Reducing Tarski to Unique Tarski (In the Black-Box Model)

Xi Chen
Columbia University, New York, NY, USA

Yuhao Li
Columbia University, New York, NY, USA

Mihalis Yannakakis
Columbia University, New York, NY, USA

Abstract

We study the problem of finding a Tarski fixed point over the k-dimensional grid $[n]^k$. We give a black-box reduction from the Tarski problem to the same problem with an additional promise that the input function has a unique fixed point. It implies that the Tarski problem and the unique Tarski problem have exactly the same query complexity. Our reduction is based on a novel notion of partial-information functions which we use to fool algorithms for the unique Tarski problem as if they were working on a monotone function with a unique fixed point.

2012 ACM Subject Classification Theory of computation → Problems, reductions and completeness; Theory of computation → Exact and approximate computation of equilibria

Keywords and phrases Tarski fixed point, Query complexity, TFNP

Digital Object Identifier 10.4230/LIPIcs.CCC.2023.21

Related Version *ECCC Report*: https://eccc.weizmann.ac.il/report/2023/073/

Funding *Xi Chen*: Supported by NSF grants IIS-1838154, CCF-2106429 and CCF-2107187.
Yuhao Li: Supported by NSF grants CCF-1563155, CCF-1703925, IIS-1838154, CCF-2106429 and CCF-2107187.
Mihalis Yannakakis: Supported by NSF grants CCF-2107187 and CCF-2212233.

Acknowledgements We would like to thank anonymous CCC reviewers for their helpful comments to improve the presentation of the paper.

1 Introduction

We start with the definition of *monotone* functions and state Tarski's fixed point theorem [12]:

▶ **Definition 1** (Monotone functions). *Let $(\mathcal{L}, \preceq)$ be a complete lattice. A function $f : \mathcal{L} \to \mathcal{L}$ is said to be monotone if $f(a) \preceq f(b)$ for all $a, b \in \mathcal{L}$ with $a \preceq b$.*

▶ **Theorem 2** (Tarski). *For any complete lattice $(\mathcal{L}, \preceq)$ and any monotone function $f : \mathcal{L} \to \mathcal{L}$, there must be a point $x \in \mathcal{L}$ such that $f(x) = x$, i.e., x is a fixed point. In fact, the fixed points form a sublattice, with a greatest and a smallest element.*

Tarski's fixed point theorem has extensive applications in many fields, including for example verification, semantics, game theory and economics. For example in game theory there is an important class of games, called supermodular games (or games with strategic complementarities) which model economic settings where a player's best response is a monotone function (or correspondence) of the other players' actions [13, 14, 10]. These games always have pure equilibria (in fact a lattice of pure equilibria) by Tarski's theorem. Computing a pure equilibrium in such a game corresponds to finding a Tarski fixed point. In fact, as shown in [4], finding a pure equilibrium in supermodular games is essentially equivalent to finding a fixed point of monotone functions.

COMPUTATIONAL COMPLEXITY CONFERENCE

There are several other types of games that reduce to the Tarski problem. For example, Condon's simple stochastic games [2] have been intensely studied both in theoretical computer science as well as in the verification field (and they subsume other well-studied problems, such as parity games); their complexity remains a notorious open problem. The problem can be reduced to the Tarski problem of finding a fixed point of a given monotone function f, and in fact in this case we can even guarantee that the function has a unique fixed point. A similar property holds for the broader class of stochastic games defined originally by Shapley [11], and studied extensively since then. These games have in general irrational solutions, but it can be shown again that approximating the solution to any desired accuracy reduces to the problem of computing a fixed point of a monotone function that is furthermore guaranteed to have a unique fixed point (see [4] for more details). More generally, uniqueness of solutions is a desirable property in many applications in game theory, economics, and other fields. For sufficient conditions that ensure the uniqueness of Tarski fixed points, see [9] and references therein.

Thus, these facts raise the question, how hard is it to find a fixed point of a given monotone function? And if we know that the function has a unique fixed point, does this make the problem easier?

In this paper, we study the deterministic query complexity of finding a fixed point of a monotone function f over the complete lattice of a k-dimensional grid $([n]^k, \preceq)$, where $[n]$ denotes $\{1, \ldots, n\}$ and $\preceq$ denotes the natural partial order over $\mathbb{Z}^k$: $a \preceq b$ if and only if $a_i \leq b_i$ for every $i \in [k]$. So a monotone function $f : [n]^k \to [n]^k$ satisfies $f(a) \preceq f(b)$ for all $a, b \in [n]^k$ with $a \preceq b$, and we write $\texttt{Fix}(f)$ to denote the set of fixed points x of f satisfying $f(x) = x$. In the applications, n is typically exponential in the input size and k is polynomial. Thus, polynomial complexity in this context means polynomial in $\log n$ and k. Under the query model, an algorithm has oracle access to an unknown monotone function $f : [n]^k \to [n]^k$. In each round, it can send a query $x \in [n]^k$ to the oracle to reveal $f(x)$, and it succeeds after making a query x that returns $f(x) = x$. We write $\textsc{Tarski}(n, k)$ to denote this problem.

Our understanding of the query complexity of $\textsc{Tarski}(n, k)$ remains rather limited. On the upper bound side, there are two basic algorithms. Tarski's algorithm (or Kleene iteration in a different literature) starts from the bottom element 1^k of the lattice (or the top element n^k) and applies repeatedly f until it reaches a fixed point; the query complexity is $\Theta(nk)$ in the worst case. Another algorithm by [3] applies a binary search strategy in a recursive way and has query complexity $O(\log^k n)$. More recently, [7] gave an algorithm for $\textsc{Tarski}(n, k)$ with $O(\log^{\lceil 2k/3 \rceil} n)$ queries, which was further improved to $O(\log^{\lceil (k+1)/2 \rceil} n)$ in [1]. Both algorithms of [7] and [1] are based on decomposition theorems that lead to more efficient recursive schemes for Tarski fixed points.

On the lower bound side, [4] showed that $\textsc{Tarski}(n, 2)$ requires $\Omega(\log^2 n)$ queries. Their lower bound uses the family of "herringbone" functions which have a unique fixed point. Therefore, the same $\Omega(\log^2 n)$ lower bound also holds for the unique Tarski fixed point problem over $[n]^2$, where the input function is not only monotone but also promised to have a unique fixed point. Let $\textsc{UniqueTarski}(n, k)$ denote the unique Tarski problem over $[n]^k$. Given that $\textsc{UniqueTarski}(n, 2)$ is as hard as $\textsc{Tarski}(n, 2)$, is it the case for general k? or maybe $\textsc{UniqueTarski}(n, k)$ is easier than $\textsc{Tarski}(n, k)$ for larger k? This was posed as an open question in [4].

Our main result is a black-box reduction from $\textsc{Tarski}(n, k)$ to $\textsc{UniqueTarski}(n, k)$, which shows that the phenomenon observed in [4] between query complexities of $\textsc{Tarski}(n, 2)$ and $\textsc{UniqueTarski}(n, 2)$ holds for general k.

▶ **Theorem 3.** *Let $q^{\mathrm{T}}(n,k)$ be the query complexity of* Tarski(n,k) *and $q^{\mathrm{UT}}(n,k)$ be the query complexity of* UniqueTarski(n,k)*. Then $q^{\mathrm{T}}(n,k) = q^{\mathrm{UT}}(n,k)$.*

Remark. In fact, we will show that the query complexity of Tarski(n,k) is exactly the same as that of a seemingly even easier (more structural) problem: finding a fixed point of a monotone function over $[n]^k$ with the promise that *every slice has a unique fixed point*. See Lemma 16 for more details.

Note that the query complexity of UniqueTarski(n,k) is trivially at most that of Tarski(n,k). In the rest of the paper, we prove the other direction by giving a reduction from Tarski(n,k) to UniqueTarski(n,k) via a novel framework we call *partial information reductions*. We believe that this framework is of independent interest and we expect that it can be applied to a wider range of search problems concerning their query complexities.

1.1 Sketch of the Reduction

Unlike standard reductions that map from instances to instances, our reduction transforms any given algorithm for UniqueTarski (denoted by $\mathcal{U}$) to an algorithm for Tarski (our main algorithm, Algorithm 1) while keeping the query complexity the same. To the best of our knowledge, we have not seen such a non-standard black-box reduction before, and we view this as a conceptual contribution of this work. We would like to highlight the following high-level roadmap: Algorithm 1 will simulate $\mathcal{U}$, but provide it with modified answers to its queries to the oracle. These modified answers are constructed adaptively on-the-fly, and depend on what previous queries $\mathcal{U}$ has made. It may seem dangerous to modify the answers to the queries in the first place, but our reduction makes sure that the answers fed to $\mathcal{U}$ are always safe, in the sense that they always correspond to some monotone function with a unique fixed point, and any fixed point that is found by $\mathcal{U}$ must also be a fixed point of the original monotone function. Let's explain the reduction in more detail next.

Let $\mathcal{U}$ be a deterministic query algorithm for UniqueTarski(n,k) with query complexity $q(n,k)$. Given any monotone function $g : [n]^k \to [n]^k$ that has a unique fixed point x^*, $\mathcal{U}$ always finds x^* by querying it within the first $q(n,k)$ queries. At a high level, we would like to simulate $\mathcal{U}$ to find a fixed point of *any* monotone function $f : [n]^k \to [n]^k$ as an input to Tarski(n,k). But clearly we cannot run $\mathcal{U}$ on f directly since the latter may have multiple fixed points and it is likely that after some queries, answers that $\mathcal{U}$ receives are not consistent with any monotone function with a unique fixed point, in which case $\mathcal{U}$ may fail to find a fixed point within $q(n,k)$ queries. (See Figure 2 in Section 4 for an example.)

Instead, our reduction needs to serve as a surrogate between f and $\mathcal{U}$ to achieve the following two goals that are seemingly contradictory to each other:

- **(i)** On the one hand, we need to fool $\mathcal{U}$ by making sure that answers it receives during the whole process are consistent with some monotone function that has a unique fixed point.
  So from $\mathcal{U}$'s point of view, the function it interacts with can totally be a monotone function with a unique fixed point. Let's refer to this function, which is made up by our reduction, by g. Given that we cannot always return $f(x)$ to each query x of $\mathcal{U}$, the true input function f can potentially disagree significantly with the fake function g that $\mathcal{U}$ interacts with (see the comparison of Figure 2b and 4d);
- **(ii)** On the other hand, the way we answer queries to $\mathcal{U}$ (or the way we make up this fake function g) needs to achieve that, whenever $\mathcal{U}$ finds a fixed point of the fake function g (which always happens within $q(n,k)$ queries if the first goal is met), the same point must be a fixed point of the true input function f as well.

We achieve these goals using *partial information functions* (or PI functions in short).

A PI function p over $[n]^k$ is a map from $[n]^k$ to $\{-1, 0, 1, \leq, \geq, \diamond\}^k$. Intuitively a PI function p reveals some partial information of an unknown function $h : [n]^k \to [n]^k$. (For example, $p(x)_i = 1$ implies that $h(x)_i > x_i$, $p(x)_i = \geq$ implies that $h(x)_i \geq x_i$ and $p(x)_i = \diamond$ implies no information about $h(x)_i$; the connection will become cleaner after we introduce the notion of simple functions at the beginning of Section 2.) Moreover, we say a PI function p is *monotone* if it reveals some partial information of an unknown monotone function so one should not be able to infer from p any violation to monotonicity; see Definition 7.

Let $f : [n]^k \to [n]^k$ be the input monotone function. Our main algorithm, Algorithm 1 solves $\textsc{Tarski}(n, k)$ on f by simulating $\mathcal{U}$ round by round as follows: During the t-th round, $t = 1, 2, \ldots$,

1. Algorithm 1 runs $\mathcal{U}$ to obtain the t-th point $q^t \in [n]^k$ that $\mathcal{U}$ would like to query;
2. Algorithm 1 queries f to obtain $f(q^t)$ and uses it to obtain the answer $a^t \in [n]^k$ to the query. (As discussed earlier, a^t is not necessarily the same as $f(q^t)$; picking a^t based on $f(q^t)$ and the query history is the part that heavily relies on the use of PI functions.)
3. Finally Algorithm 1 sends a^t to $\mathcal{U}$ as the result of its t-th query, and moves onto round $t + 1$ (unless $f(q^t) = q^t$ so a fixed point of f has already been found).

