI.*

In some reductions, we construct graph gadgets $G(C)$ for simulating the evaluation of circuits C. By $G(C_i)$ we denote the simulation of a gate C_i of circuit C with index i. We also denote this way vertex sets or vertices, e.g. $v(C) \in U(C) \subseteq V(G(C))$. If the context is clear then (C) will be omitted.

3 Hardness Results for Tournament Isomorphism

In this section we show that TI is hard for some complexity classes under AC^0 many-one reductions.

3.1 Hardness Results of TI for Modular Counting Classes

GI is hard for the logarithmic space modular counting classes $\mathrm{Mod_k L}$ for all $k \geq 2$ [21]. Since the circuit value problem restricted to modulo addition gates over $\mathbb{Z}_k$ for $k \geq 2$ is complete for $\mathrm{Mod_k L}$ and with Corollary 2.1, we prove that TI is hard for $\mathrm{Mod_k L}$ with odd $k \geq 3$.

With the following graph gadget, we can simulate a modulo addition circuit gate.

Definition 3.1 [21] *Fix $k \geq 2$. The* modulo addition graph gadget G^k *is defined as*

$$V(G^k) = \{\, x_a, y_a, z_a, u_{a,b} \mid a, b \in [0, k-1] \,\},$$

$$E(G^k) = \{\, \{x_a, u_{a,b}\}, \{y_b, u_{a,b}\}, \{u_{a,b}, z_{a\oplus b}\} \mid a, b \in [0, k-1] \,\}.$$

Let $U, X, Y, Z \subseteq V$ be vertex sets containing all the $k^2 + 3k$ vertices denoted by indexed lower case letters each. Denote vertex sets X as *left input-*, Y as *right input-* and Z as *output-vertices*. Figure 1 shows an example for $k = 3$.

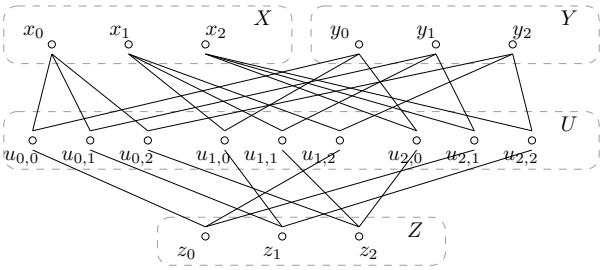

Figure 1: Modulo addition graph gadget G^3

Lemma 3.2 describes automorphism properties of this graph gadget.

Lemma 3.2 [21] *Fix $k \geq 2$. Then for any $a, b \in [0, k-1]$ there is a unique automorphism $\phi_{ab} \in Aut(G^k)$ with $\phi_{ab}(x_i) = x_{a\oplus i}$ and $\phi_{ab}(y_i) = y_{b\oplus i}$ for $i \in [0, k-1]$ and with $\phi_{ab}(u_i) = u_{a\oplus i, b\oplus j}$ and $\phi_{ab}(z_i) = z_{a\oplus b\oplus i}$.*

With this graph gadget, we construct a graph $G(C)$, simulating a circuit C of wired modulo addition gadgets.

Definition 3.3 [21] *Let C be a circuit of m modulo addition gates $C_1, \ldots, C_m$ and $k \geq 2$. Define $G(C)$ with*

$$V(G(C)) = \bigcup_{p\in[m]} V(G^k(C_p)),$$

$$E(G(C)) = \bigcup_{p\in[m]} E(G^k(C_p)) \cup \bigcup_{p,q\in[m],p<q} E_{p,q} \text{ with}$$

$$E_{p,q} = \begin{cases} \{z_i(C_p), x_i(C_q)\} & \text{if } C_p \text{ and left input of } C_q \text{ are wired,} \\ \{z_i(C_p), y_i(C_q)\} & \text{if } C_p \text{ and right input of } C_q \text{ are wired,} \\ \emptyset & \text{if } C_p, C_q \text{ are not wired directly.} \end{cases}$$

Furthermore, color vertices in $U(C_j), X(C_j), Y(C_j), Z(C_j)$ with colors (u, j), (x, j), (y, j), (z, j) in this order for all $j \in [m]$.

The reduction of this decision problem for circuit C to prefix-GA is as follows: compute a graph $G(C)$ and define prefixes for input- and output values. Thus $\text{MOD}_\text{k}\text{L} \leq_m^{\text{AC}^0}$ GI. We will show, how far hardness for $\text{MOD}_\text{k}\text{L}$ also holds for TI.

Theorem 3.4 $\text{MOD}_k\text{L} \leq_m^{\text{AC}^0}$ *TI with* $k \geq 3$ *an odd integer.*

The main proof idea is to transform the graph gadgets G^k and the circuit $G(C)$ (containing graph gadgets G^k as subgraphs) into tournaments. For this task, we first describe how graphs can be modified without changing the automorphism group.

Lemma 3.5 *Let H be an induced subgraph of a graph G with $Aut(G)$ setwise stabilizing vertices in H. Let H' be a graph with $V(H') = V(H)$ and let G' be G after replacing the induced subgraph H by subgraph H', setwise stabilizing vertices in H'. If $Aut_G(H) \subseteq Aut(H') \subseteq Aut(H)$ then $Aut(G) = Aut(G')$.*

Proof. In this prove, we assume familiarity with the prefix-GA problem. Fix $\phi \in Aut(H') \subseteq Aut(H)$. Construct $G_\phi = (G \setminus E(H), \phi)$, an instance for the prefix-GA problem. Observe that ϕ can be extended to an automorphism $\psi \in Aut(G)$, iff ψ solves prefix-GA with G_ϕ. Since $\phi \in Aut(H')$ let $G'_\phi = (G' \setminus E(H'), \phi)$ be another instance of the prefix-GA problem and then it follows, that $G_\phi = G'_\phi$. Observe that $G \setminus E(H)$ and $G' \setminus E(H')$ are identical graphs. Thus, by definition ϕ can be extended to an automorphism in G and G'. Any $\psi \in Aut(H) \setminus Aut(H')$ cannot be extended to an automorphism in $Aut(G)$, because $\psi \notin Aut_G(H)$. Thus, $Aut(G) = Aut(G')$. □

The next lemma shows, how to connect two subgraphs with arcs, such that each vertex in the first graph is connected to every vertex in the second graph, without changing the automorphism group.

