Electronic Colloquium on Computational Complexity, Report No. 172 (2026)

Design Methodologies for Interactive Proof Systems

Inbar Ben Yaacov* Oded Goldreich† Guy N. Rothblum‡

September 8, 2026

Abstract

We present a methodology for constructing interactive proof systems. This methodology, which is implicit in prior works, consists of reducing the original claim to an iteratively generated sequence of claims such that each claim is (interactively) generated based on the prior claim. Viewing each of these interactive generation steps as solving an adequate search problem, we present composition results that support a modular construction of interactive proof systems as well as their transformations to PCIPs (aka IOPs).

The development of the foregoing methodology involves the explicit introduction of a few notions, which are of independent interest. These include "dichotomous search problems" (to be solved by protocols analogous to interactive proofs) and oracle-aided protocols (in which the oracle is a search problem rather than a decision problem).

Using the foregoing methodology, we prove that every set in uniform-$\mathcal{NC}$ has a doubly-efficient PCIP. This result was conjectured by Arnon, Chiesa, and Yogev (*37th CCC*, 2022), and our construction follows their ideas. Our contribution is in adapting their "IP to PCIP" transformation to the context of oracle-aided protocols and applying the foregoing methodology while following the ideas of Goldwasser, Kalai, and Rothblum (*40th STOC*, 2008).

*Department of Computer Science, Weizmann Institute of Science, Rehovot, Israel.
†Department of Computer Science, Weizmann Institute of Science, Rehovot, Israel.
‡Apple.

ISSN 1433-8092

Contents

1 Introduction

The notion of interactive proof systems, introduced in full generality in [GMR85], requires no introduction. Of special interest are doubly-efficient interactive proofs, introduced in [GKR15], in which the honest prover is required to run in polynomial time, and the verifier is required to run in almost linear time.[1] Another strengthening of interactive proof systems, called PCIPs in [RRR16] and IOPs in [BSCS16], identifies systems in which the verifier actually reads only a constant number of bits in the messages sent to it.

Numerous constructions of interactive proof systems have appeared in the literature, achieving different goals (i.e., combinations of various efficiency measures and additional features (for various complexity classes)). Yet, the construction of interactive proof systems remains an active research area, since many other goals are still out of reach. This fact should not come as a surprise when one thinks of the construction of interactive proof systems as analogous to the design of algorithms. In light of such an analogy, methodologies for constructing interactive proof systems are of natural interest.

Such a methodology is explicitly developed in the current paper. It is also used to obtain a new result, conjectured in [ACY22b]: *Every set in uniform-$\mathcal{NC}$ has a doubly-efficient PCIP*. More generally, the result applies to any set that is decidable by a uniform family of polynomial-time circuits of bounded depth, and the number of rounds in the resulting system is polynomial in the depth of the circuit.

1.1 Overview

We present a methodology for constructing interactive proof systems. This methodology, which is implicit in prior works (e.g., [LFKN92, GKR15, ACY22b]), consists of reducing the original claim to an iteratively generated sequence of claims such that each claim is (interactively) generated based on the prior claim. Actually, we view each of these interactive generation steps as solving an adequate *search* problem.

Hence, the pivot of our methodology is the notion of solving a search problem by an IP-type protocol. In the examples we present, the inputs to these search problems are claims of a certain type, and the outputs are claims of a related type, such that the output claim is valid if and only if the input claim is valid. In general, following [BB18] and generalizing it, we consider two-party protocols in which a computationally-bounded "searcher" (which generalizes the verifier) is assisted by a more powerful but untrusted "guide" (which generalizes the prover) in solving a search problem. In fact, wishing to use such search problems as subroutines leads us to generalize the standard notion of a search problem (which is captured by a single binary relation) and use a notion of a "dichotomous" search problem (which is captured by a pair of binary relations).[2]

Indeed, the celebrated sum-check protocol of [LFKN92] and the GKR-protocol (for sets in uniform-$\mathcal{NC}$ [GKR15]) can be cast as iteratively generating a sequence of dichotomous search problems. For example, the di-

[1] Recall that no computational constraints are applied to the cheating prover (in the soundness condition). (Here, we use the most restrictive definition of doubly-efficient interactive proof systems: Under the most liberal formulation, when considering a proof system for a set that is decidable in time T, the honest prover is required to run in $\mathrm{poly}(\mathsf{T})$-time and the verifier is required to run in $\min(o(\mathsf{T}), \mathrm{poly})$-time.)

[2] In the foregoing example, the first relation consists of pairs of valid claims, and the second relation consists of invalid claim pairs.

chotomous search problem used in the sum-check protocol is given an evaluation claim regarding a multivariate polynomial and outputs an evaluation claim that refers to a related polynomial (obtained by instantiating one variable). Such a presentation of these celebrated protocols facilitates presenting "selective message reading" versions of the original protocols (and deriving corresponding PCIPs). Recall that [ACY22b] established a PCIP version of the sum-check protocol, but not for the GKR-protocol (although it was conjectured in [ACY22b]).

Selective Message Reading (SMR) protocols. For a fixed bound ρ (possibly and typically a constant), a selective message reading protocol is one in which the computationally-bounded party (i.e., verifier or searcher) reads at most ρ of the messages sent by the more powerful party. (Such protocols, considered in [ACY22b], are typically of the public-coin type.)[3]

Jumping ahead, we mention that Arnon *et al.* [ACY22b] presented two transformations of certain classes of interactive proof systems to SMR protocols (in which two messages are read). As detailed below, we generalize both transformations to the context of dichotomous search systems. Our adaptations rely on the existence of suitable "solution merging" protocols for the corresponding search problems. In particular, when given a sequence of potential solutions to the same instance, the merging protocol produces a potential solution such that the resulting solution is valid if and only if all given solutions are valid. (Recall that so far, we viewed instances and solutions as claims that may be valid or invalid.) We stress that the actual formulation is actually more general, but first let us discuss the general notion of an dichotomous search problem.

The general notion of dichotomous search problems. We first consider dichotomous search problems in which the instances and the potential solutions are claims. We envisioned such search problems as being "solved" by an interaction of a computationally-bounded searcher with a more powerful but untrusted guide. So far, while distinguishing valid claims from invalid ones, we blurred the fact that different requirements are made depending on whether the input claim is valid or not. We clarify this fact now. On the one hand, if the input claim is valid, then the output of the searcher is a related valid claim, provided that the guide follows the prescribed (honest) strategy. On the other hand, if the input claim is invalid, then, no matter how maliciously the guide behaves (under a cheating strategy), the searcher either detects cheating (and rejects) or outputs a related invalid claim. These different conditions, akin to completeness and soundness in interactive proof systems, are the reason that we use the term "dichotomous" for the foregoing search problems.

While the foregoing formulation suffices for our applications, we seize the opportunity to present a more general formulation of dichotomous search problems, hereafter referred to as D-search problems. In the general formulation, we consider two binary relations, $Q, R \subseteq \{0,1\}^* \times \{0,1\}^*$, that represent possible instance-solution pairs, but we neither require that $\mathrm{Dom}(Q) \cap \mathrm{Dom}(R) = \emptyset$ nor that $\mathrm{Dom}(Q) \cup \mathrm{Dom}(R) = \{0,1\}^*$, where for a relation $P \subseteq \{0,1\}^* \times \{0,1\}^*$ we define $\mathrm{Dom}(P) = \{x : \exists y \text{ s.t. } (x,y) \in P\}$. (Neither do we require that the ranges of Q and R are disjoint, where the range of P is $\{y : \exists x \text{ s.t. } (x,y) \in P\}$.) The corresponding completeness and soundness conditions are stated as follows.

[3] Unlike SMR systems, the analog query-round IOP of [ACY22b] also restricts the verifier from reading his own sent messages. The SMR model can be transformed into the model of [ACY22b] via the transformation of [ACY22a].

Completeness: if the input x is in $\mathrm{Dom}(Q)$ (and the guide follows the honest strategy), then the searcher's output is in $Q(x) := \{y : (x,y) \in Q\}$.

Soundness: if the input x is in $\mathrm{Dom}(R)$, then (no matter how the guide behaves) the searcher either rejects or outputs y such that $(x,y) \in R$.

(Of course, in actual protocols, we may allow a probability of error.) Note that the completeness requirement regarding Q represents the standard notion of solving the search problem captured by Q; that is, given an input $x \in \mathrm{Dom}(Q)$, the solver (composed of the prescribed pair of interactive machines) should produce a solution y in $Q(x)$. In contrast, the soundness requirement regarding R represents a "safety" condition that refers to dishonest behavior of the guide. Indeed, one can think of Q as the "qualified" relation and R as the "robust" relation.

We note that an interactive proof system for a set L may be cast as solving the dichotomous search problems (Q,R) such that $Q = \{(x,1) : x \in L\}$ and $R = \{(x,0) : x \notin L\}$. We also note that the special case of $Q = R$ is admissible (and is actually the formulation considered in [BB18]), but this special case does not suffice for our applications (e.g., a single round of the sum-check protocol can be cast as solving a dichotomous search problem but not when using $Q = R$).[4]

Composition. We re-derive a doubly-efficient interactive proof system for uniform-$\mathcal{NC}$ as well as obtain an SMR version of it (with a constant number of reads) by presenting and applying a general composition theorem for dichotomous search problems. This theorem refers to an arbitrary oracle-aided protocol that makes oracle calls to an auxiliary D-search problem. We stress that parallel oracle calls are allowed (i.e., the protocol may make several oracle calls in the same round).

Specifically, mimicking the GKR-protocol (of [GKR15]), we consider an oracle-aided protocol that uses a D-search problem that refers to adjacent layers of the circuit that decides the $\mathcal{NC}$-set. The input to the latter D-search problem is a claim regarding the (low-degree extension of the) function that represents the contents of a layer (in a computation on a fixed input). The desired output is a claim regarding the (low-degree extension of the) function that represents the contents of the prior layer in this circuit (i.e., the adjacent layer that is closer to the circuit's input). This D-search problem is essentially solved by the sum-check protocol (when we ignore some additional steps).[5]

In general, composing an "outer" oracle-aided protocol with an "inner" protocol that solves the corresponding D-search problem, we obtain a standard protocol that solves the D-search problem that is solved by the outer protocol.[6] The complexities of the resulting protocol are functions of the corresponding complexities of the two combined protocols. In particular, in the SMR model, the number of messages read by the resulting protocol equals the product of the corresponding numbers in the two protocols. The completeness (resp., soundness) error of the resulting protocol equal the product of the number of oracle calls performed by the outer protocol and the completeness (resp., soundness)

[4] As detailed in Definition 2.1, the straightforward casting uses Q as the set of all pairs of valid (evaluation) claims and R as the set of all pairs of invalid (evaluation) claims. Recall that the soundness condition is stated only for R. In contrast, the soundness condition cannot hold (in this case) for $Q \cup R$.

[5] These steps include an evaluation of a fixed polynomial that represents the circuit's topology and reducing two evaluation claims to one such claim (via a merging protocol).

[6] Recall that an interactive proof system for a set L may be cast as solving the dichotomous search problems (Q,R) such that $Q = \{(x,1) : x \in L\}$ and $R = \{(x,0) : x \notin L\}$.

error of the inner protocol, where this presupposes that the outer protocol has zero completeness (resp., soundness) error. (Otherwise, the completeness (resp., soundness) error of the outer protocol is added to the foregoing product.)

We stress that the foregoing two paragraphs immediately yield a doubly-efficient interactive proof system for uniform-$\mathcal{NC}$, which reestablishes the result of [GKR15] in a modular manner. This is the case since the outer oracle-aided protocol (outlined in the first paragraph) is quite trivial, whereas the inner protocol is essentially the well-known sum-check protocol. Furthermore, we can derive an SMR version of the GKR-protocol by using SMR versions of both the foregoing outer and inner protocols. We obtain the latter versions by adapting the two transformations presented in [ACY22b]. We warn, however, that here we need to obtain an SMR version of the sum-check protocol that has a smaller soundness error than the (constant) error obtained in [ACY22b].

From oracle-aided protocols to SMR ones. The first transformation (to SMR protocols) is easier to describe, but yields a far less efficient protocol (which does not suffice for our result). For every $\ell \ll \ell_{\mathsf{SMR}}$, given an ℓ-round oracle-aided protocol that makes a single oracle call to a D-search problem $(Q_{\mathsf{in}}, R_{\mathsf{in}})$ in each round, we derive an $(\ell_{\mathsf{SMR}} + O(1))$-round SMR version as follows. We consider $m := \binom{\ell_{\mathsf{SMR}}}{\ell}$ parallel executions of the original protocol such that each execution is associated with a different ℓ-subset of $[\ell_{\mathsf{SMR}}]$. The execution associated with $I \in \binom{[\ell_{\mathsf{SMR}}]}{\ell}$ proceeds in round i if $i \in I$ and remains idle otherwise.[7] Following these ℓ_{SMR} rounds, the searcher sends a single message that contains the transcripts of all m executions in all ℓ_{SMR} rounds, and the searcher selects $i \in [\ell_{\mathsf{SMR}}]$ uniformly at random and reads the contents of the i^{th} round and the $\ell_{\mathsf{SMR}}+1^{\text{st}}$ (last) round. The searcher rejects if either some of the m transcripts (sent in the $\ell_{\mathsf{SMR}}+1^{\text{st}}$ round) are invalid (e.g., rejecting) or if they do not match the contents of the message sent in the i^{th} round. Otherwise, the solution-merging protocol is applied to the m solutions obtained in the m executions, obtaining a single solution that is output.[8] We stress that solution-merging is applied here to the problem $(Q_{\mathsf{out}}, R_{\mathsf{out}})$ that is solved by the outer (oracle-aided) protocol.

(Our definition of a solution-merging protocol for the D-search problem $(Q_{\mathsf{out}}, R_{\mathsf{out}})$ requires that, for every instance x, if all given solutions are in $Q_{\mathsf{out}}(x)$ then the resulting solution is in $Q_{\mathsf{out}}(x)$, whereas if some of the given solutions are in $R_{\mathsf{out}}(x)$ then the searcher either rejects or the resulting solution is in $R_{\mathsf{out}}(x)$.)

In the soundness analysis (w.r.t. the D-search problem $(Q_{\mathsf{out}}, R_{\mathsf{out}})$ solved by the outer protocol), we consider two cases, while assuming (w.l.o.g.) that the message sent in the $\ell_{\mathsf{SMR}}+1^{\text{st}}$ round is valid (i.e., consists of m proper and non-rejecting transcripts). The first case is that at least ℓ of the first ℓ_{SMR} rounds match the message sent in the $\ell_{\mathsf{SMR}}+1^{\text{st}}$ round. In this case, the m parallel executions (on input x) contain at least one genuine execution (i.e., one that fits the message sent in round $\ell_{\mathsf{SMR}}+1$) that has a corresponding solution in $R_{\mathsf{out}}(x)$. It follows that the solution-merging protocol will either reject or output an element of $R_{\mathsf{out}}(x)$. In the second case (where at least $\ell_{\mathsf{SMR}}-\ell+1$ of the first ℓ_{SMR} rounds do not match the message sent in the $\ell_{\mathsf{SMR}}+1^{\text{st}}$ round), the searcher rejects with probability greater than $1-\frac{\ell}{\ell_{\mathsf{SMR}}}$, which means that the protocol's soundness error is smaller than ℓ/ℓ_{SMR}.

[7] In the latter case, the contents of the $i-1^{\text{st}}$ round is repeated.

[8] Here we assume that the solution-merging protocol uses $O(1)$ rounds.

The second transformation (to SMR protocols) is too complex to describe here (but we will provide one hint). As in the first transformation, for $\ell \ll \ell_{\mathsf{SMR}}$, given an ℓ-round oracle-aided protocol that makes a single oracle call to a D-search problem in each round, we derive an $(\ell_{\mathsf{SMR}}+1)$-round SMR version of soundness error ℓ/ℓ_{SMR}. However, here we use ℓ_{SMR} copies of the original protocol rather than $\binom{\ell_{\mathsf{SMR}}}{\ell}$ copies. On the other hand, we need a more general ($O(1)$-round) solution-merging protocol for the *inner* D-search problems used by the outer system and can handle solutions to different (but related) instances. (We stress that here we use a solution-merging protocol for the *inner* D-search problem, whereas in the first transformation we used a solution-merging protocol for the *outer* D-search problem.)