Algorithm 1 picks answers a^t to queries of $\mathcal{U}$ by maintaining a monotone PI function p to connect f with $\mathcal{U}$. After receiving the t-th query q^t from $\mathcal{U}$, Algorithm 1 uses $f(q^t)$ to update the current PI function and then uses the updated PI function to set the answer a^t to $\mathcal{U}$. The design of the updating rule for the PI function (see the main subroutine `Generate-PI-Function` in Section 3) to achieve both goals (i) and (ii) discussed earlier is the most challenging part of the paper.

2 Partial-Information Functions

For $a, b \in \mathbb{Z}^k$ with $a \preceq b$, we write $\mathcal{L}_{a,b}$ to denote the set of points $x \in \mathbb{Z}^k$ with $a \preceq x \preceq b$.

We say a function $f : [n]^k \to [n]^k$ is a *simple* function if it satisfies $|f(x)_i - x_i| \leq 1$ for all $x \in [n]^k$ and $i \in [k]$ (i.e., $f(x)_i - x_i \in \{-1, 0, 1\}$). Let $\text{sgn}(a)$ for a number a be $1, 0, -1$ respectively if $a > 0, a = 0, a < 0$. We include the following folklore observations:

▶ **Observation 4.** *For any monotone function $f : [n]^k \to [n]^k$, let $g : [n]^k \to [n]^k$ be defined as*

$$g(x)_i \coloneqq x_i + \text{sgn}(f(x)_i - x_i), \quad \text{for all } x \in [n]^k \text{ and } i \in [k].$$

Then g is a monotone simple function and satisfies $\texttt{Fix}(g) = \texttt{Fix}(f)$.

It follows that for both Tarski and UniqueTarski, we may assume without loss of generality that the input monotone function $f : [n]^k \to [n]^k$ is simple.

▶ **Observation 5.** *A simple function $f : [n]^k \to [n]^k$ is monotone if and only if it satisfies the following conditions:*

(1) *$f(x)_i = x_i + 1$ implies $f(y)_i = y_i + 1$ and $f(y + e_i)_i \geq y_i + 1$ for all y with $x \preceq y$ and $x_i = y_i$;*

(2) *$f(x)_i = x_i - 1$ implies $f(y)_i = y_i - 1$ and $f(y - e_i)_i \leq y_i - 1$ for all y with $x \succeq y$ and $x_i = y_i$; and*

(3) *$f(x)_i = x_i$ implies (a) $f(y)_i \leq y_i$ for all y with $x \succeq y$ and $x_i = y_i$, and (b) $f(y)_i \geq y_i$ for all y with $x \preceq y$ and $x_i = y_i$.*

Observation 5 provides an alternative way to check the monotonicity of a simple function. It will mainly serve to verify the monotonicity of the following introduced *partial-information* functions. All functions from $[n]^k \to [n]^k$ we deal with from now on are assumed to be simple; for convenience, we will skip the word "simple" in the rest of the paper.

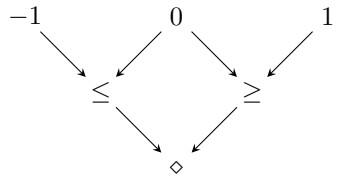


Figure 1 The information partial order. Arrow means "dominates" or "more informative".

Now we define *partial-information* (PI) functions. A PI function over $[n]^k$ is a function from $[n]^k$ to $\{-1, 0, 1, \leq, \geq, \diamond\}^k$. Intuitively a PI function reveals some partial information on the values of an underlying function $f : [n]^k \to [n]^k$; the next definition illustrates meanings of symbols in $\{-1, 0, 1, \leq, \geq, \diamond\}$:

▶ **Definition 6** (Consistency). *A function $g : [n]^k \to [n]^k$ and a PI function $p : [n]^k \to \{-1, 0, 1, \leq, \geq, \diamond\}^k$ are* consistent *if the following conditions hold for all $x \in [n]^k$ and $i \in [k]$:*

- *$p(x)_i = -1$ implies $g(x)_i - x_i = -1$;*
- *$p(x)_i = 0$ implies $g(x)_i - x_i = 0$;*
- *$p(x)_i = 1$ implies $g(x)_i - x_i = 1$;*
- *$p(x)_i = \leq$ implies $g(x)_i - x_i \in \{-1, 0\}$;*
- *$p(x)_i = \geq$ implies $g(x)_i - x_i \in \{0, 1\}$; and*
- *$p(x)_i = \diamond$ implies nothing about $g(x)_i$.*

We introduce a natural partial order over symbols in $\{-1, 0, 1, \leq, \geq, \diamond\}$, illustrated in Figure 1. We say α *dominates* β (or α is more informative than β, denoted by $\alpha \Rightarrow \beta$), for some $\alpha, \beta \in \{-1, 0, 1, \leq, \geq, \diamond\}$, if either $\alpha = \beta$ or there is a path from α to β. With this notation, we have that $g : [n]^k \to [n]^k$ is consistent with a PI function p iff $g(x)_i - x_i \Rightarrow p(x)_i$ for all $x \in [n]^k$ and $i \in [k]$. Given two PI functions $p', p : [n]^k \to \{-1, 0, 1, \leq, \geq, \diamond\}^k$, we say p' *dominates* p (or p' is more informative than p, denoted by $p' \Rightarrow p$) if $p'(x)_i \Rightarrow p(x)_i$ for all $x \in [n]^k$ and $i \in [k]$.

Given that we are interested in monotone functions $f : [n]^k \to [n]^k$, we introduce the notion of *monotone* PI functions below. Intuitively a PI function p is monotone if it reveals some partial information of a monotone function (so one cannot infer from p any violation to monotonicity):

▶ **Definition 7** (Monotone PI Functions). *A PI function $p : [n]^k \to \{-1, 0, 1, \leq, \geq, \diamond\}^k$ is said to be* monotone *if it satisfies the following conditions: For any $x \in [n]^k$ and $i \in [k]$,*

(1) *$p(x)_i = 1$ implies $p(y)_i = 1$ and $p(y + e_i)_i \in \{1, 0, \geq\}$ for all y with $x \preceq y$ and $x_i = y_i$;*
(2) *$p(x)_i = -1$ implies $p(y)_i = -1$ and $p(y - e_i)_i \in \{-1, 0, \leq\}$ for all y with $x \succeq y$ and $x_i = y_i$;*
(3) *$p(x)_i = 0$ implies (a) $p(y)_i \in \{0, -1, \leq\}$ for all y with $x \succeq y$ and $x_i = y_i$, and (b) $p(y)_i \in \{0, 1, \geq\}$ for all y with $x \preceq y$ and $x_i = y_i$;*
(4) *$p(x)_i = \leq$ implies $p(y)_i \in \{-1, \leq\}$ for all y with $x \succeq y$ and $x_i = y_i$;*
(5) *$p(x)_i = \geq$ implies $p(y)_i \in \{1, \geq\}$ for all y with $x \preceq y$ and $x_i = y_i$;*
(6) *If $x_i = 1$, then $p(x)_i \in \{0, 1, \geq\}$; and*
(7) *If $x_i = n$, then $p(x)_i \in \{0, -1, \leq\}$.*

A PI function is weakly monotone *if it satisfies (1)–(5) above, but not necessarily (6) and (7).*

Note that items (6) and (7) are only about the boundary constraints. Weak monotonicity will only appear for the simplicity of the proofs below and there are no technical details behind them.

The next lemma shows that every monotone PI function is consistent with at least one monotone function. (Looking ahead, later in Section 3.2 we will give a sufficient condition for a monotone PI function to be consistent with at least one monotone function with a unique fixed point.)

▶ **Lemma 8.** *For every monotone PI function p over $[n]^k$, there exists a monotone function $g : [n]^k \to [n]^k$ that is consistent with p.*

Proof. Given p we define $g : [n]^k \to [n]^k$ as follows:

$$g(x)_i \coloneqq \begin{cases} x_i + p(x)_i & \text{if } p(x)_i \in \{-1, 0, 1\}; \\ x_i & \text{otherwise.} \end{cases}$$

We will prove g is a monotone function that is consistent with p by Observation 5.

Fix any point x and coordinate i.

Suppose that $g(x)_i = x_i + 1$, then we have $p(x)_i = 1$. Since $p(x)_i = 1$ implies $p(y)_i = 1$ and $p(y + e_i)_i \in \{1, 0, \geq\}$ for all y such that $x \preceq y$ and $x_i = y_i$, we have $g(x)_i = x_i + 1$ implies $g(y)_i = y_i + 1$ and $g(y + e_i)_i \geq y_i + 1$ for all y such that $x \preceq y$ and $x_i = y_i$.

The proof of the case that $g(x)_i = x_i - 1$ is symmetric.

Suppose that $g(x)_i = x_i$, then we have $p(x)_i \in \{0, \leq, \geq, \diamond\}$. This implies that (a) $p(y)_i \neq 1$ for all y such that $x \succeq y$ and $x_i = y_i$, and (b) $p(y)_i \neq -1$ for all y such that $x \preceq y$ and $x_i = y_i$. So we have (a) $g(y)_i \leq y_i$ for all y such that $x \succeq y$ and $x_i = y_i$, and (b) $g(y)_i \geq y_i$ for all y such that $x \preceq y$ and $x_i = y_i$. ◀

Given two elements $\alpha, \beta \in \{-1, 0, 1, \leq, \geq, \diamond\}$, if their least upper bound (or join) in the partial order exists, we write $\alpha \cap \beta$ to denote it and say that $\alpha \cap \beta$ is well defined; otherwise (when their least upper bound does not exist), we say $\alpha \cap \beta$ is not well defined. (For example, $\geq \cap \leq = 0$ and $\geq \cap -1$ is not well defined.) Given two PI functions $p^1, p^2 : [n]^k \to \{-1, 0, 1, \leq, \geq, \diamond\}^k$, we define their *intersection* $p^1 \cap p^2$ to be the PI function p such that $p(x)_i = p^1(x)_i \cap p^2(x)_i$ for all $x \in [n]^k$ and $i \in [k]$. The intersection $p^1 \cap p^2$ is well defined only when $p^1(x)_i \cap p^2(x)_i$ is well defined for all $x \in [n]^k$ and $i \in [k]$.

The reason that we introduce the operation of intersections is the following lemma which we will often use to modify a given monotone PI function:

▶ **Lemma 9.** *Let p^1 be a monotone PI function and p^2 be a weakly monotone PI function, both over $[n]^k$, such that $p^1 \cap p^2$ is well defined. Then $p^1 \cap p^2$ is also a monotone PI function and it satisfies $p^1 \cap p^2 \Rightarrow p^1$.*

Proof. The part about $p^1 \cap p^2 \Rightarrow p^1$ is trivial.

Note that $p^1 \cap p^2$ satisfies (6) and (7) in Definition 7 since p^1 is a monotone PI function. So in what follows, we will verify (1)-(5) for $p^1 \cap p^2$.

To show $p^1 \cap p^2$ satisfies (1), fix $x \in [n]^k$ and $i \in [k]$. If $p^1 \cap p^2(x)_i = 1$, then either $p^1(x)_i = 1$ or $p^2(x)_i = 1$. Suppose that $p^\tau(x)_i = 1$ for $\tau \in \{1, 2\}$. Then we have $p^\tau(y)_i = 1$ and $p^\tau(y + e_i)_i \in \{1, \geq\}$ for all $y \succeq x$ and $y_i = x_i$. So we have $p^1 \cap p^2(y)_i = 1$ and $p^1 \cap p^2(y + e_i)_i \in \{1, \geq\}$ for all $y \succeq x$ and $y_i = x_i$.