Lemma 3.6 *Let $G[X], G[Y]$ be vertex-disjoint and setwise stabilized subgraphs of G. Suppose that in G all the edges with one endpoint in X and one endpoint in Y point from X to Y. Then G can be transformed with an AC^0 computable function into a graph G' over the same vertex set such that $Aut(G'_{[X,Y]}) = Aut(G_{[X,Y]})$. If $G[X], G[Y]$ are tournaments then G' is a tournament.*

Proof. Let $X = \{x_1, \ldots, x_m\}$, $Y = \{y_1, \ldots, y_n\}$ and let E_{XY} be the set of arcs, pointing from X to Y. Let $E_{YX} = \{ (y, x) \mid (x, y) \notin E_{XY} \}$. Observe that $E_{XY} \cup E_{YX}$ is a complete bipartite edge set. Then set $E(G') = E(G[X]) \cup E(G[Y]) \cup E_{XY} \cup E_{YX}$. Thus if $G[X], G[Y]$ are tournaments then G' as well. Now, fix any $\phi \in Aut(G_{[X,Y]})$, $e \in E_{XY}$, $e' \notin E_{XY}$. We now verify that any isomorphism has to map edges onto edges and nonedges onto nonedges: $e, \phi(e) \in E(G)$ and $e, \phi(e) \in E_{XY} \subseteq E(G'_{[X,Y]})$ and $e', \phi(e') \notin E(G)$, and $e', \phi(e') \in E_{YX}$. The other direction is similar. Thus, $Aut(G_{[X,Y]}) = Aut(G'_{[X,Y]})$. □

With Lemmas 3.5 and 3.6, we can now transform graph gadgets into tournaments and prove that they obey the same automorphism properties

as the modulo addition graph gadget. Now we define a tournament with the same automorphism properties as G^k.

Definition 3.7 *Fix $k \geq 3$ odd integer. The* tournament modulo addition graph gadget T^k *is defined by vertex set $V(T^k) = \{ x_a, y_a, z_a, u_{a,b} \mid a, b \in [0, k-1] \}$. Let $U, X, Y, Z \subseteq V(T^k)$ contain vertices denoted by indexed lower case letters each. Let $U_a = \{ u_{a,b} \mid b \in [0, k-1] \}$ for any $a \in [0, k-1]$. $E(T^k)$ unifies*

1. $\{ (x_a, x_{a\oplus i}), (y_a, y_{a\oplus i}), (z_a, z_{a\oplus i}) \mid a \in [0, k-1],\ i \in [1, \lfloor \frac{k}{2} \rfloor] \}$,
2. $\{ (u_{a,b},\ u_{a,b\oplus i}) \mid a, b, \in [0, k-1],\ i \in [1, \lfloor \frac{k}{2} \rfloor] \}$,
3. $\{ (u_{a,b},\ u_{a\oplus i, b\oplus b'}) \mid a, b, b' \in [0, k-1],\ i \in [1, \lfloor \frac{k}{2} \rfloor] \}$,
4. $\{ (x_a, u_{a,b}),\ (u_{i,b}, x_a) \mid a, b \in [0, k-1],\ i \in [0, k-1] \backslash \{a\} \}$,
5. $\{ (y_b, u_{a,b}),\ (u_{a,i}, y_b) \mid a, b \in [0, k-1],\ i \in [0, k-1] \backslash \{b\} \}$,
6. $\{ (u_{a,b},\ z_{a\oplus b}),\ (z_i,\ u_{a,b}) \mid a, b \in [0, k-1],\ i \in [0, k-1] \backslash \{a \oplus b\} \}$,
7. $\{ (x_a, y_b),\ (x_a, z_b),\ (y_a, z_b) \mid a, b \in [0, k-1] \}$.

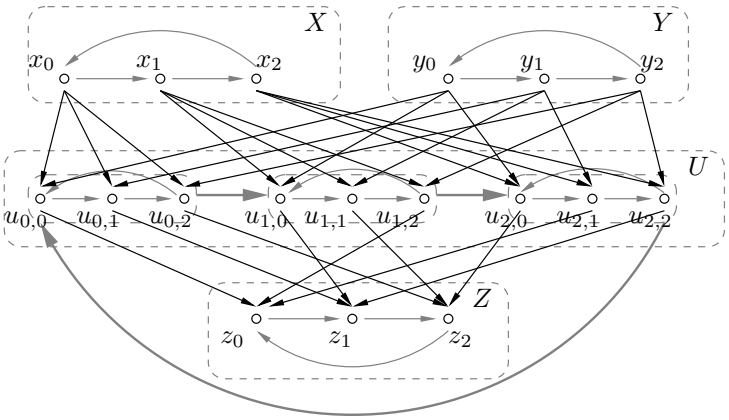

Figure 2: Sketch of tournament modulo addition graph gadget T^3

Remark that $V(T^k) = V(G^k)$. In item 1 we define cyclic tournaments for induced subgraphs on vertex sets X, Y and Z. In item 2 we define cyclic tournaments $T^k[U_a]$ induced on vertices $U_a = \{ u_{a,b} \mid b \in [0, k-1] \}$ for any fixed $a \in [0, k-1]$. In item 3 we describe the connection between subgraphs $T^k[U_a]$, such that automorphisms in $Aut(T^k)$ act cyclically on subgraphs $T^k[U_a]$ for all $a \in [0, k-1]$. Items 4 to 7 describe complete bipartite edge sets among U, X, Y, Z. Figure 2 shows T^3 and contains edge sets of items 4 to 7 partially.