We now provide one hint on the second transformation. Towards this end, we first present a slightly different version of the first transformation. In this version, we start with a single copy of the original protocol, but clone copies in each of the ℓ_{SMR} subsequent "emulation" rounds. Specifically, in each emulation-round $j \in [\ell_{\mathsf{SMR}}]$, for each copy that has already emulated $i-1$ rounds (of the original protocol), where $i \in [\ell]$, we make two clones of this copy; we let one copy emulate round i (in emulation-round j) and let the other copy remain idle (in emulation-round j).

In contrast, in the second transformation, in each emulation-round $j \in \ell_{\mathsf{SMR}}$, for each $i \in [\ell]$, we create a clone that merges all copies that have completed the emulation of $i-1$ rounds (of the original protocol) and make this cloned copy emulated round i (on a solution created by the solution-merging protocol). In addition, we let all copies (which completed $i-1$ emulation rounds) remain idle (in emulation-round j). Hence, the number of copies that completed i emulation-rounds grows by at most a single unit in each emulation-round. (In contrast, in the first transformation, the number of copies grows by a factor of two in each of the first ℓ_{SMR} emulation-rounds).

Our main focus is on the second transformation, and in this case, we assume that the oracle-aided protocol is "pure" in the sense that it contains a sequence of oracle calls (one per each round), but no "plain" communication rounds. Furthermore, we use the existence of a solution-merging protocol for the D-search problem that is solved by the oracle. Hence, the notion of D-search problems (and solution-merging protocols for them) is essential to our presentation of the second transformation, but it is not essential to the first transformation. Yet, in order to streamline our presentation (in this overview), we also presented the first transformation in terms of oracle-aided protocols, but the same transformation also applies if the original protocol is not pure (i.e., contains plain communication rounds) or, actually, is not even oracle-aided.

1.2 Organization

We begin by defining interactive search systems and their oracle-aided versions in Section 2. Section 3 is then devoted to selective message reading (SMR) search systems, their composition, and the transformations from interactive search systems to SMR ones. In Section 4 we demonstrate the use of the methodologies presented in earlier sections towards constructing an interactive proof system for bounded-depth circuit classes (reestablishing the result of [GKR15]). Lastly, in Section 5 we devise an SMR system for these bounded-depth circuit classes.

2 Interactive search systems

In this section, we present the notion of interactive search systems, which generalize interactive proof systems to the case of non-binary and non-deterministic outputs; a.k.a., search problems. Interactive search systems are central to this work and serve as building blocks in our SMR composition technique and its applications. However, we believe that this notion is important independently of this work, since, a priori, it is unclear whether allowing multiple valid outputs equips these systems with more power than standard interactive proof systems.

We start with defining interactive search systems in Section 2.1. Then, in Section 2.2, we further generalize interactive search systems to oracle-aided interactive search system, which have oracle access to other search problems. In Section 2.3, we present a composition theorem for interactive search systems, which can be viewed as a warm-up towards composing SMR systems.

2.1 Defining interactive search systems

We start by considering the standard notion of a search problem denoted $Q \subseteq \{0,1\}^* \times \{0,1\}^*$ (formalized as a relation). Let $\mathrm{Dom}(Q) := \{x : \exists y \text{ s.t. } (x,y) \in Q\}$. Then Q maps an input $x \in \mathrm{Dom}(Q)$ to the set $Q(x) := \{y : (x,y) \in Q\}$ of all valid solutions for x (with respect to Q). An interactive search system (IS) for the search problem Q is an interactive scheme, where given an input $x \in \{0,1\}^*$, the task is to find a solution in the image $Q(x)$, and reject if no such solution exists (i.e., when $Q(x) = \emptyset$). The system is carried by two parties – a searcher $\mathcal{S}$ and a guide $\mathcal{G}$, analogs of the interactive proof's verifier and prover, respectively. The searcher is a computationally bounded, randomized strategy that, given an input x, aims to find a solution in $Q(x)$. Nevertheless, the searcher is too weak to find such a solution by himself. The guide, on the other hand, is a stronger entity that can find an adequate solution in $Q(x)$ on her own, but is not to be trusted and may try to lead the searcher to output a string outside $Q(x)$. Hence, the searcher must interact with the guide in order to find a valid solution, but do so with care.

To set the requirements for interactive search systems, we slightly relax the definition hinted above and formalize a search problem as a pair relation (Q, R), which we call a dichotomous search problem, abbreviated as a D-search problem. The qualified relation Q captures the essence of the search problem (by mapping inputs $x \in \mathrm{Dom}(Q)$ to the set of corresponding valid solutions $Q(x)$). We use the qualified relation Q to define the system's required functionality when the guide strategy behaves *honestly* (i.e., as instructed by the protocol). The robust relation R, on the other hand, captures the robustness of the system when the guide strategy behaves dishonestly. That is, we define the set of robust inputs $x \in \mathrm{Dom}(R)$ the system can handle in this case, and their corresponding "tolerated" set of solutions $R(x)$. An interactive search system for a D-search problem (Q, R) guarantees that for any qualified input $x \in \mathrm{Dom}(Q)$, if the guide behaves honestly and follows the protocol as required, then the searcher succeeds in outputting a solution in $Q(x)$ (possibly except for a small probability). In addition, for any robust input $x \in \mathrm{Dom}(R)$, even a *malicious and computationally unbounded* guide strategy will not make the searcher output a string outside of the tolerated set of solutions $R(x)$, possibly except for a small probability. In this case, the searcher may either reject (by outputting the special $\perp$ symbol) or

succeed in outputting a tolerated solution in $R(x)$, regardless of the guide's attempts to mislead him. We stress that given $x \in \mathrm{Dom}(Q)$ (resp. $x \in \mathrm{Dom}(R)$), the searcher may not be able to distinguish solutions in $Q(x)$ (resp. in $R(x)$) from those not in $Q(x)$ (resp. not in $R(x)$).

Definition 2.1 (interactive search system (IS))**.** *An interactive scheme $\Pi = (\mathcal{G}, \mathcal{S})$ between a guide strategy $\mathcal{G}$ and a searcher strategy $\mathcal{S}$ is a $(1-\delta)$-complete and $(1-\varepsilon)$-sound* interactive search system (IS) *for a D-search problem (Q, R) if the following hold.*

- ***Completeness:*** *For any qualified input $x \in \mathrm{Dom}(Q)$, the probability that the searcher $\mathcal{S}$ outputs a string in $Q(x)$ after interacting with the guide $\mathcal{G}$ is at least $1-\delta$.*
- ***Soundness:*** *For any robust input $x \in \mathrm{Dom}(R)$ and any (potentially malicious and computationally unbounded) guide strategy $\widetilde{\mathcal{G}}$, the probability that the searcher $\mathcal{S}$ does not reject and outputs a string outside of $R(x)$ when interacting with $\widetilde{\mathcal{G}}$ is at most ε.*

An interactive search system is ideal *if it is both perfectly complete (i.e., 1-complete) and perfectly sound (i.e., 1-sound).*

Note that Definition 2.1 does not specify the computational power of the prescribed parties (i.e., $\mathcal{S}$ and $\mathcal{G}$). This is done intentionally, and the parties' power can vary among applications. However, in this work, we focus on the standard convention by which the searcher is a probabilistic and polynomial-time (PPT) strategy. However, later on in this work, we also consider doubly-efficient interactive search systems, introduced by [GKR15], in which the prescribed (honest) guide strategy is efficient (that is, runs in polynomial time), and the searcher is super-efficient, and runs in almost-linear time. We stress that, even for doubly-efficient ISs, the soundness guarantee must still hold against any (potentially unbounded) guide strategy. More formally,

Definition 2.2 (doubly-efficient IS)**.** *An interactive search system $\Pi = (\mathcal{G}, \mathcal{S})$ is* doubly-efficient *if for any input x of length n, the prescribed (honest) guide runs in time at most $\mathrm{poly}(n)$ and the searcher runs in time at most $n \cdot \mathrm{polylog}(n)$, which we denote by $\widetilde{O}(n)$. No limitations on the guide resources are set for non-honest guide strategies.*

Remark 2.3 (ISs generalize IPs)**.** *As already indicated in the introduction, interactive search systems, as defined in Definition 2.1, generalize interactive proofs. Indeed, any interactive proof system $(\mathcal{P}, \mathcal{V})$ for a set $\mathcal{L}$ can be viewed as an interactive search system for a D-search problem $(Q_{\mathcal{L}}, R_{\mathcal{L}})$, defined by $Q_{\mathcal{L}} = \{(x, 1) : x \in \mathcal{L}\}$ and $R_{\mathcal{L}} = \{(x, 0) : x \notin \mathcal{L}\}$ (when interpreting 1 (resp. 0) as "accept" (resp. "reject")).*

In this work, our focus is on guide-oblivious (specifically, public-coin) interactive search systems, in which the searcher's actions during communication (specifically, the content of his messages) do not depend on the guide's messages. This property is crucial in the SMR setting, where the searcher may not read any guide message during communication. By assuming, without loss of generality, that all of the searcher's actions during communication depend only on the *length* of the input, rather than on the input itself, we extend guide-oblivious interactive search systems to a setting in which either both parties receive the input x at the beginning of the protocol (as in the standard setting), or only the guide receives the explicit input.[9] In contrast, the searcher receives only its length. The protocol is

[9] This extra flexibility will be needed when defining oracle-aided protocols for the SMR setting.

then carried out by having the parties communicate with each other. After the communication ends, the searcher can (reliably) access the input given to the system (and the guide), and decide on his output based on the input and the generated transcript. A special case of guide-oblivious interactive search systems are public coin search systems, in which all of the searcher's randomness is visible to the guide.[10]

Definition 2.4 (guide-oblivious IS)**.** *An interactive search system* $\Pi = (\mathcal{G}, \mathcal{S})$ *is* guide-oblivious *if the searcher's actions during communication do not depend on the guide's messages, nor the input (but may depend on the input's length). In a guide-oblivious system, given an input* x*, the protocol begins with the guide receiving the input* x*, while the searcher either receives* x *or its length* $|x|$*. After the communication ends, the searcher* $\mathcal{S}$ *can access the input* x *and decide on his output based on the input* x *and the generated transcript.*

An guide-oblivious interactive search system is public-coin*, if at each round the searcher sends uniformly distributed strings of a predetermined length.*

2.2 Oracle-aided interactive search systems

We turn to define oracle-aided interactive search systems, which are standard interactive search systems augmented with oracle access to some D-search problems. These systems are best thought of as protocols that can delegate some of their subtasks to an ideal party (represented by the oracle), which provides perfect correctness and requires no computational resources. However, the oracle provides perfect correctness in a setting-dependent sense: when the oracle-aided system Π (that uses the oracle) is considered in the completeness setting (representing honest strategies on a potentially qualified input), the oracle guarantees perfect completeness (and nothing more); when Π is considered in the soundness setting (representing a possibly malicious and computationally unbounded guide strategy on a potentially robust input), the oracle guarantees perfect soundness (and nothing more). The D-search problem for which the oracle is defined determines the set of valid answers for each query and setting.

To formalize this, for each D-search problem (Q, R) we define *two* separate oracles. The first is the qualified oracle, which for each qualified query $\mathsf{q} \in \mathrm{Dom}(Q)$ is guaranteed to return a qualified answer $\mathsf{a} \in Q(\mathsf{q})$, and guarantees nothing on queries outside $\mathrm{Dom}(Q)$. The second is the robust oracle, which for each robust query $\mathsf{q} \in \mathrm{Dom}(R)$ is guaranteed to return a robust answer $\mathsf{a} \in R_{\mathsf{in}}(\mathsf{q})$, and guarantees nothing on queries outside $\mathrm{Dom}(R)$. Which oracle Π receives access to is determined by the setting: In the completeness setting, the oracle-aided system is given the qualified oracle, and in the soundness setting, the system is given the robust oracle. Naturally, the searcher has no way of knowing which of the oracles is being used. We stress that if the qualified (resp. robust) oracle receives a non-qualified (resp. non-robust) query, then the oracle may return an arbitrary answer of her choice.

The notion of oracle-aided protocols has been considered in the context of secure multiparty computation (see [Can00, Gol04]), though in a different sense than ours. First, the oracles considered there are not setting-dependent (i.e., the same oracle is used regardless of the setting) and are defined for randomized functions rather than for relations. Our need for a setting-dependent oracle is a reflection

[10] Indeed, public coin ISs are a special case of guide-oblivious ISs: given the searcher's randomness, the guide can simulate all of the searcher's actions, and hence w.l.o.g. all messages the searcher sends to the guide are his random coins.

of the two different input "flavors" an oracle-aided system may receive (i.e., qualified or robust), and the corresponding correctness guarantees it must provide (completeness or soundness, respectively). This distinction does not arise in the secure multiparty computation setting, where the same type of security is required for every input. Second, we stress that unlike the randomized oracles considered in [Can00, Gol04], the oracles we use in this work do not induce a distribution on the outputs but rather output an arbitrary "valid" element *non-deterministically*. These oracles are useful when one does not care which output one obtains, as long as it satisfies some requirements (formalized by the D-search problem), as is the case in this work.

To formalize the parties' interaction with the oracle, we provide them with four dedicated devices. The first is the query device, which belongs to the guide and is used by her to submit the parties' query to the oracle.[11] Second, each party is provided with an answer device, on which the oracle's answer appears. We assume without loss of generality that the answer includes the query it refers to, so the query is reliably available to both parties. The remaining device is a security device, belonging to the searcher, used to enforce a specific query or query length on the guide (as detailed below).

The query device (which, recall, is a *private* device owned by the guide) serves an additional purpose. When submitting her query, the guide may indicate her preferred answer to this query. Upon receiving a query q and the guide's preferred answer a, the oracle first checks whether the query is valid - that is, if the qualified (resp. robust) oracle is used, then the oracle checks whether q is qualified (resp. robust). If the query is invalid, the oracle complies with the guide's request and returns a, reflecting the oracle's lack of any guarantee for invalid queries. If the query *is* valid, then the oracle checks whether a is itself a valid answer for q. If so, the oracle again complies and returns the guide's preferred answer a. If, however, this is not the case (including the option of the guide requesting the oracle to answer with the rejection symbol $\perp$),[12] then the oracle responds with $\perp$.

It is crucial to note that allowing the guide to choose her preferred answer does not harm the perfect soundness guarantee of the robust oracle. This is since the robust oracle never responds with a non-robust answer to a robust query, and does not provide any guarantee regarding her outputs' distribution. However, enabling the guide to request a specific answer provides the protocol with certain benefits. For example, this mechanism is important for the soundness proof of later protocols (e.g. Theorems 2.8 and 3.4). In addition, this mechanism may also be used by an honest guide strategy in cases where she prefers to proceed according to a certain outcome; for example, in case a certain output increases the protocol's probability of producing a correct output or decreases the parties' running time. An important note is that when the guide chooses an answer, the searcher cannot know that the guide has done so unless the guide tells him that directly. Namely, this information does not appear in the oracle's answer (i.e., is not communicated to the searcher without the guide's will).

Lastly, we turn to define the security device, which, recall, belongs to the searcher. The security device is used by the searcher to either enforce the guide to a specific query or a specific query length, in the natural way. At the beginning of each oracle-calling step, before the oracle's answer is revealed,

[11] We assign the task of submitting the query to the guide (rather than the searcher) since, later in this work, when dealing with SMR systems, the searcher will not be able to read the messages the guide sends him during communication, and hence will not be able to submit queries that depend on certain parts of the transcript.