Items (2)-(5) can be verified similarly. ◀

Given a monotone function $f : [n]^k \to [n]^k$ and a monotone PI function p over $[n]^k$, we define a function $f|_p : [n]^k \to [n]^k$ as follows: For any $x \in [n]^k$ and $i \in [k]$, let

$$f|_p(x)_i = \begin{cases} x_i + p(x)_i & \text{if } p(x)_i \in \{-1, 0, 1\}; \\ \max(f(x)_i, x_i) & \text{if } p(x)_i = \geq; \\ \min(f(x)_i, x_i) & \text{if } p(x)_i = \leq; \\ f(x)_i, & \text{if } p(x)_i = \diamond. \end{cases}$$

Note that $f|_p$ is a function that is consistent with p (but may disagree with f in general). Looking ahead, our algorithm for TARSKI running on f will maintain a monotone PI function p and (essentially) use $f|_p$ to answer the next query from an algorithm for UNIQUETARSKI it simulates. As it will become clear in Section 3, using $f|_p$ (with a carefully updated p) instead of f to answer queries is crucial in making sure answers to the algorithm for UNIQUETARSKI are consistent with a monotone function with a unique fixed point (given that the input function f to TARSKI can have multiple fixed points in general).

We record the following lemma about $f|_p$:

▶ **Lemma 10.** *Let f be a monotone function and p be a monotone PI function, both over $[n]^k$. Then $f|_p : [n]^k \to [n]^k$ is also a monotone function and is consistent with p.*

Proof. The part about $f|_p$ being consistent with p is easy, since $f|_p(x)_i - x_i = p(x)_i$ when $p(x)_i \in \{-1, 0, 1\}$; $f|_p(x)_i - x_i \in \{0, 1\}$ when $p(x)_i = \geq$; $f|_p(x)_i - x_i \in \{0, -1\}$ when $p(x)_i = \leq$; and $f|_p(x)_i - x_i \in \{-1, 0, 1\}$ when $p(x)_i = \diamond$.

Note that since f is a function from $[n]^k$ to $[n]^k$ and p is a monotone PI function that satisfies the boundary conditions (6) and (7), we have $1 \leq f|_p(x)_i \leq n$ for all x and i.

To show $f|_p$ is monotone, given any $x \in [n]^k$ and $i \in [k]$, we consider three cases where $f|_p(x)_i = x_i + 1$, $f|_p(x)_i = x_i - 1$, and $f|_p(x)_i = x_i$.

Suppose that $f|_p(x)_i = x_i + 1$. Then either $f(x)_i - x_i = 1$ or $p(x)_i = 1$. If $f(x)_i - x_i = 1$ and $p(x)_i \in \{1, \geq, \diamond\}$, then we have (i) $f(y)_i - y_i = 1$ and $f(y + e_i)_i \geq y_i + 1$, and (ii) $p(y)_i \in \{1, \geq, \diamond\}$ and $p(y + e_i)_i \neq -1$ for all $y \succeq x$ with $y_i = x_i$, which imply $f|_p(y)_i - y_i = 1$ and $f(y + e_i)_i \geq y_i + 1$. The proof is similar in the case $p(x)_i = 1$.

The proof of the case $f|_p(x)_i = x_i - 1$ is symmetric.

Suppose that $f|_p(x)_i = x_i$. Then one of the following four cases meets: (1) $f(x)_i = x_i$ and $p(x)_i \in \{\leq, \geq, \diamond\}$; (2) $p(x)_i = 0$; (3) $f(x)_i - x_i = 1$ and $p(x)_i = \leq$; (4) $f(x)_i - x_i = -1$ and $p(x)_i = \geq$. Let's prove the first case and others are similar. Suppose that $f(x)_i = x_i$ and $p(x)_i \in \{\leq, \geq, \diamond\}$, then we have $f(y)_i \leq y_i$ for all $y \preceq x$ with $y_i = x_i$ and $f(y)_i \geq y_i$ for all $y \succeq x$ with $y_i = x_i$. Since $p(x)_i \in \{\leq, \geq, \diamond\}$, we have $p(y)_i \neq 1$ for all $y \preceq x$ with $y_i = x_i$ and $p(y)_i \neq -1$ for all $y \succeq x$ with $y_i = x_i$. This finishes the proof. ◀

Before moving to the main reduction, we need to introduce the notion of slices. We note that the notion of slices was also used in the literature.

▶ **Definition 11** (Slices). *A* slice *of $[n]^k$ is specified by a tuple $s \in ([n] \cup \{*\})^k$. Given s, we write $\mathcal{L}_s$ to denote the set of points x such that $x_i = s_i$ for all i such that $s_i \neq *$.*

Given a monotone PI function p and a slice s, we say a point $x \in \mathcal{L}_s$ is a *postfixed* point of p on the slice s if $p(x)_i \in \{1, 0, \geq\}$ for all i with $s_i = *$ and a point $x \in \mathcal{L}_s$ is a *prefixed* point of p on the slice s if $p(x)_i \in \{-1, 0, \leq\}$ for all i with $s_i = *$.

We use $\texttt{Post}_s(p)$ to denote the set of postfixed points of p on s and $\texttt{Pre}_s(p)$ to denote the set of prefixed points of p on s.

▶ **Lemma 12.** *Given a monotone PI function p, for any slice s, $\texttt{Post}_s(p)$ is a join-semilattice and $\texttt{Pre}_s(p)$ is a meet-semilattice.*

Proof. Fix a slice s and consider any two points $x, y \in \texttt{Post}_s(p)$. Then we have $p(x)_i, p(y)_i \in \{1, 0, \geq\}$ for all i with $s_i = *$. Let $z = x \vee y$ be the join of x and y, namely, the coordinatewise maximum of x and y. Then we have $x \preceq z$ and $y \preceq z$ and either $z_i = x_i$ or $z_i = y_i$ for all i with $s_i = *$. So by the monotonicity of p, we have $p(z)_i \in \{1, 0, \geq\}$.

The proof of that $\texttt{Pre}_s(p)$ is a meet-semilattice is similar. This finishes the proof. ◀

Lemma 12 guarantees that the join of $\texttt{Post}_s(p)$ is well defined and the meet of $\texttt{Pre}_s(p)$ is well defined. We write $J_s(p)$ to denote the join of $\texttt{Post}_s(p)$ and $M_s(p)$ to denote the meet of $\texttt{Pre}_s(p)$. When the context is clear, we may omit p for the simplicity of notations.

▶ **Proposition 13.** *Given a monotone PI function p, for any slice s, we have $p(J_s)_i \in \{0, \geq\}$ for all i with $s_i = *$ and $p(M_s)_i \in \{0, \leq\}$ for all i with $s_i = *$.*

Proof. Consider any point $x \in \texttt{Post}_s$. Suppose that there exists i with $s_i = *$ such that $p(x)_i = 1$, then we have $x + e_i \in \texttt{Post}_s$ as well. This means x can not be J_s. So we have $p(J_s)_i \in \{0, \geq\}$ for all i with $s_i = *$. The proof of $p(M_s)$ is similar. ◀

3 The Partial-Information Reduction and Proof of Theorem 3

We prove Theorem 3 in this section. Let $\mathcal{U}$ be any query algorithm for UniqueTarski(n, k) with query complexity $q(n, k)$. We show that our main algorithm, Algorithm 1, can employ $\mathcal{U}$ to solve Tarski(n, k) with the same number of queries.

Let's continue from the sketch presented in Section 1.1 and elaborate more on how Algorithm 1 works. Algorithm 1 computes the answer a^t to the t-th query q^t of $\mathcal{U}$ by maintaining a sequence of monotone PI functions $p^0, p^1, \ldots$, where p^0 is the initial monotone PI function set by the boundary conditions (see line 2 of Algorithm 1) and p^t is the monotone PI function it maintains at the end of the t-th round. During the t-th round, Algorithm 1 first continues to run $\mathcal{U}$ to obtain the t-th query q^t. It then queries f to obtain $f(q^t)$ and uses the latter to update the PI function p^{t-1} to p^t. Finally the answer a^t to $\mathcal{U}$ is set to be $f|_{p^t}(q^t) \in [n]^k$.

The correctness of Algorithm 1 relies on the following list of properties of p^t: For every t, p^t is a monotone PI function such that

1. $p^t(q^j) + q^j = a^j$ for all $j \in [t]$ (i.e., p^t agrees with answers to all queries $\mathcal{U}$ has made so far);
2. There is a monotone function g that is consistent with p^t and has a unique fixed point;
3. Any fixed point of $f|_{p^t}$ must be a fixed point of f.

To see that Algorithm 1 always finds a fixed point of f within $q(n, k)$ queries, we note that item 3 above implies that q^t is a fixed point of f if $a^t = f|_{p^t}(q^t)$ is the same as q^t. So the only bad case is that $a^t \neq q^t$ for all $t = 1, \ldots, q(n, k)$. However, this cannot happen because after $q(n, k)$ rounds, by item 2, there is a monotone function g that is consistent with $p^{q(n,k)}$ and has a unique fixed point, and by item 1, $g(q^j) = a^j$ for all $j \in [q(n, k)]$. So g is a monotone function that has a unique fixed point, on which $\mathcal{U}$ fails to find a fixed point (since $a^j \neq q^j$ for all $j \in [q(n, k)]$).

3.1 Subroutine `Generate-PI-Function`

The main challenge is about how to update the PI function p^{t-1} to p^t during the t-th round to maintain properties listed above for the correctness of the algorithm. This is done by making calls to a subroutine called `Generate-PI-Function` (see Algorithm 1; in general it may take k calls to `Generate-PI-Function` to obtain p^t during the t-th round).

The subroutine `Generate-PI-Function`(p, q, ℓ, b) takes four inputs, namely, a PI function p, a point $q \in [n]^k$, an index $\ell \in [k]$, and a sign $b \in \{-1, 0, 1\}$, and returns a new PI function. Before stating the main technical theorem about `Generate-PI-Function`, we need the following definition:

▶ **Definition 14.** *We say a monotone PI function p is* safe *if, for every slice $s \in ([n] \cup \{*\})^k$, it satisfies*

(1) *for any point* $x \in \mathcal{L}_s$ *and* $x \prec J_s(p)$, $p(x)_i \in \{-1,0,1\}$ *for all* i *with* $x_i < J_s(p)_i$ *and* $p(x)_i = 1$ *for some* i *with* $x_i < J_s(p)_i$*; and*
(2) *for any point* $x \in \mathcal{L}_s$ *and* $x \succ M_s(p)$, $p(x)_i \in \{-1,0,1\}$ *for all* i *with* $x_i > M_s(p)_i$ *and* $p(x)_i = -1$ *for some* i *with* $x_i > M_s(p)_i$.

We are now ready to state our main technical theorem:

▶ **Theorem 15** (Main Technical Theorem). *Given a monotone and safe PI function* p, $q \in [n]^k$, $\ell \in [k]$, *and* $b \in \{-1,0,1\}$ *such that* $(p(q)_\ell, b)$ *satisfies the following condition:*

$$p(q)_\ell \in \{\geq, \diamond\} \text{ if } b = 1;\ \ p(q)_\ell \in \{\leq, \diamond\} \text{ if } b = -1;\ \ p(q)_\ell \in \{\leq, \geq, \diamond\} \text{ if } b = 0, \tag{1}$$

the function p^r *returned by* `Generate-PI-Function`(p, q, ℓ, b) *satisfies the following properties:*
1. p^r *is also a monotone PI function;*
2. $p^r \Rightarrow p$*;*
3. $p^r(q)_\ell = b$*; and*
4. p^r *remains safe.*

Additionally, if $f : [n]^k \to [n]^k$ *is a monotone function such that* $\texttt{Fix}(f|_p) \subseteq \texttt{Fix}(f)$ *and* $f|_p(q)_\ell = q_\ell + b$*, then we have* $\texttt{Fix}(f|_{p^r}) \subseteq \texttt{Fix}(f|_p) \subseteq \texttt{Fix}(f)$.

We prove Theorem 15 in the rest of the section. An important property of safe, monotone PI functions is given in the following lemma which we prove in the next subsection.