Lemma 3.8 *There is an* AC^0 *computable function that transforms* G^k *with odd* $k \geq 3$ *into the tournament modulo addition graph gadget* T^k *containing unique automorphisms as described in Lemma 3.2.*

Proof. In Lemma 3.2 the unique automorphisms act cyclically on vertex sets $G^k[X]$,$G^k[Y]$ and $G^k[Z]$. First, regard $G^k[X]$; $Aut(G^k[X]) = Sym(G^k[X])$ and $Aut_{G^k}(G^k[X])$ is generated by permutation $(x_0 \dots x_{k-1})$. Clearly, $Aut(T^k[X]) \subseteq Aut(G^k[X])$ and because of $T^k[X]$ containing cyclic automorphisms, $Aut_{G^k}(G^k[X]) \subseteq Aut(T^k[X])$. Apply Lemma 3.5 and replace $G^k[X]$ by $T^k[X]$ in G^k, without changing the automorphism group. The same holds for $G^k[Y]$ and $G^k[Z]$. Second, regard $G^k[U]$. $Aut(G^k[U]) = Sym(G^k[U])$ and $Aut_{G^k}(G^k[U])$ is generated by ϕ, ψ, which are defined by the relation: $\phi(u_{i,j}) \to u_{i,j\oplus 1}$ and $\psi(u_{i,j}) \to u_{i\oplus 1,j}$ for all $i, j \in [0, k-1]$. It follows that $Aut_{G^k}(G^k[U]) \subseteq Aut(T^k[U])$. Apply Lemma 3.5 and replace $G^k[U]$ by $T^k[U]$ in G^k, without changing the automorphism group. Third, the edge sets of item 4 to item 7 in definition of $E(T^k)$ can be described as exchanging undirected edges between stabilized vertex sets X, Y, Z, U by arcs. By Lemma 3.6, this also keeps the automorphism group unchanged. Thus T^k is the union of all these modifications on G^k which can be computed in AC^0. □

With Lemma 3.8 we can prove that the replacement of gadgets G^k in $G(C)$ by tournament gadgets T^k does not change the automorphism group of $G(C)$. With Lemma 3.9, we complete the proof of Theorem 3.4.

Lemma 3.9 *Let* C *be a circuit of modulo addition gates in* $\mathbb{Z}_k$*, with odd* $k \geq 3$ *and output value* $s \in [0, k-1]$*. Construct under* AC^0 *many-one reductions a tournament* $T(C)$ *containing nontrivial prefix automorphisms,iff* C *outputs* s*.*

Transform $G(C)$ (of Definition 3.3) into a tournament $T(C)$, such that $Aut(G(C)) = Aut(T(C))$ and $T(C)$ contains a nontrivial prefix automorphism, iff $G(C)$ does. To prove this, we apply Lemmas 3.5 and 3.8.
Proof. Transform $G(C)$ (of Definition 3.3) into a tournament $T(C)$, such that $Aut(G(C)) = Aut(T(C))$ and $T(C)$ contains a nontrivial prefix automorphism, iff $G(C)$ does. First, apply Lemma 3.5 and 3.8. Exchange the subgraphs $G^k(C_i)$ by tournament gadgets $T^k(C_i)$ for all $i \in [m]$ under AC^0 many-one reductions. Since $Aut_{G(C)}(G^k(C_i)) \subseteq Aut(T^k(C_i)) = Aut(G^k(C_i))$, this does not change the automorphism group of $G(C)$. Second, let $p, q \in [m]$ with $p < q$. Let $E_{p,q}$ be the edge set with one vertex in $G^k(C_p)$ and the other one in $G^k(C_q)$ and replace them by arcs which point from vertices in $G^k(C_q)$ to $G^k(C_p)$. Applying Lemma 3.6, we amend the edge set between subgraphs $G^k(C_p)$ and $G^k(C_q)$ by a complete bipartite edge set, denoted $E'_{p,q}$. Thus, we get $T(C)$ defined as follows:

$$V(T(C)) = \bigcup_{p\in[m]} V(T^k(C_p)) = V(G(C)),$$

$E(T(C)) = \bigcup_{p\in[m]} E(T^k(C_p)) \cup \bigcup_{p,q\in[m],p<q} E'_{p,q}$,
$E'_{p,q} = \{\, (v,u) \mid \{u,v\} \in E_{p,q}, u \in V(G^k(C_p)), v \in V(G^k(C_q)) \,\} \cup$
$\{\, (u,v) \mid \{u,v\} \notin E_{p,q}, u \in V(G^k(C_p)), v \in V(G^k(C_q)) \,\}$.

Apply the same prefixes and coloring to vertices in $T(C)$ as for $G(C)$, such that $T(C)$ contains nontrivial automorphisms obeying prefixes, iff $G(C)$ does. Finally, we examine that this construction can be done in AC^0. Since $V(T^k) = V(G^k)$, we just have to compute $E(T^k)$. This provides local information, since the decision, whether two vertices are connected by an edge, can be decided by knowing their indices and to which vertex sets U, X, Y, Z they belong. The same holds for construction of $E'_{p,q}$ and adapting the prefixes from $G(C)$ to $T(C)$. □

3.2 Hardness Results of TI for NL, #L , $\mathrm{C}_=\mathrm{L}$ and PL

Now we introduce graph gadgets for simulation of AND- and OR-gates in circuits. Recall the following lemma.

Lemma 3.10 [21] *Given a uniform family of circuits C_n with logarithmic depth and polynomial size and given n tuples of graphs $((G_i, H_i)(I_i, J_i)) \in PGI$, then there is an AC^0 computable function, constructing a tuple $((G, H)(I, J)) \in PGI$ with the property that $G \cong H$, iff C_n outputs 1, and $I \cong J$, iff C_n outputs 0. The i-th input to C_n consists of the bit of the Boolean value of the statement $G_i \cong H_i$.*

We transform them into tournament gadgets, to get the same hardness results for TI which hold for GI. A NC^1 circuit can be simulated by a balanced DLOGTIME uniform family of circuits with fan-out 1, logarithmic depth, polynomial size and alternating layers of and-gates and or-gates [5].

Definition 3.11 *Let $((G_\wedge, H_\wedge)(I_\wedge, J_\wedge)) \in PGI$ be the graph tuple for simulation of conjunction and $((G_\vee, H_\vee)(I_\vee, J_\vee)) \in PGI$ of disjunction, containing $((G_0, H_0)(I_0, J_0))$, $((G_1, H_1)(I_1, J_1)) \in PGI$ as in proof of Theorem 4.3 in [21].*

The graph tuples for conjunction have the following properties:
$G_\wedge \cong H_\wedge$ iff $G_0 \cong H_0$ and $G_1 \cong H_1$; $I_\wedge \cong J_\wedge$ iff $G_0 \ncong H_0$ or $G_1 \ncong H_1$ (in this case $I_0 \cong J_0$ or $I_1 \cong J_1$). Similarly, the graph tuples for disjunction: exchange $\wedge$ with $\vee$, also exchange 'and' with 'or' and vice versa. For clear notation, we will apply prefixes e.g. PGI-$G_\wedge$, PGI-$G_\vee$. If the context is clear, we omit these prefixes. Now we transform PGI-tuples into tuples of tournaments.