[12] A reason for this (peculiar) request will be apparent later on, when analyzing the soundness of composed IS and SMR systems (in Section 2.3 and Section 3.2, respectively). Specifically, this enables a guide strategy to emulate another guide strategy; in such cases, issuing this request may help the guide perform the emulation.

the searcher uses his security device to submit either a query $\mathsf{q}_{\mathsf{sec}}$ or a string of the form $\#1^n$ for some integer n (where $\#$ distinguishes $\#1^n$ from the (possibly valid) query 1^n). The oracle then compares this submission against the guide's query q: in the first case, it verifies that $\mathsf{q} = \mathsf{q}_{\mathsf{sec}}$, and in the second that $|\mathsf{q}| = n$, rejecting if either check fails. If no mismatch is detected, the oracle proceeds as described above. We assume, without loss of generality, that the length of the queries is determined by the length of the input given to the system; therefore, the searcher always knows the length of the queries the guide should submit.

We stress that the security device is merely a formal convenience and does not increase the searcher's power.[13] Its sole purpose is to make oracle calls analogous to executions of interactive search systems, enabling us to later replace oracle calls with emulations of actual (inner) systems. Since the oracle query serves as input to the emulated system, the security device allows the searcher to enforce either the entire query or only its length, mirroring the two ways in which a guide-oblivious IS may provide the input to the searcher.

We turn to present the formal definition.

Definition 2.5 (oracle-aided IS)**.** *An* oracle-aided interactive search system *is an interactive search system augmented by* oracle-calling steps *and four* oracle devices*:*

- *A* query device*, belonging to the guide and* hidden from the searcher*.*
- *A* security device*, belonging to the searcher.*
- *Two* answer devices*, one for each of the parties.*

During an oracle-calling step, the guide submits a query q *to the oracle by writing it on her query device. Simultaneously, the searcher submits a* security string s *by writing* s *on his security device. An oracle's answer then appears immediately on both parties' answer devices. The oracle is guaranteed to answer with* $\perp$ *if the guide's query does not match the searcher's security string. That is, if* $\mathsf{s} = \#1^n$ *for some integer* n*, and* $|\mathsf{q}| \neq n$ *and if* s *does not begin with* $\#$ *(so* s *is a query demanded by the searcher), and* $\mathsf{q} \neq \mathsf{s}$*.*

An oracle-aided system Π *is said to* use oracle access to a D-search problem (Q, R) *if its oracle is instantiated as follows: in the completeness setting, the system* Π *is given access to the* qualified oracle *of* (Q, R)*, whose valid queries are those in the qualified domain* $\mathrm{Dom}(Q)$*, and whose valid answers to a query* $\mathsf{q} \in \mathrm{Dom}(Q)$ *are the elements of* $Q(\mathsf{q})$*. In the soundness setting, the system* Π *is given access to the robust oracle of* (Q, R)*, whose valid queries are those in the robust domain* $\mathrm{Dom}(R)$*, and whose valid answers to a query* $\mathsf{q} \in \mathrm{Dom}(R)$ *are the elements of* $R(\mathsf{q})$*. In particular, by definition, this implies that given a qualified (resp. robust) query* q*, the qualified (resp. robust) oracle never returns an answer outside* $Q(\mathsf{q}) \cup \{\perp\}$ *(resp.* $R(\mathsf{q}) \cup \{\perp\}$*).*

Definition 2.6 (additional postulations regarding oracle-aided IS)**.**

- *We assume, without loss of generality, that the query* q *submitted by the guide appears as the prefix of the oracle's answer. We stress that the answer (which includes the query it answers to) is revealed to both parties simultaneously.*

[13] Indeed, the searcher could instead verify from his answer device or the generated transcript that the guide submitted the required query.

- *If the submitted query* q *is invalid according to the oracle, then the oracle may answer with an arbitrary string.*
- *While submitting a query* q*, the guide may write to her (private) query device a preferred answer for* q*. Provided the oracle is not already forced to output* $\perp$ *by the conditions above, the oracle does as follows. If the submitted query* q *is valid (according to the oracle) and the preferred answer* a *is a valid answer for* q*, then the oracle answers with the guide's preferred string* a*. The oracle also answers with* a *when the query* q *is* not *valid. If, however, the query* q *is valid, but* a *is* not *a valid answer for* q*, then the oracle answers* $\perp$*.*
- *We assume that in the completeness case, all the queries submitted to the qualified oracle by the honest guide are in* $\mathrm{Dom}(Q)$*.*
- *When considering the complexity of an oracle-aided system, we distinguish between oracle-calling steps and* communication rounds*, in which the parties exchange messages. We refer to the number of communication rounds as the* communication-round complexity *of the system.*

A few important comments regarding Definition 2.5 are in place. We assume that the oracle answers are recorded in the protocol transcript. This will be especially important when we consider the SMR model (and the analogous PCIPs), in which the searcher has no *online* access to his answer tape. The searcher can, however, access the transcript after the communication ends and read messages and oracle queries of his choice (that were actually submitted to the oracle) along with their corresponding answers.

We allow oracle-aided systems to perform both sequential and parallel oracle calls. By the non-deterministic nature of the oracles we use, the given answers may be either correlated or independent of each other. Moreover, we allow oracle-aided systems to use *multiple* oracles, each answering according to a different D-search problem.

As indicated by Definition 2.5, we assume that the prescribed (honest) guide strategy always submits queries in the qualified relation $\mathrm{Dom}(Q)$. This assumption is natural in many applications, including the ones considered in this work. However, there may be applications where this assumption cannot be made (in which case, one may want to generalize oracle-aided protocols to include this scenario and analyze the completeness case accordingly).

2.3 Composing interactive search systems

Having defined interactive search systems and their oracle-aided analog, we now define their composition. Recall that oracle-aided systems delegate some of their subtasks to an ideal oracle, via oracle calls. Although oracles are a useful conceptual device, they are unavailable in practice. A natural way to eliminate these oracle calls is to replace them with an interactive search system for the same D-search problem. We call this operation composition, denoted $\circ$. Formally, let $\Pi_{\mathsf{out}} = (\mathcal{G}_{\mathsf{out}}, \mathcal{S}_{\mathsf{out}})$ be an oracle-aided interactive search system (referred to as the outer system) for a D-search problem $(Q, R)_{\mathsf{out}}$, using oracle access to a D-search problem $(Q, R)_{\mathsf{in}}$.[14] Let $\Pi_{\mathsf{in}} = (\mathcal{G}_{\mathsf{in}}, \mathcal{S}_{\mathsf{in}})$ be an interactive

[14]For ease of readability, we henceforth denote the D-search problem corresponding to a qualified relation Q_σ and robust relation R_σ by $(Q, R)_\sigma$ rather than (Q_σ, R_σ).

search system for $(Q,R)_{\mathsf{in}}$ (referred as the inner system). The composition $\Pi_{\mathsf{cmp}} := \Pi_{\mathsf{out}} \circ \Pi_{\mathsf{in}}$ is an interactive search system for the D-search problem $(Q,R)_{\mathsf{out}}$ *with no oracle calls to* $(Q,R)_{\mathsf{in}}$, defined as follows.

Given an input x, the composed guide $\mathcal{G}_{\mathsf{cmp}}$ emulates the outer guide strategy $\mathcal{G}_{\mathsf{out}}$ on x whereas the composed searcher $\mathcal{S}_{\mathsf{cmp}}$ emulates the outer searcher $\mathcal{S}_{\mathsf{out}}$ over the input he received, which is either x or $|x|$. The parties then emulate their respective strategies until they reach an oracle-calling step for $(Q,R)_{\mathsf{in}}$. At this point, rather than querying the oracle, the composed strategies $\mathcal{G}_{\mathsf{cmp}}$ and $\mathcal{S}_{\mathsf{cmp}}$ emulate the inner system Π_{in}: First, the composed guide $\mathcal{G}_{\mathsf{cmp}}$ sends the searcher $\mathcal{S}_{\mathsf{cmp}}$ the query q intended for the oracle. We call this round (with an empty searcher message) a query-announcement round. Strategies $\mathcal{G}_{\mathsf{cmp}}$ and $\mathcal{S}_{\mathsf{cmp}}$ then emulate the inner system Π_{in} on q to completion. Denoting the output of the computation $\Pi_{\mathsf{in}}(\mathsf{q})$ by a, the composed strategies then resume emulation of the outer system Π_{out}, treating q as the oracle query and a as the corresponding oracle's answer, and continue until the next oracle-calling step, repeating this substitution each time.

Remark 2.7. *Note that in case the composed system is carried by a dishonest guide strategy* $\widetilde{\mathcal{G}}_{\mathsf{cmp}}$*, at least one of the emulations of the inner system* Π_{in} *can result in the searcher* ($\mathcal{S}_{\mathsf{in}}$*, emulated by* $\mathcal{S}_{\mathsf{cmp}}$) *rejecting, causing the output* a *of this invocation to be* $\perp$*. This scenario does not require special treatment since the outer (oracle-aided) system* Π_{out} *already deals with the analogous situation of the oracle answering with* $\perp$ *(which occurs if in the completeness (resp. soundness) setting, the submitted query* q *is qualified (resp. robust), and the guide requests an answer outside* $Q_{\mathsf{in}}(\mathsf{q})$ *(resp.* $R_{\mathsf{in}}(\mathsf{q})$*), if the guide requests* $\perp$*, or if the searcher's security string is inconsistent with* q*).*

The following theorem provides some properties of composed interactive search systems.

Theorem 2.8 (composition theorem for ISs)**.** *Let* $\Pi_{\mathsf{out}} = (\mathcal{G}_{\mathsf{out}}, \mathcal{S}_{\mathsf{out}})$ *be a* $(1-\delta_{\mathsf{out}})$*-complete and* $(1-\varepsilon_{\mathsf{out}})$*-sound oracle-aided interactive search system for a D-search problem* $(Q,R)_{\mathsf{out}}$ *that uses oracle access to a D-search problem* $(Q,R)_{\mathsf{in}}$ *by calling* κ *parallel copies of* $(Q,R)_{\mathsf{in}}$ *for* τ *sequential times. Let* $\Pi_{\mathsf{in}} = (\mathcal{G}_{\mathsf{in}}, \mathcal{S}_{\mathsf{in}})$ *be a* $(1-\delta_{\mathsf{in}})$*-complete and* $(1-\varepsilon_{\mathsf{in}})$*-sound interactive search system for* $(Q,R)_{\mathsf{in}}$*. Then, their composition* $\Pi_{\mathsf{cmp}} := \Pi_{\mathsf{out}} \circ \Pi_{\mathsf{in}}$ *is a* $(1-\delta_{\mathsf{out}}-\kappa\cdot\tau\cdot\delta_{\mathsf{in}})$*-complete and* $(1-\varepsilon_{\mathsf{out}}-\kappa\cdot\tau\cdot\varepsilon_{\mathsf{in}})$*-sound interactive search system for* $(Q,R)_{\mathsf{out}}$*.*

When analyzing the composition of standard (non-SMR) search systems, there is no meaningful distinction between parallel and sequential oracle calls - each call contributes independently to the completeness and soundness error regardless. We nonetheless make this distinction explicit because it matters when composing SMR systems. There, the soundness error is *independent* of the number of sequential oracle calls, depending instead on the *reads* performed by the searcher and the number of parallel calls. In contrast, the completeness error behaves as in the standard (IS) case.

Proof. **Completeness.** Fix a qualified input $x \in \mathrm{Dom}(Q_{\mathsf{out}})$. Recall that by the definition of oracle-aided systems (see Definition 2.5), the prescribed (honest) guide strategy $\mathcal{G}_{\mathsf{out}}$ always submits the oracle $(Q,R)_{\mathsf{in}}$ qualified queries, (i.e., queries in $\mathrm{Dom}(Q_{\mathsf{in}})$). Therefore, each invocation of Π_{in} within the composed system Π_{cmp} is emulated on valid inputs (which the system Π_{in} is designed to receive in the completeness case). Let $E_{\mathsf{in}}^{\mathsf{complete}}$ denote the event that each invocation of Π_{in}, occurring during the execution of $\Pi_{\mathsf{cmp}}(x)$, is successful (as required in the completeness condition). Note that conditioning

on $E_{\mathsf{in}}^{\mathsf{complete}}$, the outer system Π_{out} cannot distinguish between its oracle-calling steps being performed by Π_{in} (as in Π_{cmp}) or by direct oracle access to $(Q,R)_{\mathsf{in}}$ (as in the original execution of Π_{out}). Hence, the probability of the search failing in this case is at most

$$\Pr\Big[\,(\mathcal{G}_{\mathsf{cmp}},\mathcal{S}_{\mathsf{cmp}})(x)\notin Q_{\mathsf{out}}(x)\;\Big|\;E_{\mathsf{in}}^{\mathsf{complete}}\,\Big]=\Pr[(\mathcal{G}_{\mathsf{out}},\mathcal{S}_{\mathsf{out}})(x)\notin Q_{\mathsf{out}}(x)]\leq\delta_{\mathsf{out}},$$

where the probability is over the searchers' random coins. By the $(1-\delta_{\mathsf{in}})$-completeness of Π_{in} and a union bound, the probability that $E_{\mathsf{in}}^{\mathsf{complete}}$ fails to occur is at most $\kappa\cdot\tau\cdot\delta_{\mathsf{in}}$. Hence, the probability that the searcher $\mathcal{S}_{\mathsf{cmp}}$ of the composed system fails (i.e., outputs a string outside $Q_{\mathsf{out}}(x)$) is at most $\delta_{\mathsf{out}}+\kappa\cdot\tau\cdot\delta_{\mathsf{in}}$.

Soundness. Fix a robust input $x\in\mathrm{Dom}(R_{\mathsf{out}})$ and assume for contradiction that there exists a guide strategy $\widetilde{\mathcal{G}}_{\mathsf{cmp}}$ that causes the searcher $\mathcal{S}_{\mathsf{cmp}}$ to not reject and output a string outside $R_{\mathsf{out}}(x)$ with probability greater than $\varepsilon_{\mathsf{out}}+\kappa\cdot\tau\cdot\varepsilon_{\mathsf{in}}$. We construct a guide strategy $\widetilde{\mathcal{G}}_{\mathsf{out}}$ that violates the soundness of the outer system Π_{out}.

The outer guide $\widetilde{\mathcal{G}}_{\mathsf{out}}$ aims to emulate the composed guide $\widetilde{\mathcal{G}}_{\mathsf{cmp}}$ on x, determining her own actions according to those of $\widetilde{\mathcal{G}}_{\mathsf{cmp}}$. Within the outer protocol Π_{out}, this is straightforward: the outer guide $\widetilde{\mathcal{G}}_{\mathsf{out}}$ relays messages between the composed guide $\widetilde{\mathcal{G}}_{\mathsf{cmp}}$ and the searcher $\mathcal{S}_{\mathsf{out}}$. The subtlety arises at oracle-calling steps.

Determining the queries (to the κ parallel copies of $(Q,R)_{\mathsf{in}}$) themselves is easy - the outer guide $\widetilde{\mathcal{G}}_{\mathsf{out}}$ advances the emulation of the composed guide $\widetilde{\mathcal{G}}_{\mathsf{cmp}}$ by one additional round, which is the query-announcement round (without sending anything to $\mathcal{S}_{\mathsf{out}}$), and sets her queries to be the strings $\mathsf{q}_1,\ldots,\mathsf{q}_\kappa$ that the composed guide $\widetilde{\mathcal{G}}_{\mathsf{cmp}}$ sends. However, knowing the queries alone is not enough, since each query may admit multiple valid answers, and different answers may lead the composed guide $\widetilde{\mathcal{G}}_{\mathsf{cmp}}$ to a different behavior in subsequent rounds. Indeed, due to the non-deterministic nature of the oracle, even an unbounded outer guide $\widetilde{\mathcal{G}}_{\mathsf{out}}$ cannot simply submit the queries to the oracle, receive an arbitrary valid sequence of answers, and then *force* the emulation of $\widetilde{\mathcal{G}}_{\mathsf{cmp}}$ to continue consistently *with those answers*. Instead, before submitting her queries, the outer guide $\widetilde{\mathcal{G}}_{\mathsf{out}}$ first interacts with the composed guide $\widetilde{\mathcal{G}}_{\mathsf{cmp}}$ (while sending nothing to $\mathcal{S}_{\mathsf{out}}$), playing the role of the composed searcher $\mathcal{S}_{\mathsf{cmp}}$ through the inner system, until the answer for each of the κ copies of Π_{in} is determined. Let a_i denote the answer for q_i.