▶ **Lemma 16.** *If* p *is a monotone and safe PI function, then there is a monotone function* g *that is consistent with* p *and has a unique fixed point in every slice* s*. In particular,* g *has a unique fixed point in the whole lattice.*

We can use Theorem 15 and Lemma 16 to prove the main theorem:

Proof of Theorem 3. Let f be the input function. We first note that every time Algorithm 1 obtains $p^{(t,i)}$ from $p^{(t,i-1)}$, either $p^{(t,i)}$ is the same as $p^{(t,i-1)}$ or $p^{(t,i)}$ is set to be

$$\texttt{Generate-PI-Function}\big(p^{(t,i-1)}, q, i, b\big)$$

for some q, i, b that satisfy $b = f|_{p^{(t,i-1)}}(q)_i - q_i$ and (1):

$$p^{(t,i-1)}(q)_i \in \{\geq, \diamond\} \text{ if } b = 1;\ \ p^{(t,i-1)}(q)_i \in \{\leq, \diamond\} \text{ if } b = -1;\ \ p^{(t,i-1)}(q)_i \in \{\leq, \geq, \diamond\} \text{ if } b = 0,$$

Given that $p^{(1,0)} = p^0$ is monotone and safe, it follows directly from an induction using Theorem 15 that every PI function p in the following list:

$$p^{(1,0)}, \ldots, p^{(1,k)}, p^{(2,0)}, \ldots, p^{(2,k)}, \ldots, p^{(t,0)}, \ldots, p^{(t,k)}, \ldots$$

is monotone and safe, and satisfies $\texttt{Fix}(f|_p) \subseteq \texttt{Fix}(f)$. Furthermore, every PI function p in the list dominates all of its predecessors and $p^{(t,i)}(q^t)_i \in \{-1,0,1\}$ for all t, i. Combining the latter with $a^t = f|_{p^t}(q^t)$, as well as that $p^t = p^{(t,k)}$ dominates all of its predecessors, we have $a^t - q^t = p^t(q^t)$. It follows that $a^j - q^j = p^t(q^j)$ for all $j \leq t$.

Let $N = q(n,k)$. Consider the following two cases:
1. If $a^t = q^t$ for some $t \in [N]$, then given that $a^t = f|_{p^t}(q^t)$ and $\texttt{Fix}(f|_{p^t}) \subseteq \texttt{Fix}(f)$, we have that q^t is a fixed point of f. In this case Algorithm 1 succeeds within $q(n,k)$ queries;
2. Otherwise, we have $a^t \neq q^t$ for all $t \in [N]$. In this case, given that p^N is both monotone and safe, Lemma 16 implies that there exists a monotone function g that is consistent with p^N and has a unique fixed point. However, given that $a^j - q^j = p^N(q^j)$ for all $j \leq N$, we have that $q^j \neq a^j = g(q^j)$ for all $j \in [N]$. As a result, $\mathcal{U}$ fails to find a fixed point of g within its N queries $q^1, \ldots, q^N$, which it should given that g is a monotone function with a unique fixed point, a contradiction.

This finishes the proof of the theorem. ◀

Algorithm 1 Algorithm for $\textsc{Tarski}(n,k)$ via the algorithm $\mathcal{U}$ for $\textsc{UniqueTarski}(n,k)$.

Let $\mathcal{U}$ be an algorithm for $\textsc{UniqueTarski}(n,k)$.
Let p^0 be an empty PI function with the initial boundary conditions, i.e., $p^0(x)_i = \geq$ if $x_i = 1$; $p^0(x)_i = \leq$ if $x_i = n$; and $p^0(x)_i = \diamond$ otherwise for all $x \in [n]^k$ and $i \in [k]$.
Let $t \leftarrow 1$ be the round number.
do
  Let q^t be the point queried by $\mathcal{U}$ and make one query to get $f(q^t)$.
  Let $p^{(t,0)} \leftarrow p^{t-1}$.
  for *each* i *from* 1 *to* k **do**
    If $p^{(t,i-1)}(q^t)_i \in \{-1,0,1\}$, let $p^{(t,i)} \leftarrow p^{(t,i-1)}$.
    Otherwise, let $p^{(t,i)} \leftarrow \texttt{Generate-PI-Function}(p^{(t,i-1)}, q^t, i, f|_{p^{(t,i-1)}}(q^t)_i - q^t_i)$.
  Let $p^t \leftarrow p^{(t,k)}$ and use $a^t \leftarrow f|_{p^t}(q^t)$ as the answer to the algorithm $\mathcal{U}$.
  If $q^t = a^t$, then **return** q^t as the fixed point and **terminate**.
  Else, let $t \leftarrow t+1$.
while;

Subroutine 2 `Generate-PI-Function`(p,q,ℓ,b).

If $b = 1$, then **return** `Generate-PI-Function-Plus`(p,q,ℓ).
If $b = -1$, then **return** `Generate-PI-Function-Minus`(p,q,ℓ).
If $b = 0$, then **return** `Generate-PI-Function-Zero`(p,q,ℓ).

Subroutine 3 `Generate-PI-Function-Plus`(p,q,ℓ).

Initialize $p' \leftarrow p$.
Let $p'(x)_\ell \leftarrow 1$ and $p'(x+e_\ell)_\ell \leftarrow p'(x+e_\ell)_\ell \cap \geq$ for all x such that $x \succeq q$ and $x_\ell = q_\ell$.
Initialize $p^+(x)_i \leftarrow \diamond$ for all x and i as a weak PI function.
for *each* $x \in [n]^k$ *and each* $i \in [k]$ **do**
  if *there exists* y *such that (a)* $x \preceq y$*; (b)* $x_i < y_i$*; and (c)* $x_j = y_j$ *for all* j *with* $p'(y)_j \notin \{1,0,\geq\}$ **then**
    If $p'(x)_i \in \{1,\geq,\diamond\}$, let $p^+(x)_i \leftarrow 1$ and $p^+(x+e_i)_i \leftarrow p^+(x+e_i)_i \cap \geq$.
    If $p'(x)_i \in \{0,\leq\}$, let $p^+(x)_i \leftarrow \geq$.
Let $p^r \leftarrow p' \cap p^+$.
return p^r.

3.2 Consequences of PI Function Being Safe

The motivation to focus on Definition 14 is that they have nice properties given in the following lemmas.

▶ **Lemma 17.** *Suppose that a PI function p is monotone and safe, then we have*
1. *$J_s(p) \preceq M_s(p)$ for all slices s; and*
2. *$g(x) \neq x$ for any monotone function g that is consistent with p and any x such that there exists s with $x \notin \mathcal{L}_{J_s,M_s}$.*

Proof. We will prove the following claim, by which we will deduce this lemma.

Subroutine 4 `Generate-PI-Function-Minus`(p, q, ℓ).

Initialize $p' \leftarrow p$.
Let $p'(x)_\ell \leftarrow -1$ and $p'(x - e_\ell)_\ell \leftarrow p'(x - e_\ell)_\ell \cap \leq$ for all x such that $x \preceq q$ and $x_\ell = q_\ell$.
Initialize $p^-(x)_i \leftarrow \diamond$ for all x and i as a weak PI function.
for *each* $x \in [n]^k$ *and each* $i \in [k]$ **do**
  if *there exists* y *such that (a)* $x \succeq y$*; (b)* $x_i > y_i$*; and (c)* $x_j = y_j$ *for all* j *with* $p'(y)_j \notin \{-1, 0, \leq\}$ **then**
    If $p'(x)_i \in \{-1, \leq, \diamond\}$, let $p^-(x)_i \leftarrow -1$ and $p^-(x - e_i)_i \leftarrow p^-(x - e_i)_i \cap \leq$.
    If $p'(x)_i \in \{0, \geq\}$, let $p^-(x)_i \leftarrow \leq$.
Let $p^r \leftarrow p' \cap p^-$.
return p^r.

Subroutine 5 `Generate-PI-Function-Zero`(p, q, ℓ).

Initialize $p^r \leftarrow p$.
If $q_\ell > 1$ and $p(q - e_\ell)_\ell \neq 1$, let $p^r \leftarrow$ `Generate-PI-Function-Plus`$(p^r, q - e_\ell, \ell)$.
If $q_\ell < n$ and $p(q + e_\ell)_\ell \neq -1$, let $p^r \leftarrow$ `Generate-PI-Function-Minus`$(p^r, q + e_\ell, \ell)$.
return p^r.

▷ Claim 18. Given the PI function p is monotone and safe, we have
(a) for any point $x \in \mathcal{L}_s$ and $x \not\preceq M_s(p)$, there exists i with $s_i = *$ and $p(x)_i = -1$; and
(b) for any point $x \in \mathcal{L}_s$ and $x \not\succeq J_s(p)$, there exists i with $s_i = *$ and $p(x)_i = 1$.

Proof. We will show that the first item in Definition 14 implies item (b), and the second item in Definition 14 implies item (a). Fix a slice s, a point $x \in \mathcal{L}_s$ and $x \not\succeq J_s(p)$. We will prove there exists i with $s_i = *$ and $p(x)_i = 1$. The proof for the item (a) is similar.

Construct a sub-slice s' as follows:

$$s'_i := \begin{cases} s_i & s_i \neq *; \\ x_i & s_i = * \text{ and } x_i \geq J_s(p^r)_i; \\ * & \text{otherwise } (s_i = * \text{ and } x_i < J_s(p^r)_i). \end{cases}$$

Then we have $s'_i = *$ implies $s_i = *$ and $x \in \mathcal{L}_{s'}$. Let z be the join of x and $J_s(p)$. Note that $z \in \mathcal{L}_{s'}$ as well. In addition, we have $x_i < z_i$ for all i with $s'_i = *$.

We will prove that $z \in \mathtt{Post}_{s'}(p)$, so we will have $z \preceq J_{s'}(p)$, which implies $x_i < J_{s'}(p)_i$ for all i with s'_i. Since p is safe, we conclude that there exists i with $s'_i = *$ and $p(x)_i = 1$. Such an i also satisfies $s_i = *$.

The statement $z \in \mathtt{Post}_{s'}(p)$ follows from the observation that whenever $s'_i = *$, we have $s_i = *$ and $z_i = J_s(p)_i$. Since $p(J_s)_i \in \{\geq, 0\}$, we have $p(z)_i \in \{1, 0, \geq\}$. ◁

We show that each of items (a) and (b) is strong enough to deduce the first item ($J_s(p) \preceq M_s(p)$ for all s). (This will be used in the proof of Lemma 22 below). Take item (b) as an example: Given any point $x \in \mathcal{L}_s$ such that $x \not\preceq M_s(p)$, since there exists i with $s_i = *$ and $p(x)_i = -1$, we have $x \notin \mathtt{Pre}_s(p)$ by definition. So $J_s(p)$ must be somewhere that is $\preceq M_s(p)$.

For the second item, consider any point x such that there exists s with $x \notin \mathcal{L}_{J_s, M_s}$. Then we know either $x \not\preceq M_s(p)$ or $x \not\succeq J_s(p)$. Since there exists i with $p(x)_i = -1$ or $p(x)_i = 1$, we have $g(x) \neq x$ as long as g is a monotone function that is consistent with p. ◀

We also present the proof of Lemma 16 in this subsection.

Proof of Lemma 16. We will refine p to a more informative monotone PI function p' such that every monotone function that is consistent with p' has in each slice s only one fixed point, $M_s(p)$.

Consider a slice s, and let $M_s = M_s(p)$ be the lowest prefixed point of p in the slice. By Claim 18, for every point $x \in \mathcal{L}_s$, if $x \not\preceq M_s$, there exists i with $s_i = *$ and $p(x)_i = -1$. Consider a point $x \in \mathcal{L}_s$ where $x \preceq M_s$, $x \neq M_s$. If i is a coordinate with $s_i = *$ and $x_i = (M_s)_i$ then $p(x)_i \in \{0, -1, \leq\}$ since $p(M_s)_i \in \{0, -1, \leq\}$. Since M_s is the lowest prefixed point in $\mathcal{L}_s$, there is a coordinate i such that $s_i = *$ and $p(x)_i \notin \{0, -1, \leq\}$, therefore $p(x)_i \in \{1, \geq, \diamond\}$ and $x_i < (M_s)_i$.

Define p' as follows. Initialize $p'(x) = p(x)$ for all $x \in [n]^k$. For each slice s and every point $x \in \mathcal{L}_s$ where $x \preceq M_s$, $x \neq M_s$, and each coordinate i such that $s_i = *$ and $p(x)_i \in \{1, \geq, \diamond\}$, set $p'(x)_i = 1$, and for every $y \succeq x$ with $y_i = x_i$ set $p'(y)_i = 1$ and $p'(y+e_i)_i = p'(y+e_i)_i \cap \geq$. (Note that $p'(y + e_i)_i$ may be also updated due to other points x', including possibly being set to 1.)