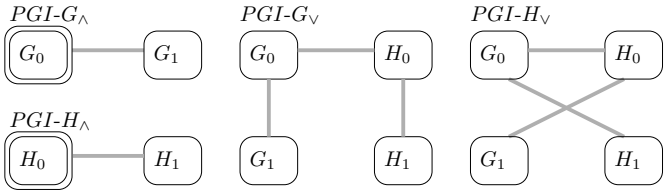

Figure 3: PGI-Tuple simulating $\wedge$ and $\vee$ gates [21]

Lemma 3.12 *There is an* AC^0 *computable function for translation of graph gadgets PGI-*$G_\vee$ *and PGI-*$H_\vee$ *into tournament graph gadgets PTI-*$G_\vee$*, PTI-*$H_\vee$ *(see Definition 3.13) having the same isomorphism properties as in Definition 3.11.*

Definition 3.13 *A* PTI-graph tuple *is a tuple of rigid tournaments* $((G,H),(I,J))$ *with* $G \cong H$*, iff* $I \ncong J$*. Let PTI be the set of all such tuples. Define for conjunction* $((G_\wedge,H_\wedge)(I_\wedge,J_\wedge)) \in PTI$ *(write e.g. PTI-*$G_\wedge$*) and for disjunction*
$((G_\vee,H_\vee)(I_\vee,J_\vee)) \in PTI$ *(write e.g. PTI-*$G_\vee$*) as follows:*

First, PTI-$G_\wedge$ *contains tournaments* G_0, G_1 *and a set of arcs, pointing from every vertex in* G_0 *to every vertex in* G_1*. Replace* G_0, G_1 *in PTI-*$G_\wedge$ *by* H_0, H_1 *for obtaining PTI-*$H_\wedge$*, by* I_0, I_1 *for PTI-*$I_\vee$ *and by* J_0, J_1 *for PTI-*$J_\vee$*.*

Second, let $i \in [0,1]$ *and* $j \in [0,2]$*. Let* X *be a graph as in Figure 4. PTI-*$G_\vee$ *contains subgraphs* $X, G_0, G_0', G_1, G_1', H_0$ *and* H_1 *with* G_0', G_1' *copies of* G_0, G_1*. Let* $E(PTI\text{-}G_\vee) = E_1 \cup \ldots \cup E_4$*.* E_1 *unifies edges of all subgraphs.* E_2 *contains edges* $(x_{i,0}, v)$ *for all* $v \in G_i$ *(call* G_i associated *to* $x_{i,0}$*), and similar edge sets with* $x_{i,1}$ *associated to* G_i' *and* $x_{i,2}$ *to* H_i*.* E_3 *contains the following arcs: If* $(x,x') \in E(X)$ *then connect every vertex of the subgraph associated to* x *with arcs, pointing to every vertex of the subgraph associated to* x'*.* E_4 *contains* (u,v) *for all* $u \in V(PTI\text{-}G_\vee \setminus X)$*,* $v \in V(X)$*, iff* $(v,u) \notin E_2$*. Now construct PTI-*$H_\vee$ *with minor changes. Associate* $x_{1,1}$ *with* H_1 *and* $x_{1,2}$ *with* G_1'*. The rest of the construction is the same. Now replace subgraphs* G_0, G_1, H_0, H_1 *in PTI-*$G_\vee$ *(and PTI-*$H_\vee$*) by* I_0, I_1, J_0, J_1 *in this order and obtain PTI-*$I_\wedge$ *(and PTI-*$J_\wedge$*).*

Lemma 3.14 *There is an* AC^0 *computable function, such that any of the PGI-tuples simulating and-gates and or-gates can be transformed into PTI-tuples with the same isomorphism properties.*

Proof. First, the construction of PTI-$G_\wedge$ can be obtained of PGI-$G_\wedge$, if coloring and undirected edges with both ends in different subgraphs G_0, G_1 and H_0, H_1 are replaced by arcs. The same holds for PTI-$H_\wedge$, PTI-$I_\vee$, PTI-$J_\vee$.

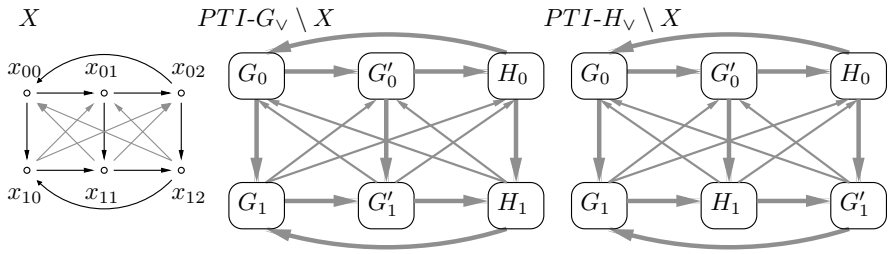

Figure 4: Construction of tournaments simulating $\wedge$ and $\vee$ gates

Second, we transform PGI-$G_\vee$ to PTI-$G_\vee$. Let $i \in [0,1]$, $j \in [0,2]$. Any automorphism $\phi \in Aut(X)$ acts cyclically on the vertices $x_{i,0}, x_{i,1}, x_{i,2}$. Every vertex in X has one associated subgraph in PTI-$G_\vee \setminus X$ via E_2. E_3 transfers the structure of $E(X)$ onto connection among associated subgraphs in $E(PTI\text{-}G_\vee \setminus X)$. So far, regarding subgraphs as vertices, then $Aut_{PTI\text{-}G_\vee \setminus E_4}(X)$ would equal $Aut(X)$.

Apply E_4 to $E(PTI\text{-}G_\vee)$, as in Lemma 3.6 this does not change the automorphism group. Thus, associated subgraphs to $x_{i,j}$ must be mapped via $\phi \in Aut(PTI\text{-}G_\vee)$ onto the associated subgraph of $\phi(x_{i,j})$. The same holds for X and PTI-$H_\vee$. Observe, that this can be done in AC^0. We now consider the isomorphism properties. Observe, that any isomorphism ϕ mapping PTI-$G_\vee$ onto PTI-$H_\vee$ satisfies:

1. $x_{i,j} \mapsto x_{i,j\oplus 0}$ if $G'_1 \cong H_1 \cong G'_1$,
2. $x_{i,j} \mapsto x_{i,j\oplus 1}$ if $G'_0 \cong H_0 \cong G_0$ and $G_1 \cong H_1 \cong G_1$,
3. $x_{i,j} \mapsto x_{i,j\oplus 2}$ if $G_0 \cong H_0 \cong G'_0$ (and $G_1 \cong G'_1$).