Submitting to the oracle. Having determined $\mathsf{a}_1,\ldots,\mathsf{a}_\kappa$, the outer guide $\widetilde{\mathcal{G}}_{\mathsf{out}}$ performs the oracle-calling step by submitting each query q_i together with the requested answer a_i to the i^{th} copy of the oracle $(Q,R)_{\mathsf{in}}$. If each a_i is valid for q_i — that is, $\mathsf{a}_i\in R_{\mathsf{in}}(\mathsf{q}_i)$ whenever $\mathsf{q}_i\in\mathrm{Dom}(R_{\mathsf{in}})$ (with no constraint for queries outside $\mathrm{Dom}(R_{\mathsf{in}})$) — then the robust oracle complies with each request, and $\widetilde{\mathcal{G}}_{\mathsf{out}}$ can resume her interaction with $\mathcal{S}_{\mathsf{out}}$ using $\widetilde{\mathcal{G}}_{\mathsf{cmp}}$'s messages.

Analyzing the probability of success for the outer guide is now straightforward. If each of the emulations of the inner system Π_{in} resulted in a valid answer for its input, then the outer guide $\widetilde{\mathcal{G}}_{\mathsf{cmp}}$ perfectly emulates the composed guide $\widetilde{\mathcal{G}}_{\mathsf{out}}$; hence, given this event, both strategies have the same success probability. On the other hand, by definition of oracle-aided systems, the probability that there exists an emulation of the inner system which resulted with an non-valid output (with respect to

its corresponding output) equals the probability that a certain emulation of the inner system received a robust input $\mathsf{q} \in R_{\mathsf{in}}(\mathsf{q})$, and yet resulted with an output a outside $R_{\mathsf{in}}(\mathsf{q})$. By the $(1-\varepsilon_{\mathsf{in}})$-soundness of the inner system and a union bound, the probability of this event is at most $\kappa \cdot \tau \cdot \varepsilon_{\mathsf{in}}$, and therefore the probability that the outer guide $\widetilde{\mathcal{G}}_{\mathsf{out}}$ succeeds in fooling the searcher $\mathcal{S}_{\mathsf{out}}$ (by making $\mathcal{S}_{\mathsf{out}}$ not reject and output a string outside $R_{\mathsf{out}}(x)$) is greater than $\varepsilon_{\mathsf{out}} + \kappa \cdot \tau \cdot \varepsilon_{\mathsf{in}} - \kappa \cdot \tau \cdot \varepsilon_{\mathsf{in}} = \varepsilon_{\mathsf{out}}$, hence contradicting the $(1-\varepsilon_{\mathsf{out}})$-soundness of Π_{out}. □

3 Selected message reading (SMR) systems

In this section, we consider a PCP-motivated version of interactive search systems called selected message reading interactive search systems (SMR-IS). In this model, the searcher is further restricted and allowed to read only a few of the guide's messages sent to him; he must decide his output based solely on these messages. (Note that, unlike PCIPs, in SMR-IS systems, the selected messages are entirely revealed to the searcher.)

We begin by defining SMR-IS systems. Then, in Section 3.2, we prove a composition theorem for SMR-ISs: we show that given an *oracle-aided* SMR-IS for a D-search problem $(Q, R)_{\mathsf{out}}$ that uses oracle access to a D-search problem $(Q, R)_{\mathsf{in}}$, and given an SMR-IS system for $(Q, R)_{\mathsf{in}}$, the two systems can be composed to yield an SMR-IS for $(Q, R)_{\mathsf{out}}$. Last, in Section 3.3, we present two transformations of ordinary IS systems into SMR-IS ones.

3.1 Defining SMR-IS systems

Loosely speaking, an SMR-IS is a *guide-oblivious* interactive search system in which the searcher has limited access to the guide's messages. We restrict the definition to guide-oblivious systems because the searcher of an SMR-IS must send his messages before he has read *any* message the guide has sent him. Only after the interaction ends can the searcher choose which of the guide's messages he would like to read. In the case of guide-oblivious systems, the searcher has no problem doing so, where we assume w.l.o.g., all other parameters (such as message length and content, round complexity, the location of oracle-calling steps, etc.) are determined by the input's length alone. Because at the system's instantiation, the searcher either receives the input itself or its length (see Definition 2.4), the searcher has no problem communicating with the guide while reading no messages.

Similarly to PCIPs, we formalize the notion of SMR-IS systems by partitioning the protocol into three phases, each dedicated to a different searcher action. As in any interactive search system, the system begins with the guide receiving an input x, and the searcher either receives the input x or its length $|x|$. The *first phase* is dedicated to communication – the parties send each other messages, while the searcher is forbidden from reading any of the messages sent by the guide (nor the input x, unless it was given to him at the beginning of the system). We stress that no further communication will occur after this phase. The *second phase* is dedicated to reading – the searcher can now choose which of the guide's messages he would like to read. For that, the searcher can use fresh randomness, and he is allowed to choose these messages adaptively. The number of messages that the searcher reads is defined as the reading complexity of the system. In addition, the searcher receives the *input* x provided to the

system. Then, in the *last phase*, the searcher needs to decide on its output based on the information he obtained. Namely, the input given to the system, his own messages and random coins, and the guide's messages he read in the reading phase.

Remark 3.1 (comparison with similar models)**.** *We highlight a few differences between SMR-IS systems and other similar models that appear in the literature. Unlike PCIPs, in which the verifier selects which* bits *from the prover's messages he would like to read, in SMR-IS systems, the messages are considered as a single unit, and a chosen message is revealed to the searcher in its entirety. However, one may view this model as an intermediate step and easily generalize it further to a bit-selecting model (i.e., PCIPs or their search analog, PCISs). See* [ACY22a] *for such generalization techniques.*

We further comment that in the SMR model of [ACY22b]*, the verifier is also "charged" for reading his own sent messages. One can transform the SMR model we consider into the one considered in* [ACY22b] *via (a trivial generalization of) the transformation of* [ACY22a].

The formal definition of SMR-IS systems is given below. Throughout this work, we denote SMR-IS systems by the letter Σ (rather than Π, which is used for general interactive search systems).

Definition 3.2 (selected message reading IS (SMR-IS))**.** *A $(1-\delta)$-complete and $(1-\varepsilon)$-sound guide-oblivious interactive search system $\Sigma = (\mathcal{G}, \mathcal{S})$ for a D-search problem (Q, R) is a $(1-\delta)$-complete and $(1-\varepsilon)$-sound* selected message reading interactive search system (SMR-IS) *for (Q, R) if the system consists of the following three phases. Given an input $x \in \{0,1\}^*$, the system is instantiated with the guide $\mathcal{G}$ receiving the input x, and the searcher $\mathcal{S}$ either receives the input x or its length $|x|$.*

1. ***Communication phase:*** *The guide $\mathcal{G}$ and the searcher $\mathcal{S}$ communicate with each other. In this phase, the searcher reads none of the messages he receives from the guide.*

2. ***Reading phase:*** *The searcher $\mathcal{S}$ receives the input x and chooses which of the guide's messages to read. These messages can be chosen randomly and adaptively. Each message the searcher $\mathcal{S}$ chooses to read is entirely revealed to him.*

3. ***Decision phase:*** *The searcher $\mathcal{S}$ decides on his output based on the input x, the selected guide's messages, and his own messages and random coins. The searcher may use fresh random coins for his decision. The decided answer satisfies the $(1-\delta)$-completeness and $(1-\varepsilon)$-soundness conditions of Definition 2.1.*

The number of guide's messages that the searcher $\mathcal{S}$ reads (in the reading phase) is the reading complexity *of the system.*

We further generalize the above notion by defining oracle-aided SMR-IS systems: these are standard SMR-ISs augmented with oracle devices and oracle-calling steps, analogously to Definition 2.5. In this case, the communication phase, which now includes oracle-calling steps, is conducted with the searcher *not reading his (answer) oracle device* (hence the searcher also does not know the queries that were actually submitted to the oracle), in addition to reading none of the messages the guide sends. Then, in the reading phase, the searcher can choose which oracle answers (along with their corresponding queries) and guide messages he wants to read, which are then revealed to him. Recall

that by Definition 2.5, any response made by the oracle includes both the submitted query and the answer given to it. We distinguish the two types of readings the searcher performs. We define the number of guide's messages the searcher reads as the protocol's message-reading complexity, and the number of oracle answers the searcher reads to be the protocol's oracle-reading complexity. Then, in the decision phase, the searcher needs to decide what string to output based on all the data he obtained.

3.2 Composition theorem for SMR-IS systems

We now present the SMR-IS composition, obtained by applying the approach we used to compose interactive search systems (see Section 2.3) to the SMR regime.

Some issues should be discussed before presenting the composition construction. The most important issue is that the outputs of the inner systems are determined only during their decision phase. Indeed, to decide on his output for an invocation of the inner system, the searcher may need to read the input given to that invocation along with some guide messages, yet he can only do so during the reading phase (of the composed system), well after the communication phase (of the composed system) has ended. This creates a potential problem: the guide may need to know the inner systems' outputs in order to continue the interaction in subsequent rounds, but those outputs are not determined by the transcript of the inner system, but may rather depend also on the random coins (which may influence both which messages he reads and his final decision). The outputs may therefore be unknown even to an unbounded guide. We resolve this problem by "immediately" providing the guide (of the composed system) with the random coins that the searcher will use during the reading and decision phases (if the searcher will later choose to check this invocation of the inner system). Specifically, we augment each invocation of the inner system with a dedicated decision-emulation round, in which the searcher sends two random strings: one to serve as his random coins for the corresponding reading phase, and one for the corresponding decision phase. Having this information, the guide can determine the searcher's output for this invocation of the inner system.

In addition, recall that in composed interactive search systems, the parties start by emulating the outer system. When they reach an oracle-calling step, they perform a full emulation of the inner system on the query intended for the oracle. In the case of SMR-IS systems, the searcher cannot know the input to the inner system (or, more precisely, its length), which may affect his messages as well as the inner system's round complexity. We solve this technicality by assuming that the length of the input given to an oracle-aided system determines the length of the queries submitted during its invocation.

We are now ready to define the composition of SMR-IS systems formally.

Construction 3.3 (composing SMR-ISs)**.** *Let $\Sigma_{\mathsf{out}} = (\mathcal{G}_{\mathsf{out}}, \mathcal{S}_{\mathsf{out}})$ be an oracle-aided SMR-IS system for a D-search problem $(Q,R)_{\mathsf{out}}$ that uses an oracle access to a D-search problem $(Q,R)_{\mathsf{in}}$ by calling κ parallel copies of $(Q,R)_{\mathsf{in}}$ for τ sequential times. Let $\Sigma_{\mathsf{in}} = (\mathcal{G}_{\mathsf{in}}, \mathcal{S}_{\mathsf{in}})$ be an SMR-IS system for $(Q,R)_{\mathsf{in}}$. The composition of Σ_{out} and Σ_{in}, denoted $\Sigma_{\mathsf{cmp}} = (\mathcal{G}_{\mathsf{cmp}}, \mathcal{S}_{\mathsf{cmp}}) := \Sigma_{\mathsf{out}} \circ \Sigma_{\mathsf{in}}$, is defined as follows. For an input $x \in \{0,1\}^*$, the system is instantiated with the guide $\mathcal{G}_{\mathsf{cmp}}$ receiving the input x, and the searcher $\mathcal{S}_{\mathsf{cmp}}$ either receives the input x or its length $|x|$.*

1. ***Communication phase:*** *The composed guide $\mathcal{G}_{\mathsf{cmp}}$ emulates the outer guide $\mathcal{G}_{\mathsf{out}}$ on x, and the composed searcher $\mathcal{S}_{\mathsf{cmp}}$ emulates the outer searcher $\mathcal{S}_{\mathsf{out}}$ on the input he received, which is either*

x *or* $|x|$. *For* $i = 1, \ldots, \tau$*:*

(*a*) *Upon reaching the* i^{th} *oracle-calling step, let* q_j^i *denote the query that the outer guide* $\mathcal{G}_{\mathsf{out}}$ *intends to submit to the* j^{th} *copy of the oracle* $(Q,R)_{\mathsf{in}}$. *The guide* $\widetilde{\mathcal{G}}_{\mathsf{cmp}}$ *and searcher* $\mathcal{S}_{\mathsf{cmp}}$ *pause their emulation of the outer system* Σ_{out} *and proceed as follows.*

i. ***Query-announcement round:*** *The composed guide* $\mathcal{G}_{\mathsf{cmp}}$ *performs a* query-announcement round (*as in the non-SMR setting*), *by sending the queries* $\mathsf{q}_1^i, \ldots, \mathsf{q}_\kappa^i$ *to the searcher* $\mathcal{S}_{\mathsf{cmp}}$ (*note however that the searcher* $\mathcal{S}_{\mathsf{cmp}}$ *does not read these queries at this point*).

ii. ***Communication:*** *The composed guide* $\mathcal{G}_{\mathsf{cmp}}$ *and searcher* $\mathcal{S}_{\mathsf{cmp}}$ *then begin emulating the* communication phase *of* κ *independent parallel copies of the inner system* Σ_{in}, *where the* j^{th} *copy is invoked on the input* q_j^i. *Specifically, the composed guide* $\mathcal{G}_{\mathsf{cmp}}$ *emulates the* j^{th} *copy of the inner guide* $\mathcal{G}_{\mathsf{in}}$ *on* q_j^i. *The composed searcher* $\mathcal{S}_{\mathsf{cmp}}$ *emulates the* j^{th} *copy of the inner searcher* $\mathcal{S}_{\mathsf{in}}$ *on* $|\mathsf{q}_j^i|$ (*as the composed searcher* $\mathcal{S}_{\mathsf{cmp}}$ *searches cannot read the guide messages, hence may not know* q_j^i).

iii. ***Decision-emulation round:*** *Upon completion of the emulation of the communication phase of* Σ_{in}, *the composed searcher* $\mathcal{S}_{\mathsf{cmp}}$ *carry out a* decision-emulation round *by sending two uniformly random string sequences,* $(\mathsf{r}_{\mathsf{read}}{}_1^i, \ldots, \mathsf{r}_{\mathsf{read}}{}_\kappa^i)$ *and* $(\mathsf{r}_{\mathsf{dec}}{}_1^i, \ldots, \mathsf{r}_{\mathsf{dec}}{}_\kappa^i)$, *to be used as his random coins in the reading and decision phases of* Σ_{in}, *respectively, should he later choose to emulate those phases for this* (i^{th}) *parallel invocation of* Σ_{in}. *The composed guide* $\mathcal{G}_{\mathsf{cmp}}$ *emulates the reading and decision phases of the inner searcher* $\mathcal{S}_{\mathsf{in}}$ *on each resulting inner system transcript, using* $\mathsf{r}_{\mathsf{read}}{}_j^i$ *and* $\mathsf{r}_{\mathsf{dec}}{}_j^i$ *as the random coins for the reading and decision phases of the* j^{th} *copy, respectively. Let* a_j^i *denote the* $\mathcal{S}_{\mathsf{in}}$*'s decision* (*output*) *for the* j^{th} *copy of* Π_{in}.