We first claim that p' is a well defined PI function and dominates (is more informative than) p. Note that p' changes the value of $p(z)_i$ for some points z and some coordinates i by either setting the value to 1 or taking the join with $\geq$. Thus, to show the claim it suffices to show that (i) p' does not set the value to 1 for any point z and coordinate i such that $p(z)_i \in \{-1, 0, \leq\}$, and (ii) it does not take the join with $\geq$ for any z and i such that $p(z)_i = -1$. To see this, consider any slice s, a point $x \in \mathcal{L}_s$ with $x \preceq M_s$, $x \neq M_s$, and a coordinate i such that $s_i = *$ and $p(x)_i \in \{1, \geq, \diamond\}$. Since $p(x)_i \in \{1, \geq, \diamond\}$, the new value $p'(x)_i = 1$ dominates $p(x)_i$. Consider any other point $y \succeq x$ with $y_i = x_i$. If $p(y)_i$ was in $\{-1, 0, \leq\}$, then $p(x)_i$ would also be in $\{-1, 0, \leq\}$ by Definition 7. We infer therefore that $p(y)_i \in \{1, \geq, \diamond\}$, thus $p'(y)_i = 1$ dominates $p(y)_i$. Also, if $p(y + e_i)_i$ was -1 then $p(x)_i$ would be in $\{-1, 0, \leq\}$. We infer therefore that $p(y + e_i)_i \neq -1$, thus the join with $\geq$ exists, it dominates $p(y + e_i)_i$, and is not -1. We conclude that p' is well defined and dominates p.

We then claim that p' is monotone. Consider any slice s, a point $x \in \mathcal{L}_s$ with $x \preceq M_s$, $x \neq M_s$, and a coordinate i such that $s_i = *$ and $p(x)_i \in \{1, \geq, \diamond\}$. Then we know $p'(x)_i = 1$. Consider any other point $y \succeq x$ with $y_i = x_i$, we have $p(y)_i \in \{1, \geq, \diamond\}$, so $p'(y)_i$ would also be 1. Also, since $p(y + e_i)_i$ is not -1, we conclude that $p'(y + e_i)_i \Rightarrow p(y + e_i)_i \cap \geq$. Since $p'(y + e_i)_i$ is well defined, we infer therefore that $p'(y + e_i)_i \in \{\geq, 0, 1\}$. For other points x and coordinates i, we have $p'(x)_i = p(x)_i$ and $p'(x)_i$ satisfying Definition 7 follow from the monotonicity of p and $p' \Rightarrow p$. We conclude that p' is monotone.

The PI function p' has the property that for every slice s and for every point $x \in \mathcal{L}_s$ with $x \neq M_s$, there exists a coordinate i such that either $p'(x)_i = -1$ (this is the case if $x \not\preceq M_s$) or $p'(x)_i = 1$ (this is the case if $x \preceq M_s$). We conclude that any monotone function that is consistent with p' has only one fixed point in each slice s, namely, M_s. Since p' dominates p, any such monotone function is also consistent with p. In particular, there exists at least one such monotone function, as constructed in Lemma 8. ◀

3.3 Preserving Monotonicity and Safety

In this subsection, we will prove items (1)–(4) of Theorem 15.

▶ **Lemma 19** (Monotonicity Preserving of Subroutine 3). *Given a monotone PI function p, a point $q \in [n]^k$ and a coordinate $\ell \in [k]$ such that $p(q)_\ell \in \{\geq, \diamond\}$ (which implies $q_\ell < n$), we have the PI function p^r returned by* `Generate-PI-Function-Plus`(p, q, ℓ) *remains monotone. Furthermore, we have $p^r \Rightarrow p$.*

Proof. We start by proving the monotonicity of p' on line 2. Since $p(q)_\ell \in \{\geq, \diamond\}$, we have $p(x)_\ell \in \{1, \geq, \diamond\}$ and $p(x + e_\ell)_\ell \in \{0, 1, \leq, \geq, \diamond\}$ for all x such that $x \succeq q$ and $x_\ell = q_\ell$. So $p(x + e_\ell)_\ell \cap \geq$ is well defined. The monotonicity of p' follows from the observation that we changed $p'(q)_\ell \leftarrow 1$ and maintained the consequences it should imply. Clearly, $p' \Rightarrow p$.

After that, we will maintain a new function p^+ from line 3 to line 7. Note that we will update $p^r \leftarrow p' \cap p^+$ on line 8 and return it. So by Lemma 9, it suffices for us to prove p^+ is a weakly monotone PI function, and $p'(x)_i \cap p^+(x)_i$ is well defined for all x and i at the end of the **for** loop.

To this end, we will prove that, at the end of the **for** loop, item (1) in Definition 7 is true for every point x and coordinate i such that $p^+(x)_i = 1$; and item (5) in Definition 7 is true for every point x and coordinate i such that $p^+(x)_i = \geq$. Before getting into details, we first provide a clearer picture of the condition of **if** on line 5.

▷ Claim 20. Given a coordinate i and two points $x \preceq x'$ such that $x_i = x'_i$, if the **if** condition on line 5 is true for x and i, then the **if** condition on line 5 is also true for x' and i.

Proof. By definition, we know there exists y such that (a) $x \preceq y$; (b) $x_i < y_i$; and (c) $x_j = y_j$ for all j with $p'(y)_j \notin \{1, 0, \geq\}$. Now we explicitly show there also exists such a y' for x'. Let y' be the join of x' and y (i.e., $y'_j = \max(x'_j, y_j)$ for all j). Then obviously we have (a) $x' \preceq y'$. Since $x_i = x'_i$, we have (b) $y'_i = y_i > x_i = x'_i$. For the last property (c), note that $y \preceq y'$. By the monotonicity, as long as $y'_j = y_j$ and $p'(y)_j \in \{1, 0, \geq\}$, we have $p'(y')_j \in \{1, 0, \geq\}$. The contrapositive tells us for every j such that $p'(y')_j \notin \{1, 0, \geq\}$, either $y'_j \neq y_j$ (then $y'_j = \max(x'_j, y_j) = x'_j$) or $p'(y)_j \notin \{1, 0, \geq\}$ (then $x_j = y_j$, so $y'_j = \max(x'_j, y_j) = \max(x'_j, x_j) = x'_j$), which is the statement of (c).

This finishes the existence of y' for x' and i. ◁

We divide the proof into two cases:

Case 1: item (1) in Definition 7. Fix a coordinate i and two points $x \preceq x'$ such that $x_i = x'_i$. Suppose that $p^+(x)_i = 1$ (which means $p'(x)_i \in \{1, \geq, \diamond\}$). By monotonicity, we have $p'(x')_i \in \{1, \geq, \diamond\}$ as well. Since the **if** condition on line 5 is true for x, by Claim 20, we know that the **if** condition is also true for x'. Combining with $p'(x')_i \in \{1, \geq, \diamond\}$, we know that $p^+(x')_i \leftarrow 1$ and $p^+(x' + e_i)_i$ is updated by $p^+(x' + e_i)_i \cap \geq$ on line 6, which means $p^+(x')_i = 1$ and $p^+(x' + e_i)_i \in \{1, \geq\}$ at the end of the **for** loop.

Case 2: item (5) in Definition 7. Fix a coordinate i and two points $x \preceq x'$ and $x_i = x'_i$. Suppose that $p^+(x)_i = \geq$. We will prove $p^+(x')_i \in \{1, \geq\}$ at the end of the **for** loop. There are two possibilities: $p^+(x)_i$ is updated on line 6 or line 7. If $p^+(x)_i$ is updated on line 6, then we have $p^+(x' - e_i)_i = 1$ and $p^+(x')_i \in \{1, \geq\}$. If $p^+(x)_i$ is updated on line 7 (which means $p'(x)_i \in \{0, \leq\}$), then we have $p'(x')_i \neq -1$. Meanwhile, by Claim 20, we know that the **if** condition on line 5 is true. So $p^+(x')_i$ will be updated by either 1 or $\geq$.

This finishes the proof that p^+ is a weakly monotone PI function before line 8.

The final step is to show that $p'(x)_i \cap p^+(x)_i$ is well defined for all x and i, which follows from the observation that $p^+(x)_i = 1$ only if $p'(x)_i \in \{1, \geq, \diamond\}$ and $p^+(x)_i = \geq$ only if $p'(x)_i \in \{0, 1, \leq, \geq, \diamond\}$ for all x and i. ◀

Symmetrically, we conclude the following lemma.

▶ **Lemma 21** (Monotonicity Preserving of Subroutine 4). *Given a monotone PI function p, a point $q \in [n]^k$ and a coordinate $\ell \in [k]$ such that $p(q)_\ell \in \{\leq, \diamond\}$ (which implies $q_\ell > 1$), we have the function p^r returned by* `Generate-PI-Function-Minus`(p, q, ℓ) *remains monotone. Furthermore, we have $p^r \Rightarrow p$.*

▶ **Lemma 22** (Safety Preserving of Subroutine 3). *Given a monotone and safe PI function $p : [n]^k \to \{-1, 0, 1, \leq, \geq, \diamond\}^k$, a point q and a coordinate ℓ such that $p(q)_\ell \in \{\geq, \diamond\}$, we have the PI function p^r returned by* `Generate-PI-Function-Plus`(p, q, ℓ) *remains safe.*

Proof. Since $p(q)_\ell \in \{\geq, \diamond\}$, we know that p^r returned by `Generate-PI-Function-Plus`(p, q, ℓ) is also monotone and $p^r \Rightarrow p$ by Lemma 19.

Note that in the subroutine `Generate-PI-Function-Plus`, p^r is obtained by only adding 1 and $\geq$ on the function p. So we have $M_s(p^r) = M_s(p)$ for every slice s. By the same reason, p is safe, and $p^r \Rightarrow p$, we have for any point $x \in \mathcal{L}_s$ with $x \succ M_s(p^r)$, $p^r(x)_i \in \{-1, 0, 1\}$ for all i with $x_i > M_s(p^r)_i$ and $p^r(x)_i = -1$ for some i with $x_i > M_s(p^r)_i$ for all slices s. As a corollary, we have $J_s(p^r) \preceq M_s(p^r)$ for all s, derived from the proof of Lemma 17. (This corollary will be used in this proof later).

So we will focus on proving the first item in Definition 14 for p^r, namely, we will prove for any point $x \in \mathcal{L}_s$ with $x \prec J_s(p^r)$, $p^r(x)_i \in \{-1, 0, 1\}$ for all i such that $x_i < J_s(p^r)_i$ and $p^r(x)_i = 1$ for some i with $x_i < J_s(p^r)_i$.

We first prove the first part: for any point $x \in \mathcal{L}_s$ and $x \prec J_s(p^r)$, $p^r(x)_i \in \{-1, 0, 1\}$ for all i such that $x_i < J_s(p^r)_i$. Fix arbitrarily a slice s, a point $x \in \mathcal{L}_s$ such that $x \prec J_s(p^r)$ and i such that $x_i < J_s(p^r)_i$. We will show that the **if** condition on line 5 is true for x and i. (Note that we need to show there exists a point y such that (a) $x \preceq y$; (b) $x_i < y_i$; and (c) $x_j = y_j$ for all j with $p'(y)_j \notin \{1, 0, \geq\}$. One may try to directly use $J_s(p^r)$ to serve as that y. But note that the definition of $J_s(p^r)$ only guarantees that $x_j = y_j$ for all j with $p^r(y)_j \notin \{1, 0, \geq\}$ instead of what we need in (c) (which concerns $p'(y)$). So extra effort is needed here.)

Let $\mathtt{Y} \coloneqq \{y \mid \text{there exists } i' \text{ such that } J_s(p^r) - e_{i'} \preceq y, (J_s(p^r) - e_{i'})_{i'} < y_{i'} \text{ and } (J_s(p^r) - e_{i'})_j = y_j \text{ for all } j \text{ with } p'(y)_j \notin \{1, 0, \geq\}\}$. Let y^* be the join of $\mathtt{Y} \cup \{J_s(p^r)\}$. Then we prove the following claim.