Either $G'_1 \cong H_1$ (step 1) or $G_0 \cong H_0$ (step 3) or both are isomorphic (step 2). Thus, this encodes an or-function. We get PTI-$I_\wedge$ and PTI-$J_\wedge$, if we exchange in PTI-$G_\vee$ and PTI-$H_\vee$ the subgraphs G_0, G_1, H_0, H_1 by I_0, I_1, J_0, J_1 in this order. □

With all the graph gadgets (G^k, PGI-$G_\wedge$, PGI-$G_\vee, \dots$) as defined so far, Torán proved that GI is hard for NL, #L , $\mathrm{C}_=\mathrm{L}$ and PL under AC^0 many-one reductions [21]. We prove that the same lower bounds hold for TI.

Theorem 3.15 *Tournament isomorphism is hard for* NL, #L , $\mathrm{C}_=\mathrm{L}$ *and* PL *under* AC^0 *many-one reductions.*

Proof. For proving all the hardness bounds, graph gadgets are needed as described above. Regard Theorems 4.1, 4.4 and Corollaries 4.5, 4.6 in [21] for details. First, $\mathrm{MOD_kL}$ circuits are needed to compute the result of a #L function $f(x) \mod k$. These gadgets encode $f(x) \mod k$ for a set of r

different primes $k \in \{ k_1, \ldots, k_r \mid 3 \leq k_1 < \cdots < k_r \}$ (in Chinese remainder representation). The results are inputs to a NC^1 -circuit, which compute bits of $f(x)$. Since tournament modulo addition gadgets for even k are not defined, the prime 2 cannot be chosen. In every step, the graph gadgets serve as subgraphs in new PGI-tuples. Applying Lemma 3.9 and 3.14, the graph gadgets can be transformed into tournaments under AC^0 many-one reductions. □

3.3 Hardness Results of TI for DET

Observe that DET $\leq_m^{\mathrm{AC}^0}$ GI (Theorem 4.9 in [21]) and that the complexity class DET coincides with $\mathrm{NC}^1(\#\mathrm{L})$. We already described, how NC^1 -circuits and #L -functions can be reduced to graph gadgets. For implementing oracle questions, with another graph gadget every #L function f can be transformed in AC^0 into a sequence of PGI-tuples, encoding the bits of $f(x)$. The input $x \in \Sigma^n$ is also encoded as PGI-tuples. For details see proof of Lemma 4.7 in [21].

Definition 3.16 [21] *The* oracle graph gadget Gad_k *contains subgraphs* $G_a, H_a^h,\ I_a^i, J_a^{i,j}$ *with* $h \in [1, k-1]$, $i, j \in [0, k-1]$, *which are copies of graphs in* $((G_a, H_a)(I_a, J_a)) \in PGI$, *encoding bit* x_a *of* $f(x) \mod k$. *Let* $W = \{w_0, \ldots, w_{k-1}\}$, $Z = \{z_0, \ldots, z_{k-1}\} \subseteq V(Gad_k)$. *Henceforth, for simplifying notations, let* $W^0 = G_a$ *and* $W^h = H_a^h$. *We also denote* $Z[i,j] = J_a^{i,j}$ *for* $j \neq i$ *and* $Z[i,i] = I_a^i$ *for* $i, j \in [0, k-1]$. *Let* $Z^i = \bigcup_{j \in [0,k-1]} Z[i,j]$. *We now describe the edge set* $E(Gad_k)$ *as the union of the following edge sets*

1. $E(G_a), E(H_a^h), E(I_a^i), E(J_a^{i,j})$,
2. $\{ \{u, v\} \mid u = z_i,\ v \in Z^i \ \ or \ \ u \in W^i,\ v = w_i \}$,
3. $\{ \{u, v\} \mid u \in Z[i,j],\ v \in W^j \}$,
4. $\{ (u, v) \mid u = w_i, v = w_{i \oplus 1} \ \ or \ \ u = z_i, v = z_{i \oplus 1} \}$.

This gate has the property, that if $G_a \cong H_a$ then for $c \in [0, k-1]$, any automorphism mapping z_i to $z_{i \oplus c}$ also maps w_i onto $w_{i \oplus c}$. But, if $I_a \cong J_a$ then any automorphism mapping z_i to $z_{i \oplus c}$ will fix all vertices w_i.

For proving hardness results for GI, the graphs have PGI tuples as subgraphs (Lemma 4.8 in [21]). With the assumption, that the subgraphs of item 1 are PGI tuples and that any automorphism acts cyclically on vertex sets W and Z, then any automorphism of Gad_k acts cyclically on subgraphs $\{ Z[i,j] \mid j \in [0, k-1] \}$ for each $i \in [0, k-1]$, and on subgraphs Z^i for all $i \in [0, k-1]$. Therefore, we can introduce arcs as in item 4, to restrict the automorphism group of Gad_k and for simplifying proofs.

We need all graph gadgets described so far to prove that GI is hard for DET. We want to show the same result for TI.

Theorem 3.17 *$DET \leq_m^{\mathrm{AC}^0} TI$.*

More precisely, we show hardness of TI for $\mathrm{NC}^1(\#\mathrm{L})$. Oracle questions will be implemented by using oracle graph gadgets like Gad_k. We transform Gad_k into a tournament and consider its automorphism properties.