(*b*) *The composed strategies* $\mathcal{G}_{\mathsf{cmp}}$ *and* $\mathcal{S}_{\mathsf{cmp}}$ *resume the emulation of the outer system* Σ_{out}, *where the guide* $\mathcal{G}_{\mathsf{cmp}}$ *treats* q_j^i *as the query submitted to the* j^{th} *copy of the oracle* $(Q,R)_{\mathsf{in}}$ *in this* (i^{th}) *oracle-calling step, and* a_j^i *as the oracle's corresponding answer.*

Strategies $\mathcal{G}_{\mathsf{cmp}}$ *and* $\mathcal{S}_{\mathsf{cmp}}$ *continue to emulate the communication phase of the outer system* Σ_{out} *as above until completion.*

2. ***Reading phase:*** *The composed searcher* $\mathcal{S}_{\mathsf{cmp}}$ *receives the input* x *given to the composed system and emulates the reading phase of the outer searcher* $\mathcal{S}_{\mathsf{out}}$ *as follows.*

(*a*) *The composed guide* $\mathcal{S}_{\mathsf{cmp}}$ *provides* $\mathcal{S}_{\mathsf{out}}$ *with* x *as the input of the outer system* Π_{out}.

(*b*) *Whenever the outer searcher* $\mathcal{S}_{\mathsf{out}}$ *requests to read the* i^{th} *oracle-calling step, strategy* $\mathcal{S}_{\mathsf{cmp}}$ *starts by reading the corresponding query-announcement round. Letting* $(\widetilde{\mathsf{q}}_1^i, \ldots, \widetilde{\mathsf{q}}_\kappa^i)$ *denoting the read queries* (*where allegedly,* $\widetilde{\mathsf{q}}_j^i = \mathsf{q}_j^i$ *for each* $j \in [\kappa]$), *the composed searcher* $\mathcal{S}_{\mathsf{cmp}}$ *does not yet "reveal" the queries to the outer searcher* $\mathcal{S}_{\mathsf{out}}$, *but rather fully emulates the* reading *and* decision *phases of the inner searcher* $\mathcal{S}_{\mathsf{in}}$ *for each copy of* Σ_{in}, *providing the* j^{th} *copy of the inner searcher* $\mathcal{S}_{\mathsf{in}}$ *with* $\widetilde{\mathsf{q}}_j^i$ *as the system's input and* $\mathsf{r}_{\mathsf{read}}{}_j^i$ (*resp.* $\mathsf{r}_{\mathsf{dec}}{}_j^i$) *as* $\mathcal{S}_{\mathsf{in}}$*'s random coins for the reading* (*resp. decision*) *phase. Letting* a_j^i *denote* $\mathcal{S}_{\mathsf{in}}$*'s output for the* j^{th} *copy*

of Σ_{in}, the composed searcher now "reveals" the "oracle answer" to the outer searcher $\mathcal{S}_{\mathsf{out}}$, using $\widetilde{\mathsf{q}}_j^i$ as the submitted query and a_j^i as the corresponding oracle answer.[15]

(c) *Whenever the outer searcher $\mathcal{S}_{\mathsf{out}}$ requests to read an ordinary message from the guide $\mathcal{G}_{\mathsf{out}}$, strategy $\mathcal{S}_{\mathsf{cmp}}$ reads the corresponding message from the transcript and "reveals" it to $\mathcal{S}_{\mathsf{out}}$.*

3. ***Decision phase**: The composed searcher $\mathcal{S}_{\mathsf{cmp}}$ emulates the decision phase of the outer searcher $\mathcal{S}_{\mathsf{out}}$ over the read oracle-calling steps, the corresponding read guide messages, the input x, the random coins $\mathcal{S}_{\mathsf{cmp}}$ tossed during the outer system's emulation, and $\mathcal{S}_{\mathsf{cmp}}$ messages. The composed searcher $\mathcal{S}_{\mathsf{cmp}}$ outputs the same string as the outer searcher $\mathcal{S}_{\mathsf{out}}$.*

We are now ready to state our main composition theorem.

Theorem 3.4 (composition theorem for SMR-IS systems)**.** *Let $\Sigma_{\mathsf{out}} = (\mathcal{G}_{\mathsf{out}}, \mathcal{S}_{\mathsf{out}})$ be a $(1-\delta_{\mathsf{out}})$-complete and $(1-\varepsilon_{\mathsf{out}})$-sound oracle-aided SMR-IS system for a D-search problem $(Q,R)_{\mathsf{out}}$ that uses an oracle access to a D-search $(Q,R)_{\mathsf{in}}$ by calling κ parallel copies of $(Q,R)_{\mathsf{in}}$ for τ sequential times. Let $\Sigma_{\mathsf{in}} = (\mathcal{G}_{\mathsf{in}}, \mathcal{S}_{\mathsf{in}})$ be a $(1-\delta_{\mathsf{in}})$-complete and $(1-\varepsilon_{\mathsf{in}})$-sound SMR-IS for $(Q,R)_{\mathsf{in}}$. Then, letting $\rho^{\mathcal{O}}_{\mathsf{out}}$ denote the* oracle-*reading complexity of $\mathcal{S}_{\mathsf{out}}$,*[16] *the composed system $\Sigma_{\mathsf{cmp}} = (\mathcal{G}_{\mathsf{cmp}}, \mathcal{S}_{\mathsf{cmp}}) := \Sigma_{\mathsf{out}} \circ \Sigma_{\mathsf{in}}$ is a $(1-\delta_{\mathsf{out}}-\tau\cdot\kappa\cdot\delta_{\mathsf{in}})$-complete and $(1-\varepsilon_{\mathsf{out}}-\rho^{\mathcal{O}}_{\mathsf{out}}\cdot\kappa\cdot\varepsilon_{\mathsf{in}})$-sound SMR-IS for $(Q,R)_{\mathsf{out}}$. In addition, letting $\rho^{\mathcal{G}}_{\mathsf{out}}$ denote the* message-*reading complexity of $\mathcal{S}_{\mathsf{out}}$*[17] *and ρ_{in} denote the (overall) reading complexity of $\mathcal{S}_{\mathsf{in}}$ (of oracle-calling steps + ordinary guide messages), the composed system has reading complexity $\rho^{\mathcal{G}}_{\mathsf{out}} + \rho^{\mathcal{O}}_{\mathsf{out}} \cdot (\rho_{\mathsf{in}} + 1)$.*

Both the completeness and soundness proofs of Theorem 3.4 follow very similar lines to their non-SMR analogs in the proof of Theorem 2.8, with natural modifications due to the slightly different structure of composed SMR systems from non-SMR ones. The $(1-\varepsilon_{\mathsf{out}}-\rho^{\mathcal{O}}_{\mathsf{out}}\cdot\kappa\cdot\varepsilon_{\mathsf{in}})$-soundness error follows from essentially the same analysis as its non-SMR analog, together with the observation in Remark 3.5.

Remark 3.5. *Note the asymmetry in the error bounds that Theorem 3.4 guarantees. In the completeness error, the error term δ_{in} accumulates with every* invocation *of the inner protocol. In the soundness error, however, an additional $\varepsilon_{\mathsf{in}}$ term is incurred only for invocations of Π_{in} that the searcher $\mathcal{S}_{\mathsf{cmp}}$ actually* checks *during his reading and decision phases. This asymmetry arises from a fundamental difference between the completeness and soundness settings. In the completeness setting, simulating a transcript that could result from an interaction with the honest outer guide $\mathcal{G}_{\mathsf{out}}$ requires* all *invocations of the inner system Σ_{in} to be complete. In the soundness setting, by contrast, invocations that the composed searcher $\mathcal{S}_{\mathsf{cmp}}$ does not check need not be sound, because anyhow, a malicious composed guide $\widetilde{\mathcal{G}}_{\mathsf{cmp}}$ could continue the subsequent rounds according to an arbitrary output of her choice (and gone undetected) just as the outer malicious guide $\widetilde{\mathcal{G}}_{\mathsf{out}}$ can lie about oracle answers that $\mathcal{S}_{\mathsf{out}}$ does not read.*

[15] Note that the composed searcher $\mathcal{S}_{\mathsf{cmp}}$ cannot postpone the decision phase of the inner searcher $\mathcal{S}_{\mathsf{in}}$ to his own ($\mathcal{S}_{\mathsf{cmp}}$'s) decision phase, as $\mathcal{S}_{\mathsf{cmp}}$ needs to provide the outer searcher $\mathcal{S}_{\mathsf{out}}$ with the "oracle answer" in real-time (now replaced with the inner searcher's decision).

[16] Recall that the oracle-reading complexity of an oracle-aided SMR-IS is the number of oracle-calling steps that the searcher reads during the reading phase.

[17] Recall that the message-reading complexity of an oracle-aided SMR-IS is the number of ordinary guide messages that the searcher reads during the reading phase.

3.3 Transforming IS into SMR-IS

We show how to transform two natural subclasses of interactive search systems into SMR-ISs with low reading complexity (independent of the number of rounds in the original IS). Both transformations extend the constructions of [ACY22b] to the context of interactive search systems, while improving the soundness analysis of one of them.

The first (basic) transformation applies to interactive search systems with $O(\log n)$ rounds that admit a certain merging mechanism (see Section 3.3.3). The second transformation extends the basic construction to systems of arbitrary round complexity, albeit of a particular structure, and achieves an exponentially smaller soundness error (see Section 3.3.4).

3.3.1 Construction Overview

The transformation from IS to SMR-IS addresses two issues: The first is enabling selective reading of the guide's messages (also in the context of an interactive proof system).[18] The approach taken to address this issue (which involves multiple executions of the original protocol) gives rise to a second challenge: generating a final solution from a list of candidate solutions (which is trivial in the context of IPs).

Enabling selective reading in interactive protocols (for ISs and IPs). The naive solution of transforming an arbitrary ℓ-round interactive protocol into a selective reading protocol is to augment the original protocol by an additional round in which the guide/prover sends the entire transcript. The searcher/verifier reads this message (i.e., the transcript) and checks that the transcript is valid/accepting. In addition, the searcher/verifier selects at random a round $i \in [\ell]$ and checks that the transcript is consistent with the message sent by the guide/prover at round i.

This works relatively well, provided that, when communicating with a cheating searcher/prover, the transcript disagrees with the actual messages sent in at least half of the rounds (where the constant half is used for simplicity and can be replaced by an arbitrary positive constant, which only affects the constant soundness error). Unfortunately, an arbitrary ℓ-round interaction can be foiled even when the transcript disagrees with the actual interaction on a single round. Hence, the main challenge is transforming arbitrary ℓ-round interactions to $O(\ell)$-round interactions in which cheating requires providing a transcript that disagrees with the actual interaction on most rounds.

Consider a 2ℓ-round proctocol that consists of running, in parallel, $\binom{2\ell}{\ell}$ copies of the original ℓ-round protcol such that each copy uses only ℓ of the 2ℓ rounds (and is idle in the other ℓ rounds). Of course, each of these copies is associated with a different choice of an ℓ-subset $I \in \binom{[2\ell]}{\ell}$. Now, for every choice of ℓ rounds, there exists a copy that was not idle in all these rounds. Hence, if the transcript sent by the cheating guide/prover agrees with the actual transcript on at least ℓ rounds, then this transcript contains the *actual* execution of one of the copies. Hence, one of the $\binom{2\ell}{\ell}$ solutions provided by these copies was produced by a correct emulation of the original protocol.

[18] In the case of oracle-aided systems, we also want to enable a selective reading of oracle answers.

Generating a solution from a list of possible solutions (for ISs, trivial for IPs). In the case of interactive *proof* systems, we may have the verifier accept if and only if all copies of the emulations accept. Now, assuming the transcript sent by the (cheating) prover agrees with at least ℓ rounds guarantees a correct emulation of a *single* copy (out of $\binom{2\ell}{\ell}$), by applying a union bound over all choices of ℓ out of 2ℓ rounds (and assuming the original protocol had soundness error $2^{-2\ell}$ or so), we get soundness. In the case of interactive *search* systems, however, we need a mechanism that transforms a list of candidate solutions to a single solution such that if the original list contained *one* valid solution, then the resulting (single) solution will be valid (with high probability). Moreover, this mechanism must be *guide-oblivious* in order to fit the SMR setting. Such a mechanism depends on the search problem we deal with (and indeed, for IPs, the mechanism is merely taking the AND of all the bit solutions). The mechanism we use for that purpose is a special kind of interactive search system called solution-merging system, defined in Section 3.3.2.

The transformations. The first (more basic) transformation works along the lines described above: it emulates $\binom{2\ell}{\ell}$ copies of the original protocol, and merges the resulting solutions into one by applying an adequate solution-merging mechanism (see Definition 3.6). Note that this approach can only be applied to interactions with $O(\log n)$-rounds, because it causes an exponential blowup in the message complexity (among the other parameters).[19]

The second transformation provides a way to deal with this exponential blowup, provided that each round of the given protocol can be seen as an independent interactive search system that has a solution-merging mechanism.[20]

3.3.2 Solution-merging systems

A solution-merging system for a D-search problem (Q, R) is an interactive search system for a D-search problem (related to (Q, R), see Remark 3.7) that, given a sequence of input-output pairs $(x_1, y_1), \ldots, (x_t, y_t) \in \{0,1\}^* \times \{0,1\}^*$ (where we think of y_i as a suggested solution for x_i), the system's task is to "merge" the given inputs to a single pair $(x_{\mathsf{out}}, y_{\mathsf{out}})$ that satisfies the following conditions. If the given pairs are all qualified, namely, $(x_i, y_i) \in Q$ for each $i \in [t]$, then the system outputs another qualified pair $(x_{\mathsf{out}}, y_{\mathsf{out}}) \in Q$. On the other hand, if the system is given a sequence that contains a robust pair (that is, there exists $i \in [t]$ such that $(x_i, y_i) \in R$), then the system either rejects or outputs a robust pair $(x_{\mathsf{out}}, y_{\mathsf{out}}) \in R$.

On top of the above, we add the following natural requirement (from a solution-merging system) in case all the given pairs $(x_1, y_1), \ldots, (x_t, y_t)$ correspond to the same input x (i.e., $x := x_1 = \cdots = x_t$). We require the merging system to *always* output a new suggested solution for x, rather than for an arbitrary x_{out} (i.e., we require $x_{\mathsf{out}} = x$).[21] We will use this last requirement when describing the basic

[19] For transforming an IS into a *doubly-efficient* SMR system (as done in this work), in which the searcher's running time is limited to be almost-linear, this approach is even further limited: it can only be applied to systems with $\frac{1}{2}\log_2 n + o(\log n)$ rounds, where the exact bound depends on the precise definition of "almost-linear", which varies across different works.

[20] In fact, it is only required that every $O(\log n)$ can be packed into a mergeable IS.

[21] Defining solution-merging systems can be further generalized for other requirements on x_{out} one wishes to impose. This can be formalized by including a function (or relation) $f : \cup_{t\in\mathbb{N}}(\{0,1\}^*)^t \to \{0,1\}^*$, and requiring $x_{\mathsf{out}} = f(x_1, \ldots, x_t)$. Note that some restrictions need to be set on f, for example, that the searcher can verify that indeed $x_{\mathsf{out}} = f(x_1, \ldots, x_t)$.

IS$\mapsto$SMR-IS transformation in Section 3.3.3.