▷ Claim 23. y^* could serve as the y for the **if** condition on line 5 for x and i.

Proof. Since $x \preceq J_s(p^r)$ and $x_i < J_s(p^r)_i$, we have $x \preceq y^*$ and $x_i < y^*_i$. So in what follows, we will show $p'(y^*)_j \in \{1, 0, \geq\}$ for all j such that $x_j < y^*_j$.

If $\mathtt{Y} = \emptyset$, then we know that $p^+(J_s(p^r))_i = \diamond$ for all i (since any $y \in \mathtt{Y}$ along with the i' should active the condition on line 5, which will update $(J_s(p^r))_{i'}$). So we have $p^r(J_s(p^r))_i = p'(J_s(p^r))_i$ for all i, which implies $x_j = J_s(p^r)_j$ for all j with $p'(J_s(p^r))_j \notin \{1, 0, \geq\}$. This means $y^* = J_s(p^r)$ itself could serve as the y for the **if** condition on line 5 for x and i.

Now let's consider the case $\mathtt{Y} \neq \emptyset$ and let j be such that $x_j < y^*_j$. Let $y \in \mathtt{Y}$ be such that $y_j = y^*_j$ (which must exist since $J_s(p^r) \preceq y$ for all $y \in \mathtt{Y}$). Since $J_s(p^r)_j \leq y^*_j$, we have $(J_s(p^r) - e_j)_j < y_j$. So we have $p'(y)_j \in \{1, 0, \geq\}$, which implies $p'(y^*)_j \in \{1, 0, \geq\}$ as well.

This finishes the proof. ◁

Claim 23 tells us that y^* could serve as the y for the **if** condition on line 5 for x and i. So we know that $p^+(x)_i \in \{1, \geq\}$. Furthermore, $p^+(x)_i = \geq$ only if $p'(x)_i \in \{0, \leq\}$, which implies $p^r(x)_i \in \{-1, 0, 1\}$.

We then prove the second part: for any point $x \in \mathcal{L}_s$ and $x \prec J_s(p^r)$, $p^r(x)_i = 1$ for some i with $x_i < J_s(p^r)_i$. Fix arbitrarily a slice s and a point $x \in \mathcal{L}_s$ such that $x \prec J_s(p^r)$. Assume for the sake of contradiction that $p^r(x)_i \in \{-1, 0\}$ for all i with $x_i < J_s(p^r)_i$. Then construct a new slice s' as follows:

$$s'_i \coloneqq \begin{cases} s_i & s_i \neq *; \\ x_i & s_i = * \text{ and } x_i = J_s(p^r)_i; \\ * & \text{otherwise } (s_i = * \text{ and } x_i < J_s(p^r)_i). \end{cases}$$

Then clearly $x, J_s(p^r) \in \mathcal{L}_{s'}$ and $x_i < (J_s(p^r))_i$ for all i with $s'_i = *$. Note that $p^r(J_s(p^r))_i \in \{\geq, 0\}$ for all i with $s'_i = *$. However, we have $p^r(x)_i \in \{-1, 0\}$ for all i with $s'_i = *$ by assumption. This means $J_{s'}(p^r) \not\preceq M_{s'}(p^r)$, which leads to a contradiction.

This finishes the proof. ◀

Again, symmetrically, we conclude the following lemma.

▶ **Lemma 24** (Safety Preserving of Subroutine 4). *Given a monotone and safe PI function $p : [n]^k \to \{-1, 0, 1, \leq, \geq, \diamond\}$, a point q and a coordinate i such that $p(q)_i \in \{\leq, \diamond\}$, we have the PI function p^r returned by* `Generate-PI-Function-Minus`(p, q, i) *remains safe.*

Before proving the analogs for `Generate-PI-Function-Zero`, we first derive a simple but crucial characterization for any 1-dimensional slice s from the safety.

▷ Claim 25. Given a monotone and safe PI function p, and any 1-dimensional slice s with its free coordinate j, we have

- $p(x)_j = 1$ for all $x \in \mathcal{L}_s$ and $x \prec J_s$; and
- $p(x)_j = -1$ for all $x \in \mathcal{L}_s$ and $x \succ M_s$.

In addition, if $J_s = M_s$ then $p(J_s)_j = p(M_s)_j = 0$; otherwise ($J_s \prec M_s$), we have

- $p(x)_j = \diamond$ for all $J_s \prec x \prec M_s$; and
- $p(J_s)_j = \geq$ and $p(M_s)_j = \leq$.

Proof. Note that in 1-dimensional slice, for any point $x \in \mathcal{L}_s$, $x \not\succeq J_s$ is actually equivalent to $x \prec J_s$. So by the first item of the Definition 14, we have $p(x)_j = 1$ for all $x \in \mathcal{L}_s$ and $x \prec J_s$. Symmetrically, we also have $p(x)_j = -1$ for all $x \in \mathcal{L}_s$ and $x \succ M_s$.

Given that $J_s \preceq M_s$ by Lemma 17, we divide the proof into two simple cases.

Case 1: $J_s = M_s$. Note that by Proposition 13, we have $p(J_s)_j \in \{0, \geq\}$ and $p(M_s)_j \in \{0, \leq\}$. Take the intersection then we have $p(J_s)_j = p(M_s)_j = 0$;

Case 2: $J_s \prec M_s$. Given s is a 1-dimensional slice and $J_s(p) \prec M_s(p)$, for any point $J_s(p) \prec x \prec M_s(p)$, the only way that is consistent with the definition of $J_s(p)$ and $M_s(p)$ is to have $p(x)_j = \diamond$.

Then we move to $p(J_s)_j$. By Proposition 13, we have $p(J_s)_j \in \{0, \geq\}$. Since $J_s \prec M_s$, we know that $p(J_s)_j = \geq$. Symmetrically, we have $p(M_s)_j = \leq$. ◁

Now we are ready to present the analogs for `Generate-PI-Function-Zero`.

▶ **Lemma 26** (Monotonicity and Safety Preserving of Subroutine 5). *Given a monotone and safe PI function $p : [n]^k \to \{-1, 0, 1, \leq, \geq, \diamond\}$, a point q, and a coordinate ℓ such that $p(q)_\ell \in \{\leq, \geq, \diamond\}$, we have the PI function p^r returned by* `Generate-PI-Function-Zero`(p, q, ℓ) *remains monotone and safe. Furthermore, we have $p^r \Rightarrow p$.*

Proof. We first prove two easy cases.

Case 1: $p(q)_\ell = \geq$. We note that in this case, line 2 (the call of `Generate-PI-Function-Plus`) will be skipped, since we have either $q_\ell = 1$ or $p(q - e_\ell)_\ell = 1$ given p is safe. So when we run line 3, *either* it is also skipped then nothing gets changed *or* this lemma can be deduced directly by Lemma 21 and Lemma 24, whose conditions can be verified easily.

Case 2: $p(q)_\ell = \leq$. This case follows from a similar reason. It is easy to show line 3 (the call of `Generate-PI-Function-Minus`) will be skipped and this lemma can be deduced directly by Lemma 19 and Lemma 22.

The following claim essentially proves the last trickier case.

▷ Claim 27. Suppose that we are given a monotone and safe PI function $p : [n]^k \to \{-1, 0, 1, \leq, \geq, \diamond\}$, a point q and a coordinate ℓ such that $1 < q_\ell < n$ and $p(q)_\ell = \diamond$. Let p^r be the PI function returned by `Generate-PI-Function-Plus`$(p, q - e_\ell, \ell)$, then we have $p^r(q)_\ell = \geq$ (so that $p^r(q + e_\ell)_\ell \in \{\leq, \diamond\}$).

Proof. Note that $p'(q)_\ell = \geq$ at the end of line 2. So it suffices for us to prove that $p^+(q)_\ell \neq 1$ at the end of **for** loop.

Assume that $p^+(q)_\ell = 1$ for the sake of contradiction. Then we know there exists y such that (a) $q \preceq y$; (b) $q_\ell < y_\ell$; and (c) $q_j = y_j$ for all j with $p'(y)_j \notin \{1, 0, \geq\}$. Since $q_\ell < y_\ell$, we have $p'(y)_j = p(y)_j$ for all j. So we have (a) $q \preceq y$; (b) $q_\ell < y_\ell$; and (c) $q_j = y_j$ for all j with $p(y)_j \notin \{1, 0, \geq\}$. Define the slice s as follows:

$$s_j \coloneqq \begin{cases} y_j & q_j = y_j; \\ * & \text{otherwise}. \end{cases}$$

Then we have $q, y \in \mathcal{L}_s$ and $y \in \mathtt{Post}_s(p)$. Since $q \preceq y$ and $q_\ell < y_\ell$, by the first property in Definition 14, we know that $p(q)_\ell \neq \diamond$, which contradicts the condition that $p(q)_\ell = \diamond$.

This finishes the proof. ◁

Case 3: $p(q)_\ell = \diamond$.

This implies that $1 < q_\ell < n$. At the end of line 2, by Lemma 19 and Lemma 22, we have p^r remains monotone and safe. Furthermore, $p^r \Rightarrow p$. Now by Claim 27, we know that $p^r(q)_\ell = \geq$, which means $p^r(q + e_\ell) \in \{\leq, \diamond\}$ by Claim 25.

So at the end of line 3, by Lemma 21 and Lemma 24 (which need the condition of $p^r(q + e_\ell) \in \{\leq, \diamond\}$), we have p^r remains monotone and safe. Furthermore, $p^r \Rightarrow p$.

This finishes the proof. ◀

▶ **Lemma 28.** *Given a monotone and safe PI function p, a point $q \in [n]^k$, a coordinate $\ell \in [k]$, and $b \in \{-1, 0, 1\}$ such that $(p(q)_\ell, b)$ satisfies the following condition:*

$$p(q)_\ell \in \{\geq, \diamond\} \text{ if } b = 1; \;\; p(q)_\ell \in \{\leq, \diamond\} \text{ if } b = -1; \;\; p(q)_\ell \in \{\leq, \geq, \diamond\} \text{ if } b = 0,$$

the function p^r returned by `Generate-PI-Function`(p, q, ℓ, b) *satisfies $p^r(q)_\ell = b$.*

Proof. When $b = 1$, we have $p'(q)_\ell = 1$ at the end of line 2. Since $p^r \Rightarrow p'$, we have $p^r(q)_\ell = 1$ as well. The case of $b = -1$ is similar.

If $q_\ell = 1$, $q_\ell = n$, or $p(q)_\ell \neq \diamond$, then $p^r(q)_\ell = 0$ can be derived by previous cases since at most one of `Generate-PI-Function-Plus` and `Generate-PI-Function-Minus` is called. For the case $1 < q_\ell < n$ and $p(q)_\ell = \diamond$, by Claim 27, we have both `Generate-PI-Function-Plus` and `Generate-PI-Function-Minus` are called and $p^r(q - e_\ell)_\ell = 1$ and $p^r(q - e_\ell)_\ell = -1$, which forces $p^r(q)_\ell = 0$ since p is monotone. ◀

3.4 Not Creating New Fixed Points

▶ **Lemma 29** (Fixed Points of Subroutine 3)**.** *Given a monotone function $f : [n]^k \to [n]^k$, a PI function p, a point q and a coordinate ℓ such that*

- *p is monotone and safe;*
- *$p(q)_\ell \in \{\geq, \diamond\}$;*
- *$f|_p(q + e_\ell)_\ell \geq q_\ell + 1$; and*
- *$\mathtt{Fix}(f|_p) \subseteq \mathtt{Fix}(f)$,*

the function p^r returned by `Generate-PI-Function-Plus`(p, q, ℓ) *satisfies* $\mathtt{Fix}(f|_{p^r}) \subseteq \mathtt{Fix}(f|_p) \subseteq \mathtt{Fix}(f)$.

Proof. Note that whenever we have $p^r(x)_i \neq p(x)_i$ for some x and i, it must be the case that $p^r(x)_i = 1$ or $p^r(x)_i = p(x)_i \cap \geq$. If $p^r(x)_i = 1$, then we have $f|_{p^r}(x) \neq x$, which means x is not a fixed point of $f|_{p^r}$. So we only need to analyze the case that $p(x)_i \neq p^r(x)_i = p(x)_i \cap \geq$.