Definition 3.18 *Let $k \geq 3$ be an odd integer and $i, j \in [0, k-1]$. The* oracle tournament gadget *$TGad_k$ has vertex set $V(TGad_k) = V(Gad_k)$ and edge set $E(TGad_k)$ as the union of the following edge sets (see also Figure 5)*

1 $E(Gad_k)$ *but write '(u,v)' for items 2, 3 in Definition 3.16 of* $E(Gad_k)$,

2 $E(W) \cup E(Z) = \{\, (w_i, w_{i\oplus h}),\ (z_i, z_{i\oplus h}) \mid h \in [1, \lfloor \frac{k}{2} \rfloor] \,\}$,

3 $\{\, (u,v) \mid u \in W^i,\ v \in W^j, \quad \textit{iff } (w^i, w^j) \in E(W) \,\}$,

4 $\{\, (u,v) \mid u \in Z^i,\ v \in Z^j, \quad \textit{iff } (z_i, z_j) \in E(Z) \,\}$,

5 $\{\, (u,v) \mid u \in Z[i,j],\ v \in Z[i, j \oplus h] \quad \textit{with } h \in [1, \lfloor \frac{k}{2} \rfloor] \,\}$,

6 $\{\, (u,v) \mid u \in Z^i,\ v = z_j \quad \textit{with } j \neq i \,\}$,

7 $\{\, (u,v) \mid u \in W^i,\ v \in Z^j \quad \textit{with } j \neq i \,\}$,

8 $\{\, (u,v) \mid u = w_i,\ v \in W^j \quad \textit{with } j \neq i \,\}$.

Lemma 3.19 *If Gad_k contains tournaments (PTI-tuples) as subgraphs of item 1 in Definition 3.16, then Gad_k with odd $k \geq 3$ can be transformed into a tournament $TGad_k$ under AC^0 many-one reductions, without changing the automorphism group.*

Proof. Item 1 in Definition 3.18 of $TGad_k$ can be done in AC^0 . Since in Gad_k automorphisms are considered, acting cyclically on vertex set W, Apply Lemma 3.5 and replace edge set of $Gad_k[W]$ by that of a cyclic tournament as described in item 2 for $TGad_k$. The same is done for vertex set Z, for both in AC^0.

The edge set of item 3 describes, how the subgraphs W^i are transformed into tournaments. Thus, if an automorphism ϕ maps w_i onto $w_{i\oplus c}$ with $c \in [0, k-1]$ then by item 1 the subgraph W^i must be maped onto $W^{i\oplus c}$ and thus, ϕ acting cyclically on vertex set W also acts cyclically on whole subgraphs W^i. Since the formulation 'iff $(w^i, w^j) \in E(W)$' is the same as 'iff $j = i \oplus h, h \in [1, \lfloor \frac{k}{2} \rfloor]$', only local information is needed for adjoining edges, this can be done in AC^0. The same is done for item 4 with subgraphs Z^i instead of W^i and vertex set Z instead of W.

Item 5 describes the edge sets inside of Z^i, for now keep i fixed. Any automorphism ϕ in Gad_k acts cyclically on subgraphs $Z[i,0], \ldots, Z[i,k-1]$. By item 1 there is an edge-connection between subgraphs $Z[i,j]$ and W_j and by item 3 there is the connections among all subgraphs in W. This

directly leads to conditions for connecting subgraphs in Z_i. That is, $Z[i,j]$ is conntected to $Z[i,j \oplus c]$, iff W_j is connected to $W_{j \oplus c}$ for any $c \in [0, k-1]$. This can be regarded as a generalization of Lemma 3.5. Now, Z^i forms a tournament if contained subgraphs $Z[i,j]$ (that is $I_a^i, J_a^{i,j}$) are tournaments. Figure 5 shows the current construction up to step 5.

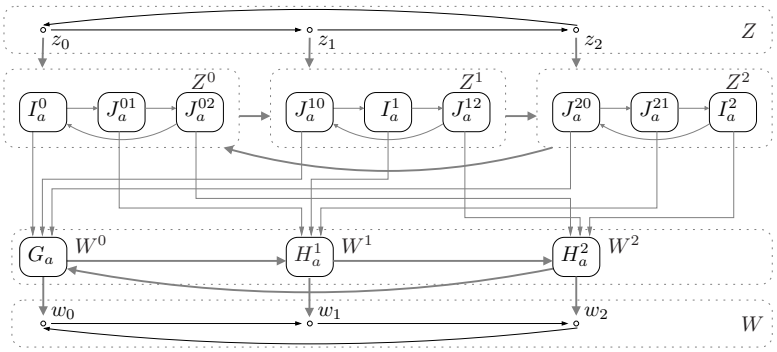

Figure 5: Graph gadget $TGad_3$ after performing edge sets of items 1 to 5

Items 6 to 9 follow immediately by Lemma 3.6 since the automorphism group is not changed. This also can be done in AC^0. Regard thereby, that the layers with vertices Z, $V(Z^i)$, $V(W^i)$ and W are setwise stabilized.

Finally, with $E(TGad_k)$ the graph $TGad_k$ is a tournament and has the desired automorphism properties as oracle graph gadget Gad_k. □

Our aim is to construct a graph G^* with prefixes, which contains non-trivial prefix automorphisms, iff an NC^1 circuit with #L oracle questions outputs *true*. The oracle questions thereby must have the same size. This makes it possible to use exactly one type of oracle graph gadget Gad_k (for exactly one value of k). For more details we refer to [21]. Any subgraph like Gad_k inside of G^* is setwise stabilized in $Aut(G^*)$. By Lemma 3.5 and 3.19, transform any subgraph in G^* isomorphic to Gad_k into a tournament $TGad_k$ without changing the automorphism group of G^*. Then apply Lemma 3.6 in order to connect this graph gadget with any other graph gadget. Since G^* contains only graph gadgets as described in this chapter and every graph gadget is setwise stabilized in automorphism group of G^*, replace them all by tournaments and connect them with each other as described in Lemma 3.6. Thus we can transform G^* into a tournament T^* under AC^0 many-one reductions. It follows that TI is hard for $\mathrm{NC}^1(\#\mathrm{L})$ and thus for DET. Corollary 3.20 follows by the fact that the $\mathrm{MOD_kL}$ hierarchy is logspace Turing reducible to DET.

Corollary 3.20 $\mathrm{MOD}_k\mathrm{L} \leq_T^L$ *TI for any* $k \geq 2$.

4 Hardness Results for Tournament Automorphism

GA is many-one hard for the MOD_kL hierarchy (Theorem 5.1 [21]). Transform a circuit C into a rigid graph $G(C)$ as in Definition 3.3, having a unique automorphism satisfying prefixes, iff the output value of the circuit is 1. Take two copies G_1 and G_2 of graph G and apply colorings $Col(G_1), Col(G_2)$ to the vertices which represent the input and output values of the circuit in order to encode the prefixes the same way as for prefix-GA. Thus the graph $G_1 \cup G_2$ has a nontrivial automorphism, iff the output of the original circuit is 1. We show how this result also holds for TA.