Definition 3.6 (solution-merging system)**.** *An interactive search system* $\Pi_{\mathsf{merge}} = (\mathcal{G}_{\mathsf{merge}}, \mathcal{S}_{\mathsf{merge}})$ *is a* $(1-\delta)$*-complete and* $(1-\varepsilon)$*-sound* solution-merging system *for a D-search problem* (Q, R) *if the following hold.*

- ***Completeness:*** *For any sequence of qualified input-output pairs* $(x_1, y_1), \ldots, (x_t, y_t) \in Q$ *given as input for* Π_{merge}*, the probability that the searcher* $\mathcal{S}_{\mathsf{merge}}$ *outputs a qualified pair* $(x_{\mathsf{out}}, y_{\mathsf{out}}) \in Q$ *after interacting with the guide* $\mathcal{G}_{\mathsf{merge}}$ *is at least* $1-\delta$*. In addition, if* $x := x_1 = \cdots = x_t$ *then* $x_{\mathsf{out}} = x$ *(with probability 1), and therefore,* $y_{\mathsf{out}} \in Q(x)$ *with probability at least* $1-\delta$*.*
- ***Soundness:*** *For any sequence of pairs* $(x_1, y_1), \ldots, (x_t, y_t)$ *such that* there exists *an index* $i \in [t]$ *for which either* $(x_i, y_i) \in R$ *or* $y_i = \perp$*, and any (potentially malicious and computationally unbounded) guide strategy* $\widetilde{\mathcal{G}}_{\mathsf{merge}}$*, the probability that the searcher* $\mathcal{S}_{\mathsf{merge}}$ *does not reject and outputs a pair* $(x_{\mathsf{out}}, y_{\mathsf{out}})$ *outside of* R *when interacting with* $\widetilde{\mathcal{G}}_{\mathsf{merge}}$ *is at most* ε*. In addition, if* $x := x_1 = \cdots = x_t$ *then either the searcher* $\mathcal{S}_{\mathsf{cmp}}$ *rejects or* $x_{\mathsf{out}} = x$ *(with probability 1).*

Remark 3.7. *Ignoring the additional requirement (regarding* $x_1 = \ldots = x_t$*), a solution-merging system for a D-search problem* (Q, R) *is an interactive search system for the D-search problem* $(Q, R)^{\mathsf{merge}}$ *where*

$$Q^{\mathsf{merge}} = \left(\bigcup_{t\in\mathbb{N}} Q^t\right) \times Q \quad \text{and} \quad R^{\mathsf{merge}} = \left(\bigcup_{t\in\mathbb{N}} \bigcup_{i\in[t]} (\{0,1\}^* \times \{0,1\}^*)^{i-1} \times R \times (\{0,1\}^* \times \{0,1\}^*)^{t-i}\right) \times R. \tag{1}$$

Hence, solution-merging systems are a special case of interactive search systems.

Definition 3.8 (solution-merging problem/mechanism)**.** *A D-search problem* $(Q, R)^{\mathsf{merge}}$ *is a* solution-merging problem *if there exists a D-search problem* (Q, R) *satisfying (1). In this case, we say that* $(Q, R)^{\mathsf{merge}}$ *is a* merging-mechanism for (Q, R)*.*

3.3.3 The basic IS$\mapsto$SMR-IS transformation

We turn to present the basic transformation from IS to SMR-IS (sketched in Section 3.3.1), using the terminology of interactive search systems (recalling that any decision problem is a mergeable search problem).

Construction 3.9 (basic IS$\mapsto$SMR-IS transformation)**.** *Let* $\ell = O(\log n)$ *and let* $\Pi = (\mathcal{G}, \mathcal{S})$ *be an* ℓ*-round interactive search system for a D-search problem* (Q, R)*, and let* $\Pi_{\mathsf{merge}} = (\mathcal{G}_{\mathsf{merge}}, \mathcal{S}_{\mathsf{merge}})$ *be a solution-merging system for* (Q, R)*, where both systems* Π *and* Π_{merge} *are guide-oblivious. For an input* $x \in \{0,1\}^*$*, the SMR system* $\Sigma_{\mathsf{SMR}} := (\mathcal{G}_{\mathsf{SMR}}, \mathcal{S}_{\mathsf{SMR}})$ *is instantiated with the guide* $\mathcal{G}_{\mathsf{SMR}}$ *receiving the input* x*, and the searcher* $\mathcal{S}_{\mathsf{SMR}}$ *either receives the input* x *or its length* $|x|$*. The guide* $\mathcal{G}_{\mathsf{SMR}}$ *and searcher* $\mathcal{S}_{\mathsf{SMR}}$ *proceed as follows.*

1. ***Communication phase:***

(a) Emulating Π: *The guide* $\mathcal{G}_{\mathsf{SMR}}$ *initiates* $\binom{2\ell}{\ell}$ *emulations of* Π *for* x*, indexed by the sets in* $\binom{[2\ell]}{\ell}$*, where the copy* Π^I *will be advanced on rounds in* $I \in \binom{[2\ell]}{\ell}$ *and be idle on the rest. That is, for rounds* $i = 1, \ldots, 2\ell$*:*[22]

- *For each set* $I \in \binom{[2\ell]}{\ell}$ *containing* i*, the searcher* $\mathcal{S}_{\mathsf{SMR}}$ *computes the message* σ_i^I *that* $\mathcal{S}_{\mathsf{out}}$ *would sent in the* i^{th} *round of copy* I*, and sends* $(\sigma_i^I)_{I \ni i}$ *to the guide* $\mathcal{G}_{\mathsf{SMR}}$*.*
- *The guide* $\mathcal{G}_{\mathsf{SMR}}$ *advances all copies of the system* Π^I *for which* $i \in I$*, by computing* $\mathcal{G}$*'s responds* γ_i^I *to the message* σ_i^I*, and sends* $(\gamma_i^I)_{I \ni i}$ *to the searcher. Let* $\widetilde{\gamma}_i$ *denote the actual message the guide* $\mathcal{G}_{\mathsf{SMR}}$ *sent at round* i *(where allegedly* $\widetilde{\gamma}_i = (\gamma_i^I)_{I \ni i}$*.)*

(b) Transcript round: *The guide* $\mathcal{G}_{\mathsf{SMR}}$ *sends a sequence of all the messages she sent in the forgoing* 2ℓ *emulation rounds. Let* $(\widetilde{\mathsf{tr}}_1, \ldots, \widetilde{\mathsf{tr}}_{2\ell})$ *denote the message the guide actually sent (where allegedly,* $\widetilde{\mathsf{tr}}_i = \widetilde{\gamma}_i$ *for each* $i \in [2\ell]$*).*

(c) Solution-merging: *The guide* $\mathcal{G}_{\mathsf{SMR}}$ *computes the solution sequence* $(y^I)_I$ *of all* Π*'s copies, namely,* y^I *is* $\mathcal{S}$*'s output for the transcript* $(\sigma_i^I, \gamma_i^I)_{i \in I}$*. The parties then emulate the solution-merging system* Π_{merge} *on the sequence* $(x, y^I)_{I \in \binom{[2\ell]}{\ell}}$ *(recall that* x *is the input given to the system* Σ_{SMR}*).*

2. ***Reading phase:*** *The searcher* $\mathcal{S}_{\mathsf{SMR}}$ always *reads the transcript message* $(\widetilde{\mathsf{tr}}_1, \ldots, \widetilde{\mathsf{tr}}_{2\ell})$ *and the entire emulation of the solution-merging system* Π_{merge}*. In addition, the searcher samples a uniformly random index* $i^* \in [2\ell]$ *and reads the message* $\widetilde{\gamma}_{i^*}$ *that the guide* $\mathcal{G}_{\mathsf{SMR}}$ *sent in the* i^{*th} *round. The input* x *is revealed to the searcher.*

3. ***Decision phase:*** *The searcher* $\mathcal{S}_{\mathsf{SMR}}$ *performs the following checks:*

   (a) Transcript check: *The searcher checks that the transcript message* $(\widetilde{\mathsf{tr}}_1, \ldots, \widetilde{\mathsf{tr}}_{2\ell})$ *is a sequence of* 2ℓ *messages, such that* $\widetilde{\mathsf{tr}}_i$ *contains a guide* $\mathcal{G}$ *response for each* $I \ni i$*. Namely,* $\widetilde{\mathsf{tr}}_i = (\widetilde{\gamma}_i^I)_{I \ni i}$ *for some strings* $\widetilde{\gamma}_i^I$*, (where allegedly* $\widetilde{\gamma}_i^I = \gamma_i^I$ *for all* $I \in \binom{[2\ell]}{\ell}$ *and* $i \in I$*).*

   (b) Consistency check: *The searcher checks that the message* $\widetilde{\gamma}_{i^*}$ *he read is consistent with the transcript round (namely, that* $\widetilde{\mathsf{tr}}_{i^*} = \widetilde{\gamma}_{i^*}$*).*

   (c) Solution-merging check: *The searcher computes the solution sequence* $(\widetilde{y}^I)_I$*, according to his random coins, his messages and the guide messages indicated by the transcript round. Namely,* $\widetilde{y}^I := \mathcal{S}((\sigma_i^I, \widetilde{\gamma}_i^I)_{I \ni i})$ *for each* $I \in \binom{[2\ell]}{\ell}$*. The searcher checks that the merging system* Π_{merge} *was indeed emulated on the sequence* $(x, \widetilde{y}^I)_I$*.*

   The searcher $\mathcal{S}_{\mathsf{SMR}}$ *rejects if either of the above checks fails. Otherwise, the searcher computes the output of the merging searcher* $\mathcal{S}_{\mathsf{merge}}$ *on the transcript generated during the solution-merging step, and outputs whatever* $\mathcal{S}_{\mathsf{merge}}$ *does.*

Proposition 3.10 (properties of Construction 3.9)**.** *Assume that* Π *(resp.* Π_{merge}*) is a* $(1-\delta)$*-complete (resp.* $(1-\delta_{\mathsf{merge}})$*-complete) and* $(1-\varepsilon)$*-sound (resp.* $(1-\varepsilon_{\mathsf{merge}})$*-sound) interactive search (resp. solution-merging) system for the D-search problem* (Q, R)*, and let* ℓ *(resp.* ℓ_{merge}*) denote the round complexity of*

[22] The choice of the constant 2 in the number of emulation rounds 2ℓ is made only for concreteness. More generally, it may be replaced by any fixed constant $c > 1$. The construction and its analysis remain unchanged, with all quantitative parameters (such as the number of emulated copies, completeness and soundness guarantees, etc.) adjusted according to the chosen value of c.

Π (*resp.* Π_{merge}). *Then Construction 3.9 is a* $(1-\binom{2\ell}{\ell}\cdot\delta-\delta_{\mathsf{merge}})$*-complete and* $\max\{1-\binom{2\ell}{\ell}\cdot\varepsilon-\varepsilon_{\mathsf{merge}}, \frac{1}{2}\}$*-sound SMR-IS system for* (Q,R) *with reading complexity* $2+\ell_{\mathsf{merge}}$ *and* $2\ell+1+\ell_{\mathsf{merge}}$ *rounds.*

Proof. For completeness, fix a qualified input $x \in \mathrm{Dom}(Q)$, and let $E^{\neg\mathsf{comp}}$ denote the (bad) event that there exists a solution in the sequence $(y^I)_I$ that is *not* in $Q(x)$ (and note that in the completeness case, the sequences $(y_i^I)_I$ and $(\widetilde{y}_i^I)_I$ computed by the guide and searcher, respectively, are identical). By the completeness of Π and a union bound, the probability of $E^{\neg\mathsf{comp}}$ (over $\mathcal{S}_{\mathsf{SMR}}$'s coins) is at most $\binom{2\ell}{\ell}\cdot\delta$. Furthermore, since Π_{merge} guarantees completeness whenever all its given input-output pairs are qualified (i.e., when $y^I \in Q(x)$ for each $I \in \binom{[2\ell]}{\ell}$) and the searcher $\mathcal{S}_{\mathsf{SMR}}$ outputs whatever Π_{merge} does, the probability that the searcher fails in this case is at most δ_{merge}. Hence, the probability of the searcher failing is at most $\binom{2\ell}{\ell}\cdot\delta+\delta_{\mathsf{merge}}$.

For soundness, let a robust input $x \in \mathrm{Dom}(R)$, and fix some malicious guide strategy $\widetilde{\mathcal{G}}_{\mathsf{SMR}}$. We assume without loss of generality that $\widetilde{\mathcal{G}}_{\mathsf{SMR}}$ always sends a transcript that passes the Transcript Check 3(a) and performs the solution-merging properly (as otherwise, the searcher immediately rejects). We split the analysis into two cases: if the sent transcript $(\widetilde{\mathsf{tr}}_1,\ldots,\widetilde{\mathsf{tr}}_{2\ell})$ disagrees with more than half the messages that the guide actually sent (that is, $\widetilde{\mathsf{tr}}_i \neq \widetilde{\gamma}_i$ for at least ℓ of the $i \in [2\ell]$), the searcher rejects with probability at least $\frac{1}{2}$. Otherwise, there exists a set $I^* \in \binom{[2\ell]}{\ell}$ such that $\widetilde{\mathsf{tr}}_i = \widetilde{\gamma}_i$ for all $i \in I^*$ (we call such sets $\mathsf{consistent}$). Observe that the solution $\widetilde{y}^{I^*}$ corresponding to a consistent set I^* (and computed by the searcher according to the sent transcript) is in $R(x)\cup\{\perp\}$ with probability at least $(1-\varepsilon)$ (over the corresponding searcher's coins). Indeed, the consistency of I^* guarantees that the guide messages of the corresponding copy Π^{I^*} were generated in "real-time" (namely, on rounds $i \in I^*$, in which Π^{I^*} should be advanced) and appear in the sent transcript $(\widetilde{\mathsf{tr}}_1,\ldots,\widetilde{\mathsf{tr}}_{2\ell})$. Thus, the corresponding solution $\widetilde{y}^{I^*}$ was generated by a true (real-time) emulation of Π. Then, by union bounding over all sets $I \in \binom{[2\ell]}{\ell}$, the probability that *none* of the solutions $(\widetilde{y}^I)_I$ is in $R(x)\cup\{\perp\}$ (given the existence of a consistent set) is at most $\binom{2\ell}{\ell}\cdot\varepsilon$. Otherwise, the existence of a robust solution $\widetilde{y}^I \in R(x)\cup\{\perp\}$ (not necessarily corresponding to a consistent set) and the soundness guarantee of the merging system Π_{merge} (along with the searcher's verification that the merging system Π_{merge} was invoked on the correct input $(x,\widetilde{y}^I)_I$), together guarantee $(1-\varepsilon_{\mathsf{merge}})$-soundness. The soundness bound follows by combining the bounds of the two cases. □

3.3.4 More efficient IS↦SMR-IS transformation

While Construction 3.9 cannot be used to transform interactive search systems with $\omega(\log n)$ rounds into SMR-IS, an alternative exists if the interactive search system Π in question has the following properties (and an arbitrary round complexity). First, we require the system Π to be pure, which, loosely speaking, means that Π proceeds solely by solving a sequence of D-search problems $(Q,R)_1,\ldots,(Q,R)_\ell$ such that the solution to the j^{th} problem $(Q,R)_j$ serves as the input to the next problem $(Q,R)_{j+1}$, and the system outputs the solution to the final problem $(Q,R)_\ell$. No further actions are taken in Π (see subsequent Definition 3.11 for a formal definition).[23] Second, we require each of the search

[23] Strictly speaking, any IS can be made pure for any $\ell \in \mathbb{N}$, by partitioning the protocol into ℓ segments, encapsulating each part as a sub-problem, and transmitting the relevant parts of the transcript between sub-problems. The meaningful requirement is that Π be pure in such a way that each sub-problem admits a solution-merging mechanism, as required next.

problems $(Q,R)_1,\ldots,(Q,R)_\ell$ (that the main system Π uses as subroutines) to have a solution-merging mechanism. Having both the above properties, an interactive search system Π can be transformed into an SMR-IS.

Another important aspect of the efficient IS$\mapsto$SMR-IS transformation (besides enabling to transform certain systems with $\omega(\log n)$ rounds into SMR-ISs), is that the resulting SMR-IS has an arbitrary small but noticeable soundness error (i.e., soundness error $1/\mathrm{poly}(n)$) rather than a constant soundness error, achieved by the basic IS$\mapsto$SMR-IS transformation (Construction 3.9). Therefore, the efficient transformation is also beneficial for pure systems with $O(\log n)$ rounds, when one wants to achieve a sub-constant soundness error (as well as improving the guide's running time). See Proposition 3.13.