Fix arbitrary z and i such that $p(z)_i \neq p^r(z)_i = p(z)_i \cap \geq$. First consider the updating rule on line 2, in which case $i = \ell$ and $z = x + e_\ell$ for some $x \succeq q$ and $x_\ell = q_\ell$, then we have $f|_p(z)_\ell \geq z_\ell$ by the third condition. Note that it suffices for us to know $f|_p(x)_i \geq x_i$, since it implies that either $f|_p(x)_i = f|_{p^r}(x)_i$ or $f|_{p^r}(x)_i = x_i + 1$ since $p^r(x)_i \in \{0, 1, \geq\}$ given that $p^r(x)_i = p(x)_i \cap \geq$.

Next, we consider the case that $p^+(z)_i$ is updated on line 6 and 7, where we will show $f|_{p^r}(x) \neq x$. Let y be such that (a) $z \preceq y$; (b) $z_i - 1 < y_i$ ($z_i \leq y_i$); and (c) $z_j = y_j$ for all j with $p(y)_j \notin \{1, 0, \geq\}$ on line 5. Define a slice s as follows:

$$s_j := \begin{cases} y_j & p(y)_j \notin \{1, 0, \geq\}; \\ * & \text{otherwise.} \end{cases}$$

Then we have $z, y \in \mathcal{L}_s$ and $z \preceq J_s(p)$ (given that $z \preceq y$ and $y \preceq J_s(p)$). Further note that $z \neq J_s(p)$, otherwise we have $p(z)_i = p^r(z)_i = p(z)_i \cap \geq$. So it suffices for us to argue $f|_p(z) \neq z$ for all $z \prec J_s(p)$, which follows from that p is safe and Lemma 17.

This finishes the proof. ◀

We conclude the analog for `Generate-PI-Function-Minus`.

▶ **Lemma 30** (Fixed Points Preserving of Subroutine 4). *Given a monotone function $f : [n]^k \to [n]^k$, a PI function p, a point q and a coordinate ℓ such that*

- *p is monotone and safe;*
- *$p(q)_\ell \in \{\leq, \diamond\}$;*
- *$f|_p(q - e_\ell)_\ell \leq q_\ell - 1$; and*
- *$\texttt{Fix}(f|_p) \subseteq \texttt{Fix}(f)$,*

the function p^r returned by `Generate-PI-Function-Minus`(p, q, ℓ) *satisfies* $\texttt{Fix}(f|_{p^r}) \subseteq \texttt{Fix}(f|_p) \subseteq \texttt{Fix}(f)$.

▶ **Lemma 31** (Fixed Points of Subroutine 5). *Given a monotone function $f : [n]^k \to [n]^k$, a PI function p, a point q and a coordinate ℓ such that*

- *p is monotone and safe;*
- *$p(q)_\ell \in \{\leq, \geq, \diamond\}$;*
- *$f|_p(q)_\ell = q_\ell$; and*
- *$\textit{Fix}(f|_p) \subseteq \textit{Fix}(f)$,*

the function p^r returned by `Generate-PI-Function-Zero`(p, q, ℓ) *satisfies* $\textit{Fix}(f|_{p^r}) \subseteq \textit{Fix}(f|_p) \subseteq \textit{Fix}(f)$.

Proof. Let us consider the non-trivial case where both subroutines `Generate-PI-Function-Plus` and `Generate-PI-Function-Minus` are called. Otherwise, this lemma can be derived by either Lemma 29 or Lemma 30 (given that $f|_p(q)_\ell = q_\ell$).

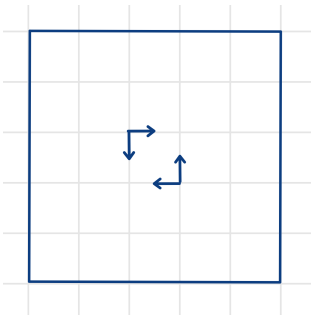
Suppose that both subroutines are called, then we have $1 < q_\ell < n$ and $p(q)_\ell = \diamond$. Since $f|_p(q)_\ell = q_\ell$, we have $f(q)_\ell = q_\ell$.

By Claim 27, we know that at the end of line 2, we have $p^r(q)_\ell = \geq$ and $p^r(q + e_\ell)_\ell \in \{\leq, \diamond\}$. At this time, we still have $f|_{p^r}(q)_\ell = q_\ell$ as well as other properties in the condition of this lemma by Lemmas 19, 22, and 29. So this lemma can be derived by Lemmas 21, 24, and 30.

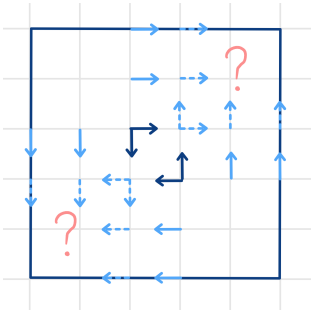
This finishes the proof. ◀

4 An Illustrating Example

In this section, we illustrate how our reduction works in one concrete but tricky example. Recall that we have to make sure our Algorithm 1 works for *any* monotone function and *any* algorithm solving UniqueTarski. For simplicity, we pick the following 2D example: a monotone function $f : [6]^2 \to [6]^2$ with $f(3,4) = (4,3)$ and $f(4,3) = (3,4)$ as shown in Figure 2a, as well as an algorithm $\mathcal{U}$ for UniqueTarski, which will first query $(3,4)$, given the answer $f(3,4) = (4,3)$, then query $(4,3)$.

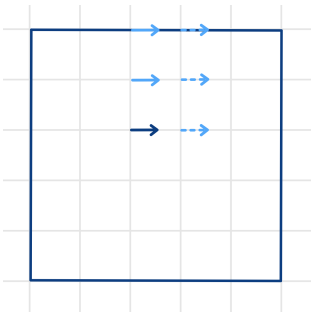


(a) A monotone function $f : [6]^2 \to [6]^2$ with $f(3,4) = (4,3)$ and $f(4,3) = (3,4)$.

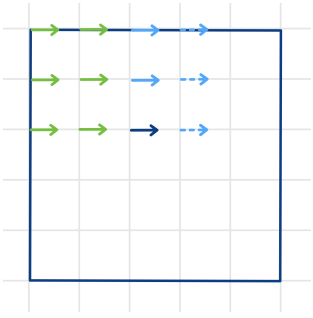


(b) The standard partial information derived by $(3,4)$ and $(4,3)$, described in the light blue color. The solid arrows mean -1 or 1 and the dashed arrows mean $\leq$ or $\geq$ (the same rule applies below).

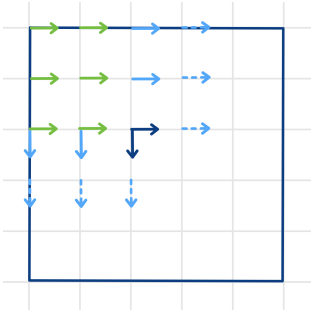
Figure 2 A 2D example for which after two queries the algorithm $\mathcal{U}$ for UniqueTarski will fail.



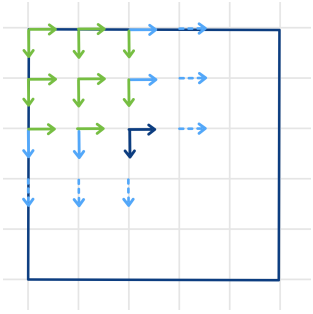
(a) The partial information by adding $f(3,4)_1 = 4$.



(b) The safe PI function constructed by Algorithm 1. The new information is described in the green color (the same rule applies below).

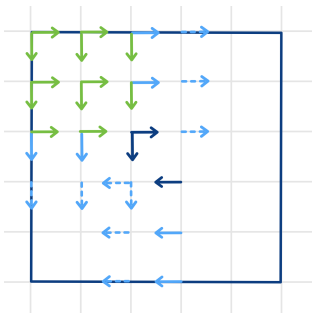


(c) The partial information by adding $f(3,4)_2 = 3$.

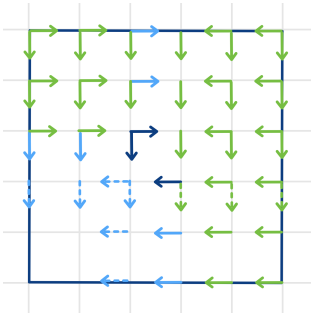


(d) The safe PI function constructed by Algorithm 1.

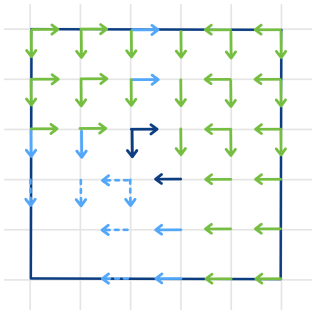
Figure 3 The evolution of PI function when adding $f(3,4)_1 = 4$ and $f(3,4)_2 = 3$.



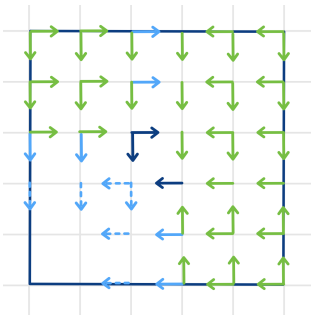
(a) The partial information by adding $f(4,3)_1 = 3$.



(b) The safe PI function constructed by Algorithm 1.



(c) The partial information by adding $f(4,3)_2 = 4$.



(d) The safe PI function constructed by Algorithm 1.

Figure 4 The evolution of PI function when adding $f(4,3)_1 = 3$ and $f(4,3)_2 = 4$.

Note that the function (actually partial function) in Figure 2a does not violate monotonicity. But clearly, no monotone function that is consistent with Figure 2a has a unique fixed point. This is because the partial information derived from $f(3,4) = (4,3)$ and $f(4,3) = (3,4)$ is sufficient to conclude the existence of fixed points in both the bottom left corner and top right corner, as shown in Figure 2b. Observe that if the algorithm $\mathcal{U}$ is not fooled, it could immediately reject the function f and return "the underlying function has multiple fixed points" once it gets the true answer $f(4,3) = (3,4)$.

Perhaps surprisingly, our reduction will modify the answer the algorithm $\mathcal{U}$ gets when querying $(3,4)$, by creating safe PI functions p that satisfy $\texttt{Fix}(f|_p) \subseteq \texttt{Fix}(f)$ (the formal statement is in Theorem 15).

We show how the PI function evolves step by step in Figure 3 and 4. The figures on the left-hand side are obtained by adding one piece of information (namely, $f(3,4)_1 = 4$, $f(3,4)_2 = 3$, $f(4,3)_1 = 3$, and $f(4,3)_2 = 4$). The figures on the right-hand side are obtained by the Subroutine `Generate-PI-Function`. Note that in the last step after Figure 4b, we will try to add the last piece of information $f(4,3)_2 = 4$. However, since $p(4,3)_2 =\leq$ already, the algorithm $\mathcal{U}$ will get $f|_p(4,3)_2 = 3$.

It is easy to verify that all PI functions of the figures on the right-hand side are safe and satisfy $\texttt{Fix}(f|_p) \subseteq \texttt{Fix}(f)$. In particular, for Figure 4d, every point outside the bottom left corner is certainly not a fixed point of $f|_p$, and the fixed point(s) in the bottom left corner is not affected.

5 Promise Problem versus TFNP Version

The problems $\textsc{Tarski}(n,k)$ and $\textsc{UniqueTarski}(n,k)$ are *promise problems*. In the former, we want to compute a fixed point of the given function under the promise (condition) that it is monotone; in the latter the function is promised to be monotone and have a unique fixed point.

From a promise problem, one can define a total search problem, where on any given arbitrary input one seeks either a desired solution as in the promise problem, or a violation certificate showing that the input does not satisfy the promise. The total search version of the Tarski problem is formally the following search problem.