Theorem 4.1 $\text{MOD}_k\text{L} \leq_m^{\text{AC}^0}$ *TA with* $k \geq 3$ *odd integer.*

Proof. First, transform $G(C)$ into a tournament $T(C)$ as described in proof of Lemma 3.9. Instead of taking two copies we need three copies T_0, T_1, T_2 of $T(C)$ and apply the colorings $Col(G_1)$ to T_0 and $Col(G_2)$ to T_1 and T_2. Then include complete bipartite edge sets $\{ (u, v) \mid u \in V(T_i), v \in V(T_{i\oplus 1}), i \in [0, 2] \}$. Thus $T(C)$ is a tournament and contains two nontrivial automorphisms (that is mapping T_0 onto T_1 or T_2), iff $G(C)$ contains one nontrivial automorphism. □

Now we discuss the counterpart of PGI and PTI tuples, in order to prove lower bounds for TA. Therefore, we define the following graph tuples.

Theorem 4.2 $DET \leq_m^{\text{AC}^0}$ *TA.*

Definition 4.3 [21] *A* PGA-graph tuple *is a tuple of rigid graphs* $((G, H), (I, J))$ *with* $G \cong H \Leftrightarrow I \ncong J$. *Let PGA be the set of all such tuples. The graph tuple* $(G_\wedge, H_\wedge)(I_\wedge, J_\wedge)) \in PGA$ *(write e.g.* $PGA\text{-}G_\wedge$*) for simulating conjunction and* $(G_\vee, H_\vee)(I_\vee, J_\vee)) \in PGA$ *(write e.g.* $PGA\text{-}G_\vee$*) for disjunction is defined as follows:*

$PGA\text{-}G_\wedge = PGI\text{-}G_\wedge$, $PGA\text{-}H_\wedge = PGI\text{-}H_\wedge$, *the same for* $PGA\text{-}I_\wedge, PGA\text{-}J_\wedge$. *For* $PGA\text{-}G_\vee, PGA\text{-}H_\vee$ *regard Figure 6.* $PGA\text{-}I_\wedge$ *can be obtained from* $PGA\text{-}G_\vee$ *and* $PGA\text{-}J_\wedge$ *from* $PGA\text{-}H_\vee$ *if subgraphs* G_i, H_i, I_i, J_i *will be replaced by* I_i, J_i, G_i, H_i *for* $i \in [0, 1]$ *in this order.*

These tuples have the properties, that $G_\wedge \cong H_\wedge \Leftrightarrow G_0 \cong H_0$ 'and' $G_1 \cong H_1$, and $G_\vee \cong H_\vee \Leftrightarrow G_0 \cong H_0$ 'or' $(G_1 \cong H_1 \wedge I_0 \cong J_0)$. Moreover, if all the subgraphs are rigid then the new graphs forming tuples are rigid as well. We transform now these PGA-tuples into tuples of tournaments.

Lemma 4.4 *There is an* AC^0 *computable function, such that the PGA graph tuples can be transformed into tournaments, having the same automorphism properties and rigidity properties as in Definition 4.3.*

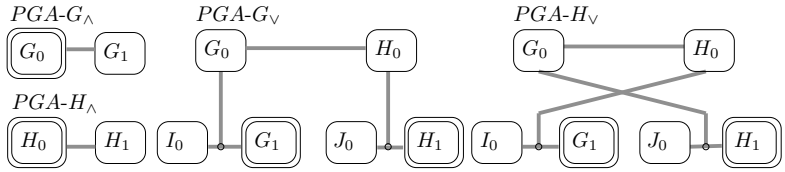

Figure 6: PGA-tuples simulating $\wedge$ and $\vee$ functions [21]

Definition 4.5 *A* PTA-graph tuple *is a tuple of rigid tournaments* $((G,H),(I,J))$ *with* $G \cong H \Leftrightarrow I \ncong J$. *Let PTA be the set of all such tuples.*

The graph tuple $(G_\wedge, H_\wedge)(I_\wedge, J_\wedge)) \in PTA$ *(write e.g. PTA-*$G_\wedge$*) for simulating conjunction and* $(G_\vee, H_\vee)(I_\vee, J_\vee)) \in PTA$ *(write e.g. PTA-*$G_\vee$*) for disjunction is defined as follows:*

The graph PTA-$G_\vee$ *contains subgraphs* $X, G_0, G_0', H_0, A_0, A_1, A_2$. *Let subgraphs* G_0', G_1', I_0' *be copies of* G_0, G_1, I_0. *Let* X *be defined as shown in Figure 7.* A_0 *contains subgraphs* I_0 *and* G_1, *with vertices in* I_0 *pointing to all vertices in* G_1. A_1 *is a copy of* A_0. A_2 *is constructed like* A_0, *containing* J_0 *instead of* I_0 *and* H_1 *instead of* G_1. *Let subgraph* G_0 *be associated to* $x_{0,0}$, G_0' *to* $x_{0,1}$ *and* H_0 *to* $x_{0,2}$ *and let the* A_i *be associated to* $x_{1,i}$ *for* $i \in [0,2]$. *Concerning edge sets, the rest of the construction is similar to that of PTI-*$G_\vee$.

The subgraph $H_\vee$, *is constructed like* $G_\vee$ *but with* A_1 *associated to* $x_{1,2}$ *and* A_2 *to* $x_{1,1}$. *Obtain* $I_\wedge$ *from* $G_\vee$ *and* $J_\wedge$ *from* $H_\vee$, *if subgraphs* G_i, H_i, I_i, J_i *will be replaced by* I_i, J_i, G_i, H_i *for* $i \in [0,1]$ *in this order.*

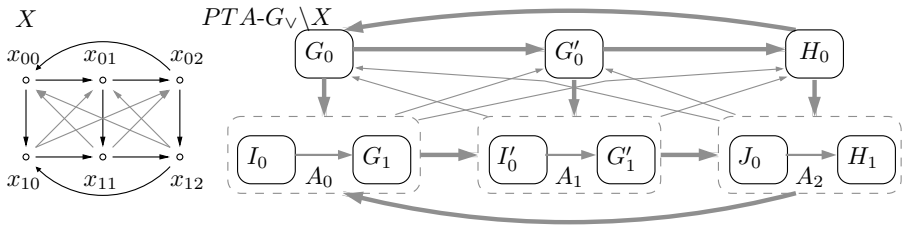

Figure 7: PTA-tuples simulating $\wedge$ and $\vee$ functions

Proof. The proof is like that of Lemma 3.14, simulate the alternating layers of AND's and OR's of an NC^1 circuit with certain PTA-tuples. The main difficulty is to preserve the rigidity of the tuple components.