For the rest of this section, we let Π be a pure oracle-aided interactive search system for (Q,R), proceeding by sequentially using oracle access for the D-search problems $(Q,R)_1,\ldots,(Q,R)_\ell$, to whom we refer as the advancing oracles/problems (for emphasizing when we use the oracles $(Q,R)_1,\ldots,(Q,R)_\ell$, which advance the computation of Π, rather than using merging oracles, which leave the computation of Π at the same level). Specifically, note that (Q,R) is *guide-oblivious*. Let $(Q,R)_1^{\mathsf{merge}},\ldots,(Q,R)_\ell^{\mathsf{merge}}$ denote some merging mechanisms for the advancing D-search problems $(Q,R)_1,\ldots,(Q,R)_\ell$ respectively. In addition, fix some integer $\ell_{\mathsf{SMR}}=\mathrm{poly}(\ell)$, which corresponds to the round complexity of the resulting oracle-aided SMR-IS.

Construction overview. The efficient IS$\mapsto$SMR-IS transformation proceeds through ℓ_{SMR} epochs, maintaining the following invariant: at each epoch, and each level $j\in[\ell]$, the system maintains a collection of *multiple* j-level subcomputations of Π, which are partial executions of Π that have been advanced through the first j oracle-calling steps of Π. The key step at each epoch is that the copies of the different subcomputations are not advanced individually, which prevents the exponential blowup that would otherwise accrue at the basic transformation (Construction 3.9). Instead, for each level $j\in[\ell]$, the parties first *merge* all currently available j-level subcomputations by invoking the corresponding merging mechanism $(Q,R)_j^{\mathsf{merge}}$, and then submit the merged answer as the single query to the next oracle $(Q,R)_{j+1}$, producing one new $(j+1)$-level subcomputation. Consequently, all different j-level subcomputations are advanced at the "cost" of one, and the system gradually advances the computation of Π while introducing only a *single* advancing subcomputation per level. Crucially, the different j-level subcomputations are not discarded: each subcomputation remains in the j-level collection and will be merged again in later epochs, alongside any new subcomputations that subsequently reach level j.

More concretely, at epoch $i=1,\ldots,\ell_{\mathsf{SMR}}$ and each $j\in[\min(i-1,\ell-1)]$, the guide $\mathcal{G}_{\mathsf{SMR}}$ collects all answers obtained so far from the j^{th} oracle $(Q,R)_j$ (which is the current collection of j-level subcomputation) and submit them to $(Q,R)_j^{\mathsf{merge}}$.[24] The answer m_j returned by the merging system $(Q,R)_j^{\mathsf{merge}}$ is then submitted as the query to $(Q,R)_{j+1}$. Thus, all subcomputations at the same level are first compressed into a single merged subcomputation, and only this merged subcomputation is advanced. In addition, the guide instantiates a new 1-level subcomputation of Π by feeding the input of the system as a query to the first oracle $(Q,R)_1$. At the start of the next epoch, the collection

[24] Formally, solution-merging systems receive input-output pairs rather than outputs alone (see Definition 3.6), which in the oracle setting correspond to query-answer pairs. Recall, however, that we assume without loss of generality that oracle answers include the corresponding query, so whenever the full pair is required, the answer should be read as the corresponding query-answer pair.

at level j consists of the previous j-level subcomputations together with the one newly produced by advancing the merge of the $(j-1)$-level subcomputations. See illustration in Figure 1.

After ℓ_{SMR} epochs, the guide sends the searcher the full transcript accumulated throughout. The parties then invoke the final merging mechanism $(Q,R)_\ell^{\mathsf{merge}}$ on all ℓ-level subcomputations (that is, on all query-answer pairs obtained from $(Q,R)_\ell$) and use the resulting merged answer as the output of the transformed system.

The searcher then performs his reading and decision phases. In the current setting, the full transcript (sent in the last round) consists of the queries the guide allegedly submitted to the various oracles, along with the alleged corresponding oracle answers. The searcher reads the transcript and verifies that it corresponds to a valid emulation of the system (as specified in Construction 3.12). In addition, the searcher samples a uniformly random epoch, reads the *actual* queries submitted by the guide during this epoch together with the actual oracle answers received, and rejects if he detects any inconsistency with the transmitted transcript.

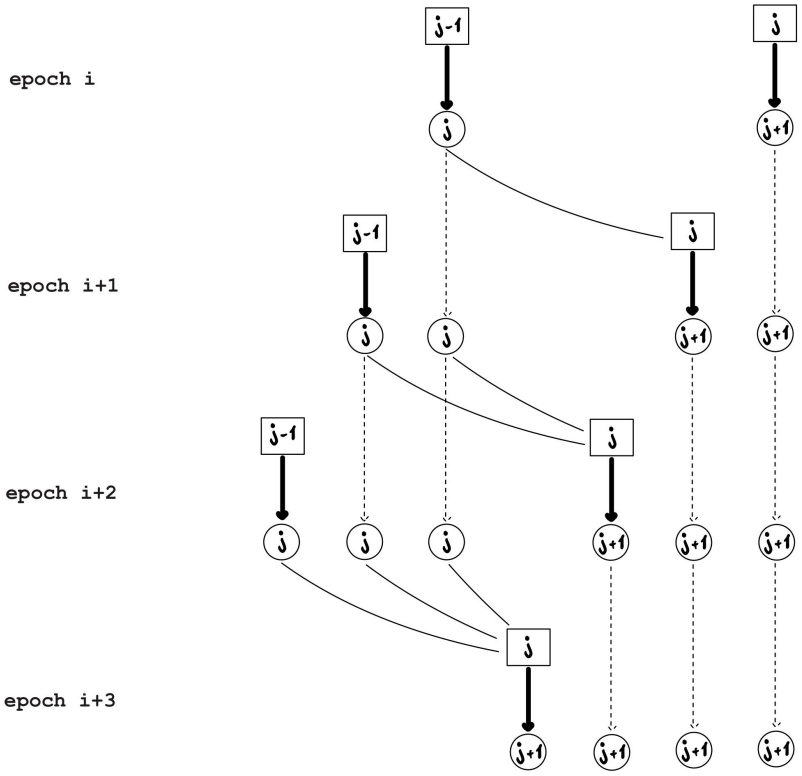

Figure 1: A snapshot of the SMR system resulting by the efficient IS$\mapsto$SMR-IS transformation. Each horizontal row represents an epoch of Π_{SMR}, and each vertical column represents a subcomputation of the original system Π. At each epoch, the system Π_{SMR} first *merges* all subcomputations at the same level (the subcomputations are represented by circles, and the boxes represent the merging). Then, the system *advances* only the resulting merged subcomputation to the next level (represented by the thick arrow), while retaining the original subcomputations for future epochs (represented by the dashed arrows). Repeating this process across epochs gradually advances the computation while maintaining only a single advancing subcomputation per level.

Analysis overview. The soundness analysis proceeds similarly to the analysis of the basic IS $\mapsto$ SMR-IS transformation (Construction 3.9). If the transcript disagrees with more than $\ell_{\mathsf{SMR}} - \ell$ epochs, then the searcher rejects with probability at least $1 - \frac{\ell}{\ell_{\mathsf{SMR}}}$. Otherwise, there are at least ℓ epochs for which the reported queries and answers in the transcript were actually generated in real time. We refer to such epochs as consistent. By an inductive argument, assume that during the j^{th} consistent epoch, the j^{th} advancing oracle $(Q,R)_j$ receives a robust query q_j^* and therefore returns a robust answer a_j^*, (i.e., $(\mathsf{q}_j^*, \mathsf{a}_j^*) \in R_j$). By consistency of the epoch, this robust query-answer pair $(\mathsf{q}_j^*, \mathsf{a}_j^*)$ appears in the transcript sent by the guide. Now, consider the subsequent $j+1^{\text{st}}$ consistent epoch. Since the epoch is consistent and the transcript passed the searcher's verification, the searcher can verify that this robust pair $(\mathsf{q}_j^*, \mathsf{a}_j^*)$ was indeed submitted to the j^{th} merging oracle $(Q,R)_j^{\mathsf{merge}}$ as part of its query. Receiving this (possibly single) robust pair guarantees that $(Q,R)_j^{\mathsf{merge}}$ outputs another robust query-answer pair $(\mathsf{q}_j^{\mathsf{merge}*}, \mathsf{a}_j^{\mathsf{merge}*}) \in R_j$. Again by consistency of the epoch, the $j+1^{\text{st}}$ advancing oracle $(Q,R)_{j+1}$ indeed receives $\mathsf{a}_j^{\mathsf{merge}*}$ as its query and therefore returns a robust answer a_{j+1}^* satisfying $(\mathsf{a}_j^{\mathsf{merge}*}, \mathsf{a}_{j+1}^*) \in R_{j+1}$. Soundness then follows by taking $j = \ell - 1$ and having the searcher check that the guide called the final merging oracle $(Q,R)_\ell^{\mathsf{merge}}$ as required.

We now turn to formalizing the above-sketched construction and its analysis.

Definition 3.11 (pure IS)**.** *An oracle-aided interactive search system Π is* pure*, if there exists a sequence of D-search problems $(Q,R)_1, \ldots, (Q,R)_\ell$ for some integer $\ell := \ell(n)$[25] such that computing the system Π on a given input $x \in \{0,1\}^n$ is carried as follows. First, the parties invoke the first oracle $(Q,R)_1$ on the input x. Then, for $j = 1, \ldots, \ell - 1$, the parties submit the answer a_j received from the j^{th} oracle $(Q,R)_j$ as a query for the $j+1^{st}$ oracle $(Q,R)_{j+1}$. Finally, computing $\Pi(x)$ ends with the searcher outputting the answer received by the last oracle $(Q,R)_\ell$.*

Construction 3.12 (more efficient IS $\mapsto$ SMR-IS transformation)**.** *Let Π be a* pure *oracle-aided interactive search system for a D-search problem (Q,R), carried by sequentially using oracle access to the D-search problems $(Q,R)_1, \ldots, (Q,R)_\ell$. In addition, for each $j \in [\ell]$, let $(Q,R)_j^{\mathsf{merge}}$ be a merging system for $(Q,R)_j$ and define the "zeroth" D-search problems $(Q,R)_0 = (Q,R)_0^{\mathsf{merge}} := \mathsf{Id}$ to be the identity function (both in the qualified and robust cases). On an input $x \in \{0,1\}^n$, letting $\ell_{\mathsf{SMR}} \in [\ell, \mathrm{poly}(\ell)]$, the SMR-IS system $\Sigma_{\mathsf{SMR}} := (\mathcal{G}_{\mathsf{SMR}}, \mathcal{S}_{\mathsf{SMR}})$ for (Q,R) proceed as follows.*

1. ***Communication phase:***

   (a) Emulating Π: *The guide initiates ℓ empty answer sets $\mathsf{A}_1, \ldots, \mathsf{A}_\ell$, and an additional "zeroth" set $\mathsf{A}_0 := \{(x,x)\}$. For epoch $i = 1, \ldots, \ell_{\mathsf{SMR}}$:*

   i. Answer merging: *The parties invoke the merging oracles $(Q,R)_0^{\mathsf{merge}}, \ldots, (Q,R)_{\min(i-1,\ell-1)}^{\mathsf{merge}}$*[26] *in parallel: the guide submits all answers in A_j as the query to the j^{th} merging oracle $(Q,R)_j^{\mathsf{merge}}$, and the searcher submits the expected query length. Let m_j^i denote the answer received from $(Q,R)_j^{\mathsf{merge}}$ at epoch i.*

[25] Although the sequence length ℓ may depend on n (i.e., the input length), note that the number of *different* D-search problems in the sequence $(Q,R)_1, \ldots, (Q,R)_\ell$ must be *constant*. This is since there must be a uniform description (i.e., a Turing machine) that describes the operation of the pure system Π.

[26] Note that $\mathsf{A}_j \neq \emptyset$ at the beginning of epoch i if and only if $j < i$, hence $(Q,R)_j^{\mathsf{merge}}$ is invoked only when $j < i$.

ii. Advancing the computation: *The parties invoke the advancing oracles* $(Q,R)_1,\ldots,(Q,R)_{\min(i,\ell)}$[27] *in parallel: for each* j *the guide submits the merged answer* m^i_{j-1} *as a query for the* j^{th} *advancing oracle* $(Q,R)_j$*, and the searcher submits the expected query length. The guide adds the answer* a^i_j *received from* $(Q,R)_j$ *to the set* A_j *(where the answer* a^i_j *includes the corresponding query* m^i_{j-1}*). Namely, the guide updates* $\mathsf{A}_j = \mathsf{A}_j \cup \{\mathsf{a}^i_j\}$*.*
Let tr^i *denote the transcript obtained during the* i^{th} *epoch. Namely,* tr^i *contains all the queries the guide actually submitted to the oracles during the* i^{th} *epoch, along with the received oracle answers. Hence, allegedly,*

$$\mathsf{tr}^i = \left((\mathsf{A}_0,\mathsf{m}^i_0),\ldots,(\mathsf{A}_{\ell-1},\mathsf{m}^i_{\ell-1})\right) \,||\, \left((\mathsf{m}^i_0,\mathsf{a}^i_1),\ldots,((\mathsf{m}^i_{\ell-1},\mathsf{a}^i_\ell)\right).$$

(*b*) Transcript round: *The guide sends a sequence of all the transcripts obtained during the foregoing* ℓ_{SMR} *emulation epochs (namely, the guide sends all the queries and contexts he submitted during the* ℓ_{SMR} *epochs, along with the received answer). Let* $(\widetilde{\mathsf{tr}}^1,\ldots,\widetilde{\mathsf{tr}}^{\ell_{\mathsf{SMR}}})$ *denote the transcript message the guide actually sent (where allegedly,* $\widetilde{\mathsf{tr}}^i = \mathsf{tr}^i$*, for each* $i \in [\ell_{\mathsf{SMR}}]$*).*

(*c*) Solution-merging round: *The guide submits the answers in the set* A_ℓ *as a query to the merging oracle* $(Q,R)^{\mathsf{merge}}_\ell$ *and the searcher submits the expected query length. Let* $\mathsf{m}^{\mathsf{fin}}_\ell$ *denote the oracle's answer.*

2. ***Reading phase:*** *The searcher reads the transcript message* $(\widetilde{\mathsf{tr}}^1,\ldots,\widetilde{\mathsf{tr}}^{\ell_{\mathsf{SMR}}})$ *and the (final) solution-merging round (namely, the searcher reads the query and context the guide actually submitted to the oracle* $(Q,R)^{\mathsf{merge}}_\ell$ *and the received answer). In addition, the searcher samples a uniformly random* $i^* \in [\ell_{\mathsf{SMR}}]$ *and reads the transcript* tr^{i^*} *actually obtained during epoch* i^**.*

3. ***Decision phase:*** *The searcher performs the following checks:*

(*a*) Transcript check: *The searcher verifies that the transcript message* $(\widetilde{\mathsf{tr}}^1,\ldots,\widetilde{\mathsf{tr}}^{\ell_{\mathsf{SMR}}})$ *corresponds to a valid emulation of Step 1(a) of the system. To this end, the searcher initializes new sets* $\mathsf{A}^{\mathsf{sim}}_0 := \{(x,x)\}$ *and* $\mathsf{A}^{\mathsf{sim}}_j := \emptyset$ *for every* $j \in [\ell]$*, where* $\mathsf{A}^{\mathsf{sim}}_j$ *serves to simulate* A_j*. The searcher then simulates each epoch* $i = 1,\ldots,\ell_{\mathsf{SMR}}$ *of Step 1(a) as follows.*
In the answer-merging step, for every $j = 0,\ldots,\min(i-1,\ell-1)$*, the searcher compares the query to* $(Q,R)^{\mathsf{merge}}_j$*, as reported in the transcript message, with the correct query according to the simulation, in which he submits all the query-answer pairs in* $\mathsf{A}^{\mathsf{sim}}_j$*. The searcher rejects if he sees any discrepancy. Otherwise, the searcher takes the corresponding answer reported in* $\widetilde{\mathsf{tr}}^i$ *as the answer of* $(Q,R)^{\mathsf{merge}}_j$*. Let* $\widetilde{\mathsf{m}}^i_j$ *denote this answer.*
In the advancing step, for every $j = 1,\ldots,\min(i,\ell)$*, the searcher checks that the reported query to* $(Q,R)_j$ *is exactly* $\widetilde{\mathsf{m}}^i_{j-1}$*, rejecting if he sees any discrepancy, and takes the corresponding answer reported in* $\widetilde{\mathsf{tr}}^i$ *as the answer to* $(Q,R)_j$*. Let* $\widetilde{\mathsf{a}}^i_j$ *denote this answer. The searcher then updates* $\mathsf{A}^{\mathsf{sim}}_j = \mathsf{A}^{\mathsf{sim}}_j \cup \{\widetilde{\mathsf{a}}^i_j\}$*, mirroring the guide's update of* A_j*.*

(*b*) Consistency check: *The searcher checks that the transcript* tr^{i^*} *he read from (the actual emulation of) epoch* i^* *is consistent with the sent transcript message (namely, that* $\widetilde{\mathsf{tr}}^{i^*} = \mathsf{tr}^{i^*}$*).*

[27] In similar to Footnote 26, $(Q,R)_i$ can only be invoked when $j \le i$.