▶ **Definition 32** (Total search version of $\textsc{Tarski}(n,k)$)**.** *Given a function $f : [n]^k \to [n]^k$, find one of the following:*

- *a point $x \in [n]^k$ such that $f(x) = x$; or*
- *two points $x, y \in [n]^k$ such that $x \preceq y$ and $f(x) \not\preceq f(y)$.*

In the black box setting, the function f is given by a black box (an oracle). In the white box setting, the function f is given by a poly$(k, \log n)$-size circuit C with $k * \lceil \log n \rceil$ input gates and $k * \lceil \log n \rceil$ output gates.

The total search version of Tarski in the white box setting is in TFNP, in fact it is PLS ∩ PPAD. Any algorithm for the total search version of a promise problem (whether in the white box or the black box setting) can be obviously used also to solve the promise problem, so the total version is always at least as hard as the promise problem. In general the converse may not hold, since in the total search version, the algorithm is not allowed to simply fail if the input does not satisfy the promise, but it must provide a violation certificate (and in general the complexity of the total problem may depend on the type of certificate that is required). In the case of the Tarski problem in the black box setting however it is easy to see that the total version is no harder than the promise problem. This is because of the following property.

▶ **Lemma 33.** *Let $Q = \{q^1, \ldots, q^m\}$ be a set of query points in $[n]^k$ and $A = \{a^1, \ldots, a^m\}$ the corresponding answers of the black box. There is a monotone function f that is consistent with all the answers (i.e such that $f(q^i) = a^i$ for all $i \in [m]$) if and only if there is no pair i, j such that $q^i \preceq q^j$ and $a^i \not\preceq a^j$.*

Proof. If there is a pair i, j such that $q^i \preceq q^j$ and $a^i \not\preceq a^j$ then clearly there is no monotone function f that is consistent with the answers. Suppose now that there is no such pair. Define the function f as follows: For every point $x \in [n]^k$ and every coordinate i, set $f(x)_i = \min\{a^j_i \mid x \preceq q^j\}$; if the set on the right-hand side is empty then set $f(x)_i = n$. We have to show that f is monotone and is consistent with the answers.

Consider any two points $x \preceq y$ and any coordinate i. Then $y \preceq q^j$ implies $x \preceq q^j$, thus $f(x)_i = \min\{a^j_i \mid x \preceq q^j\} \leq f(y)_i = \min\{a^j_i \mid y \preceq q^j\}$. Therefore, f is monotone.

By the definition of f, for any query point q^t and coordinate i, $f(q^t)_i = \min\{a^j_i \mid q^t \preceq q^j\} \leq a^t_i$. If $f(q^t)_i < a^t_i$ then there is another query point q^j such that $q^t \preceq q^j$ and $a^j_i < a^t_i$, hence $a^t \not\preceq a^j$. ◀

▶ **Corollary 34.** *In the black-box setting, suppose that $\textsc{Tarski}(n,k)$ (the promise problem) can be solved in $q(n,k)$ queries and $t(n,k)$ time, then total search version of $\textsc{Tarski}(n,k)$ can be solved in $q(n,k)$ queries and $O(t(n,k) + q(n,k)^2 \cdot k)$ time.*

Proof. Run the algorithm for the promise problem. Either the algorithm will find a fixed point within the query and time complexity of the promise problem, or two of the query points provide a violation certificate. ◀

We showed that TARSKI(n,k) reduces to UNIQUETARSKI(n,k) with the same query complexity. Therefore, we have.

▶ **Corollary 35.** *Any black-box algorithm for UNIQUETARSKI(n,k) (the promise problem) can be used to solve also the total search version of TARSKI(n,k) with the same query complexity.*

We can define a total search version of UNIQUETARSKI(n,k) that includes as a possible answer also a violation certificate of uniqueness. One way to define it is as follows.

▶ **Definition 36** (Total search version of UNIQUETARSKI(n,k))**.** *Given a function $f : [n]^k \to [n]^k$, find one of the following:*

- *a point $x \in [n]^k$ such that $f(x) = x$; or*
- *two points $x, y \in [n]^k$ such that $x \preceq y$ and $f(x) \not\preceq f(y)$; or*
- *two points $x, y \in [n]^k$ such that $x \preceq f(x)$, $y \succeq f(y)$ and $x \not\preceq y$.*

In the black box setting, the function f is given by a black box (an oracle). In the white box setting, the function f is given by a poly$(k, \log n)$-size circuit C with $k * \lceil \log n \rceil$ input gates and $k * \lceil \log n \rceil$ output gates.

Note that if f is monotone and $x \preceq f(x)$ then f has a fixed point in $\mathcal{L}_{x,n^k}$, and if $y \succeq f(y)$ then f has a fixed point in $\mathcal{L}_{1^k,y}$. If $x \not\preceq y$ then $\mathcal{L}_{1^k,y}$ and $\mathcal{L}_{x,n^k}$ are disjoint, and hence f has at least two fixed points. Clearly, the total search version of TARSKI(n,k) is at least as hard as that of UNIQUETARSKI(n,k), both in the white box and the black box setting, since the latter includes one more option for an acceptable output. It is not much harder however. Let T-TARSKI(n,k) and T-UNIQUETARSKI(n,k) denote the total search versions of the two problems, as defined above.

▶ **Theorem 37.** *If T-UNIQUETARSKI(n,k) can be solved in $q(n,k)$ queries in the black box setting, then T-TARSKI(n,k) can be solved in $q(n,k)$ queries. If T-UNIQUETARSKI(n,k) can be solved in time $t(n,k)$ in the black box (respectively, white box) setting, then T-TARSKI(n,k) can be solved in time $O(t(n,k) * (k \cdot \log n))$ in the black box (resp. white box) setting.*

Proof. The statement in the first sentence follows from Corollary 35. Next, we show the statement in the second sentence.

Given a black-box or white-box algorithm $\mathcal{U}$ for T-UNIQUETARSKI(n,k), the algorithm for T-TARSKI(n,k) in the same setting is as follows. Use the algorithm $\mathcal{U}$ to find a solution of T-UNIQUETARSKI(n,k). If the solution is a fixed point (i.e., a point $x \in [n]^k$ such that $f(x) = x$) or a violation certificate of monotonicity (i.e., two points $x, y \in [n]^k$ such that $x \preceq y$ and $f(x) \not\preceq f(y)$) then we are done, since they are also solution of T-TARSKI(n,k). Otherwise, we find a solution that is a violation certificate of uniqueness (i.e., two points $x, y \in [n]^k$ such that $x \preceq f(x)$, $y \succeq f(y)$ and $x \not\succeq y$). Then there exists i such that $x_i > y_i$, which means either $x_i > n/2$ or $y_i \leq n/2$. If $x_i > n/2$, then we shrink the search space to $\mathcal{L}_{x,n^k}$ and recursively call $\mathcal{U}$ to find a solution in $\mathcal{L}_{x,n^k}$; If $y_i \leq n/2$, then we shrink the search space to $\mathcal{L}_{1^k,y}$ and recursively call $\mathcal{U}$ to find a solution in $\mathcal{L}_{1^k,y}$. The function f may map a point q in the reduced space to a point outside the space; in that case the point q together with either the top or the bottom point of the reduced space form a violation certificate for monotonicity. In the black box setting, if the algorithm ever queries such a point q then we immediately get a violation of monotonicity and can terminate. In the white box setting,

when we recurse to the reduced space, we replace the circuit for f with a modified circuit for a function f' which restricts the coordinates of the output point to lie in the reduced space. When the recursive call returns a solution to T-TARSKI for the reduced space, i.e. either a fixed point x of f' or a pair of points x, y that certify that f' is not monotone, then we test if f and f' have the same value on these points. If they do, then they constitute a solution for f in the original space; if one of them does not, then that point with the bottom or the top element provide a certificate for the violation of monotonicity of f.

The search space goes down by a factor of two after each call of $\mathcal{U}$. So after at most $k \cdot \log n$ many rounds, we can find a solution of T-TARSKI(n, k). ◀

6 Discussion and Open Problems

Our results resolve an open question in [4] and could potentially shed new light on the upper bounds and lower bounds of the query complexity of TARSKI(n, k). As we showed, TARSKI(n, k) is no harder, with respect to query complexity, than the special case of monotone functions that have a unique fixed point in the lattice, and even further, have a unique fixed point in every slice of the lattice. There is a lot of structure in such monotone functions. In a function f with a unique fixed point, the least fixed point and the greatest fixed point coincide. There is a path connecting the bottom element 1^k of the lattice with the top element n^k, the fixed point lies on this path, and the function f on all points in this path point in the direction of the fixed point. The same structure holds for every slice if the function has a unique fixed point on all slices. This structure may well be useful in helping to design an algorithm with low complexity. On the lower bound side, it may also provide a useful framework; indeed the lower bound constructions for two dimensions in [4] use this structure. Can we use uniqueness to improve the bounds on the query complexity of Tarski?

A second question concerns the time complexity of the algorithms in the black box setting. Our reduction involves the maintenance of a partial information function p that is defined over the whole lattice. A straightforward implementation would take of course exponential time. Note however that we do not need to compute p on the whole domain; we only need to be able to compute p on demand on specific points, namely the query points of the Unique Tarski algorithm. Is it possible to implement the algorithm so that it runs in polynomial time in the number of queries? More generally, does the black-box time complexity of TARSKI(n, k) reduce also to that of UNIQUETARSKI(n, k)?

Regarding the white-box complexity, we know that the total search version of TARSKI(n, k) is in PLS ∩ PPAD [4] and thus by the results of [5, 8], it is in the classes CLS (Continuous-Local-Search) and EOPL (End-of-Potential-Line). Is the total search version of UNIQUETARSKI in the class UEOPL (Unique-EOPL) [6]? Is it hard for UEOPL?

References

1 Xi Chen and Yuhao Li. Improved upper bounds for finding tarski fixed points. In *Proceedings of the 23rd ACM Conference on Economics and Computation*, pages 1108–1118, 2022.

2 Anne Condon. The complexity of stochastic games. *Information and Computation*, 96(2):203–224, 1992.

3 Chuangyin Dang, Qi Qi, and Yinyu Ye. Computational models and complexities of tarski's fixed points. Technical report, Stanford University, 2011.

4 Kousha Etessami, Christos Papadimitriou, Aviad Rubinstein, and Mihalis Yannakakis. Tarski's theorem, supermodular games, and the complexity of equilibria. In *11th Innovations in Theoretical Computer Science Conference (ITCS 2020)*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2020.

5 John Fearnley, Paul W. Goldberg, Alexandros Hollender, and Rahul Savani. The complexity of gradient descent: CLS = PPAD ∩ PLS. In *STOC '21: 53rd Annual ACM SIGACT Symposium on Theory of Computing*, pages 46–59. ACM, 2021.

6 John Fearnley, Spencer Gordon, Ruta Mehta, and Rahul Savani. Unique end of potential line. *J. Comput. Syst. Sci.*, 114:1–35, 2020.

7 John Fearnley, Dömötör Pálvölgyi, and Rahul Savani. A faster algorithm for finding tarski fixed points. *ACM Transactions on Algorithms (TALG)*, 18(3):1–23, 2022.

8 Mika Göös, Alexandros Hollender, Siddhartha Jain, Gilbert Maystre, William Pires, Robert Robere, and Ran Tao. Further collapses in TFNP. In *37th Computational Complexity Conference, CCC 2022, July 20-23, 2022, Philadelphia, PA, USA*, volume 234 of *LIPIcs*, pages 33:1–33:15. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022.

9 Massimo Marinacci and Luigi Montrucchio. Unique tarski fixed points. *Math. Oper. Res.*, 44(4):1174–1191, 2019. doi:10.1287/moor.2018.0959.

10 Paul Milgrom and John Roberts. Rationalizability, learning, and equilibrium in games with strategic complementarities. *Econometrica: Journal of the Econometric Society*, pages 1255–1277, 1990.

11 L. Shapley. Stochastic games. *Proc. Nat. Acad. Sci.*, 39(10):1095–1100, 1953.

12 Alfred Tarski. A lattice-theoretical fixpoint theorem and its applications. *Pacific journal of Mathematics*, 5(2):285–309, 1955.

13 Donald M Topkis. Equilibrium points in nonzero-sum n-person submodular games. *Siam Journal on control and optimization*, 17(6):773–787, 1979.

14 Donald M Topkis. *Supermodularity and Complementarity*. Princeton University Press, 1998.