First, the PTA-and-gadgets equal the PTI-and-gadgets. Thus, we can argue as in proof of Lemma 3.14. With the directed edge set between vertices of G_0 pointing to G_1 and Lemma 3.6 it follows, that $Aut(PTA\text{-}G_\wedge) = Aut(G_0) \times Aut(G_1)$. Thus, if both subgraphs are rigid, then PTA-$G_\wedge$ is

rigid. For the other tuple components PTA-$H_\wedge$, PTA-$I_\vee$ and PTA-$J_\vee$, the proof is similar.

Second, the automorphism properties for PTA-or-gadgets are similar to that for PTI-tuples as in Lemma 3.12. Recall, that any automorphism must map $x_{i,j}$ onto $x_{i,j\oplus k}$ for $i \in [0,1]$, $j,k \in [0,2]$. PTA-$G_\vee$ is rigid, since the mapping of $x_{i,j}$ onto $x_{i,j\oplus 1}$ will map associated subgraphs G'_0 onto H_0 and I'_0 onto J_0. Since $G_0 \cong H_0 \Leftrightarrow I_0 \ncong J_0$, this mapping is no automorphism of PTA-$G_\vee$. The same holds for mapping $x_{i,j}$ onto $x_{i,j\oplus 2}$. The automorphism properties of PTA-$G_\vee$ are the same as that for PTI-$G_\vee$. For the other tuple components PTA-$H_\vee$, PTA-$I_\wedge$ and PTA-$J_\wedge$, the proof is similar. □

An immediate consequence of this result is, that TA is hard for NC^1 under AC^0 many-one reductions. By applying Theorem 4.1, it is possible to prove hardness of TA for complexity class DET. The proof of this result follows (similar to that in [21]) exactly the same lines as that for Theorem 3.17 taking in consideration that the tournament graph pairs produced in the reduction from Theorem 3.4 are rigid and that the gadgets in the proof of Theorem 3.17 also preserve rigidity. Similar to the Corollary 3.20, the Corollary 4.6 immediately follows:

Corollary 4.6 $\mathrm{MOD}_k\mathrm{L} \leq_T^L$ *TA for any* $k \geq 2$.

Acknowledgments. I am grateful to my supervisor Jacobo Torán , Sebastian Dörn, Thanh Minh Hoang for helpful discussion and the anonymous referees for helpful comments and suggestions.

References

[1] V. Arvind, R. Beigel, A. Lozano, *The Complexity of Modular Graph Automorphism* Symp. on Theoret. Aspects of Computer Sci., 1998, pp. 172-182.

[2] V. Arvind, B. Das, P. Mukhopadhyay, *On Isomorphism and Canonization of Tournaments and Hypertournaments* ISAAC 2006, pp. 449-459.

[3] E. Allender, M. Ogihara, *Relationships among PL, #L and the determinant* RAIRO Inform. Theor. Appl., 30, 1996, pp. 1-21.

[4] C. Alvarez, B. Jenner, *A very hard logspace counting class* Theoretical Computer Science 107, 1993, pp. 3-30.

[5] D.A.M. Barrington, N. Immerman, H. Straubing, *On uniformity within* NC^1 J. Comput. System Sci. 41, 1990, pp. 274-306.

[6] L. Babai, E. Luks, *Canonical labeling of graphs* in Proceedings of the 15th Annual ACM Symposium on Theory of Computing, 1983, pp. 171-183.

[7] R. Boppana, J. Hastad, S. Zachos, *Does co-NP have short interactive proofs?* Inform. Process. Lett. 25, 1987, pp. 27-32.

[8] G. Buntrock, C. Damm, U. Hertrampf, C. Meinel, *Structure and importance of logspace-MOD-classes* Math. System Theory 25, 1992, pp. 223-237.

[9] S. R. Buss, *Alogtime algorithms for tree isomorphism, comparison, and canonization in Computational Logic and Proof Theory* Lecture Notes in Comput. Sci. 1289, Springer Verlag Berlin 1997, pp. 18-33.

[10] S. A. Cook, *A taxonomy of problems with fast parallel algorithms* Information and Control 64, 1985, pp. 2-22.

[11] J. Gill, *Computational complexity of probabilistic Turing machines* SIAM J. Comput. 6, 1977, pp. 675-695.

[12] J. L. Gross, J. Yellen, *Discrete Mathematics and its Applications - Handbook of Graph Theory* CRC Press LLC, 2004.

[13] J.E. Hopcroft, R.E. Tarjan, *A V^2 algorithm for determining isomorphism of planar graphs* 1971, pp. 32-34.

[14] J. Köbler, U. Schöning, J. Torán , *The Graph Isomorphism Problem - Its Structural Complexity* Prog. Theor. Comp.Sci., Birkhaeuser, Boston, MA, 1993.

[15] S. Lindell, *A logspace algorithm for tree canonization* in Proceedings of the 24th Annual ACM Symposium on Theory of Computing, 1992, pp. 400-404.

[16] E. Luks, *Isomorphism of bounded valence can be tested in polynomial time* J. Comput. System Sci. 25, 1982, pp. 42-65.

[17] E. Luks, *Parallel algorithms for perumtation groups and graph isomorphism* in Proc of the 27th IEEE Symp. on Found. of Comp. Sci. 1986, pp. 292-302.

[18] G.L. Miller, J.H. Reif, *Parallel tree contraction* Part 2: further applications SIAM Journal on Computing 20(6), 1991, pp. 1128-1147.

[19] W. Ruzzo, J. Simon, M. Tompa, *Space bounded hierarchies and probabilistic computations* J. Comput. System Sci. 28, 1984, pp. 216-230.

[20] U. Schöning: *Graph isomorphism is in the low hierarchy* J. Comput. System Sci. 37, 1988, pp. 312-323.

[21] J. Torán , *On the Hardness of Graph Isomorphism* SIAM J. Comput. Vol. 33, No. 5, 2004, pp. 1093-1108.