(c) Solution-merging check: *The searcher checks that the solution-merging round (i.e., Step 1(c)) was carried correctly by verifying that the guide indeed submitted the answers in* A_ℓ^{sim} *(as resulting at the end of the searcher's emulation) as a query to the merging oracle* $(Q,R)_\ell^{\mathsf{merge}}$.

The searcher rejects if either of the above checks fails. Otherwise, the searcher outputs the answer $\mathsf{m}_\ell^{\mathsf{fin}}$ *received from the merging oracle* $(Q,R)_\ell^{\mathsf{merge}}$ *in the solution-merging round.*

Proposition 3.13 (properties of Construction 3.12)**.** *Let* Π *be a pure oracle-aided interactive search system for a D-search problem* (Q,R)*, as in Construction 3.12, and* $\ell_{\mathsf{SMR}} = \mathrm{poly}(\ell)$*. Then, Construction 3.9 is a perfectly complete and* $(1-\frac{\ell}{\ell_{\mathsf{SMR}}})$*-sound oracle-aided SMR-IS system for* (Q,R)*, with reading complexity* $O(1)$ *and* $O(\ell_{\mathsf{SMR}})$ *rounds.*

Proof. The complexity of Construction 3.12 is easily derived from the construction, and perfect completeness follows trivially from the ideality of the oracles.

For soundness, let $x \in \mathrm{Dom}(R)$ be a robust input, and fix an arbitrary malicious guide strategy $\widetilde{\mathcal{G}}_{\mathsf{SMR}}$. As in the proof of Proposition 3.10, we assume without loss of generality that $\widetilde{\mathcal{G}}_{\mathsf{SMR}}$ always passes both the Transcript Check 3(a) and Solution-Merging Check 3(c). Now, if at least $\ell_{\mathsf{SMR}} - \ell$ epochs disagree with the sent transcript (i.e., $|\{i : \widetilde{\mathsf{tr}}^i \neq \mathsf{tr}^i\}| \geq \ell_{\mathsf{SMR}} - \ell$,), then the searcher detects an inconsistency (and rejects) with probability at least $1 - \frac{\ell}{\ell_{\mathsf{SMR}}}$. Otherwise, there exists a sequence of indices $i_1 < \cdots < i_\ell \in [\ell_{\mathsf{SMR}}]$ such that $\widetilde{\mathsf{tr}}^{i_j} = \mathsf{tr}^{i_j}$ for each $j \in [\ell]$. We refer to these epochs as $\mathsf{consistent}$. For the remainder of the proof, let i_0 denote the "0^{th}" epoch corresponding to the system initialization (before the first epoch begins), and recall that the "zero$^{\text{th}}$" robust relation R_0 was defined to be the identity relation.

Claim 3.14. *For every* $j \in \{0,\ldots,\ell\}$*, the searcher's emulation of epoch* i_j *ends with the answer set* $\mathsf{A}_j^{\mathsf{sim}}$ *containing a robust element of* $(Q,R)_j$*, namely,* $\mathsf{A}_j^{\mathsf{sim}} \cap (R_j \cup \{\perp\}) \neq \emptyset$.

Note that Claim 3.14 implies perfect soundness in this case (i.e., when a sequence $i_1 < \cdots < i_\ell$ of consistent indices exists). This follows by using Claim 3.14 for $j = \ell$, the searcher's Solution-Merging Check 3(c) and the functionality of the merging oracle $(Q,R)_\ell^{\mathsf{merge}}$.

Proof of Claim 3.14. The proof is by induction on $j \in \{0,\ldots,\ell\}$. The base case $j = 0$ trivially holds by $(x,x) \in \mathsf{A}_0^{\mathsf{sim}} \cap R_0$. For the induction step, fix some $j \in [\ell - 1]$, and assume that the searcher's simulation of epoch i_j ends with $\mathsf{A}_j^{\mathsf{sim}}$ containing a robust answer $\widetilde{\mathsf{a}}_j^* \in R_j \cup \{\perp\}$ (where formally, $\widetilde{\mathsf{a}}_j^*$ is the corresponding query–answer pair). Consider the subsequent consistent epoch i_{j+1}. Since the searcher never removes elements from answer sets, the simulation of epoch i_{j+1} begins with $\mathsf{A}_j^{\mathsf{sim}}$ still containing the robust answer $\widetilde{\mathsf{a}}_j^*$. By consistency of epoch i_{j+1} and assuming the Transcript Check 3(a) passed, during the actual execution of epoch i_{j+1} the guide indeed submitted the contents of $\mathsf{A}_j^{\mathsf{sim}}$ (and in particular, $\widetilde{\mathsf{a}}_j^*$) as a query to the merging oracle $(Q,R)_j^{\mathsf{merge}}$. By the functionality of $(Q,R)_j^{\mathsf{merge}}$, the returned answer $\widetilde{\mathsf{m}}_j^{i_{j+1}}$ is also robust, namely, $\widetilde{\mathsf{m}}_j^{i_{j+1}} \in R_j \cup \{\perp\}$. Again, using consistency of epoch i_{j+1} together with the transcript check, the guide must have submitted the answer $\widetilde{\mathsf{m}}_j^{i_{j+1}}$ as the query to the advancing oracle $(Q,R)_{j+1}$, as otherwise, the inconsistency would have appeared in the transmitted transcript, leading the searcher to reject. Therefore, by the functionality of $(Q,R)_{j+1}$, the returned answer $\widetilde{\mathsf{a}}_{j+1}^{i_{j+1}}$ is robust. Finally, the consistency of epoch i_{j+1} guarantees that this actual

answer $\widetilde{\mathsf{a}}_{j+1}^{i_{j+1}}$ is indeed recorded in the sent transcript and is therefore inserted into $\mathsf{A}^{\mathsf{sim}}_{j+1}$ at the end of emulating epoch i_{j+1}, completing the induction. □

This completes the proof of Proposition 3.13. □

4 An interactive proof system for bounded-depth circuits

We use the tools and terminology developed in earlier sections to construct an interactive proof system for sets decidable by uniform circuits of bounded size and depth, thereby reestablishing the result of [GKR15]. We thereby present a *modular* approach for constructing an interactive proof system for bounded-circuit classes, which is then leveraged in Section 5, where we use the system's modularity to construct an SMR-IS for the same classes.

The construction is based on the interactive proof system of Goldwasser, Kalai, and Rothblum [GKR15], which originally proved their result for sets decidable by log-space uniform circuits of bounded depth. To simplify the presentation, we use the interactive proof systems of [RRR16] for bounded computations, which also extends [GKR15] to broader classes. For a presentation of the [GKR15] protocol and its prerequisites, we refer the readers to [GKR15] or to Goldreich's alternative exposition [Gol18, Chapter 3].

For integer functions $s, d, \mathsf{T}, \mathsf{S} : \mathbb{N} \to \mathbb{N}$, let $\mathcal{UC}irc(s, d, \mathsf{T}, \mathsf{S})$ denote the class of all sets decided by $\mathsf{T}(n)$-time and $\mathsf{S}(n)$-space uniform Boolean circuit family $\{C_n\}_n$,[28] where C_n receives inputs of length n, has size $s(n)$, and depth $d(n)$. We reestablish the following.

Theorem 4.1 ([GKR15, Theorem 1.1])**.** *Let $\mathsf{T} := \mathsf{T}(n)$, $\mathsf{S} := \mathsf{S}(n)$, $s := s(n)$ and $d := d(n)$ integer functions such that $n \leq \mathsf{T} \leq \exp(n)$ and $\log \mathsf{T} \leq \mathsf{S} \leq \mathrm{poly}(n)$, and let a constant $\mu \in (0, \frac{1}{2})$. Any set in $\mathcal{UC}irc(s, d, \mathsf{T}, \mathsf{S})$ has a perfectly complete and $\frac{2}{3}$-sound[29] interactive proof system with the following properties. The prescribed (honest) prover runs in time $\mathsf{T}^{1+O(\mu)} \cdot \mathrm{poly}(s, \mathsf{S})$. The verifier runs in time $n \cdot \mathrm{poly}(d, \log \mathsf{T}) + (\mathsf{T} + \mathrm{poly}(s))^{O(\mu)} \cdot \mathrm{poly}(\mathsf{S})$. The protocol is public coin and has $O(d \log s)$ rounds.*

Corollary 4.2. *Any set L in $\mathcal{UC}irc(s, d, \mathsf{T}, \mathsf{S})$ with $s, \mathsf{T} = \mathrm{poly}(n)$ and $d, \mathsf{S} = \mathrm{polylog}(n)$ has a doubly-efficient[30] interactive proof system which is perfectly complete and $\frac{2}{3}$-sound. The prescribed (honest) prover runs in time $\mathrm{poly}(n)$. The verifier runs in time $\widetilde{O}(n)$. The protocol is public coin and has $\mathrm{polylog}(n)$ rounds.*

We present a modular approach for constructing the interactive proof system for any set L in $\mathcal{UC}irc(s, d, \mathsf{T}, \mathsf{S})$. The method we use leverages the modularity of interactive search systems and their composition by identifying subtasks that need to be fulfilled to decide any set L and formulating them as independent D-search problems. We then formalize the main module for deciding L as an oracle-aided system that uses oracle access to these identified D-search problems, and relies on future protocols to solve them. Next, we independently consider each of these well-defined D-search problems

[28] Recall that a circuit family $\{C_n\}_n$ is $\mathsf{T}(n)$-time and $\mathsf{S}(n)$-space uniform if there exists a deterministic Turing machine that on input 1^n, outputs a description of C_n in time $\mathsf{T}(n)$ and space $\mathsf{S}(n)$.

[29] We took $\frac{2}{3}$ as an arbitrary constant larger than $\frac{1}{2}$. The soundness can be improved using parallel repetition.

[30] Recall that an interactive search (proof) system is *doubly-efficient* if the guide (prover) runs in time $\mathrm{poly}(n)$ and the searcher (verifier) runs in time $\widetilde{O}(n) = n \cdot \mathrm{polylog}(n)$ (see Definition 2.2).

and present interactive search systems that solve them. Lastly, we use the composition theorem for interactive search systems Theorem 2.8 to compose the main module with the protocols that solve its subtask D-search problems, and derive a (plain) interactive proof system for $\mathcal{UC}irc(s, d, \mathsf{T}, \mathsf{S})$.

This modular approach enables one to consider each subtask (D-search problem) independently and focus on constructing a protocol that solves it and has the best properties one can provide. Apart from viewing each D-search problem as an independent challenge to be solved (regardless of the main module that inspired it), using this modular approach enables *any* interactive search system that needs to solve this D-search problem as a subtask to use any protocol that solves it. The benefits of using this approach are demonstrated in the following section and are further illustrated in Section 5, where we leverage this modularity to construct an SMR-IS system for $\mathcal{UC}irc(s, d, \mathsf{T}, \mathsf{S})$.

4.1 Preliminaries for this section

4.1.1 Low-degree extensions

For a finite field $\mathbb{F}$, a subset $H \subseteq \mathbb{F}$ and integers ℓ and s, consider a function $v : [s] \to \mathbb{F}$ as $v : |H|^\ell \to \mathbb{F}$ where $|H|^\ell \geq s$. A low-degree extension of v is a *polynomial* $\widehat{v} : \mathbb{F}^\ell \to \mathbb{F}$ of individual degree $|H| - 1$ that agrees with v on its entire domain (i.e., $\widehat{v}(\beta) = v(\beta)$ for each $\beta \in H^\ell$). Any function v as above has a *unique* low-degree extension $\widehat{v} : \mathbb{F}^\ell \to \mathbb{F}$, given by

$$\widehat{v}_i(X) = \sum_{\beta \in H^\ell} \mathsf{EQ}(X, \beta) \cdot v_i(\beta), \tag{2}$$

where $\mathsf{EQ} : \mathbb{F}^\ell \times \mathbb{F}^\ell \to \mathbb{F}$ is a polynomial of individual-degree $|H| - 1$ that checks equality over H^ℓ.[31] We refer to the above extension as the low-degree extension of v with respect to $\mathbb{F}, H$ and ℓ.

We note that there exists a Turing machine that evaluates the low-degree extension of v with respect to $\mathbb{F}, H$ and ℓ in time $|H|^{\ell+1}\text{polylog}\,|\mathbb{F}|$ and space $O(\ell \log |H| + \log |\mathbb{F}|)$.

4.1.2 Circuits

Let C_n be a layered Boolean circuit of size s and depth d, and assume for simplicity that each of the layers has exactly s gates and that all C_n's internal gates are NAND.[32] For an input $x \in \{0,1\}^n$, we represent the layers of the computation $C_n(x)$, top to bottom, by functions $v_0^x, \ldots, v_d^x : [s] \to \{0,1\}$, where v_0^x represents the output layer and v_d^x the input layer. Hence, $v_0^x = C_n(x)0^{s-1}$ and $v_d^x = x0^{s-n}$. In addition, for a field $\mathbb{F}$, a subset $H \subseteq \mathbb{F}$ and an integer ℓ such that $s \leq |H|^\ell$, let $\widehat{v}_0^x, \ldots, \widehat{v}_d^x : \mathbb{F}^\ell \to \mathbb{F}$ be the low-degree extensions of $v_0^x, \ldots, v_d^x$, respectively, with respect to $\mathbb{F}$, H and ℓ.

The interactive search system we are about to present in this section (specifically, the system for the next-layer problem $(Q, R)^V_{\text{next-layer}}$ Construction 4.9) uses the following observation, dated back to [BFLS91, FGL+96], which relates adjacent layers v_i^x, v_{i+1}^x of the circuit.

[31] For example, EQ can be written as $\mathsf{EQ}(X_1, \ldots, X_\ell, \beta_1, \ldots, \beta_\ell) = \prod_{j \in [\ell]} \prod_{\beta \in H \setminus \{\beta_j\}} \frac{X_j - \beta}{\beta_j - \beta}$, where $\beta_1, \ldots, \beta_\ell$ are some elements in H.

[32] Throughout this work, we assume that all circuits we consider are layered and use only NAND gates. For any circuit of depth d, these assumptions incur a multiplicative overhead of at most $O(d)$ in the circuit size and at most a constant multiplicative factor in the circuit depth.

Observation 4.3 ([BFLS91, FGL+96]). *For an index $i \in \{0,\ldots,d-1\}$, let v_i^x and v_{i+1}^x be functions that correspond to adjacent layers of $C_n(x)$, and view them as functions from H^ℓ to $\{0,1\}$. Let $\phi_i^n : H^\ell \times H^\ell \times H^\ell \to \{0,1\}$ be a function that describes the relation between these two layers by evaluating to 1 if and only if the gate indexed by its first input at layer i is fed by the two $(i+1)^{st}$-layer gates indexed by its second and third inputs. Then,*

$$v_i^x(X) = \sum_{\beta',\beta'' \in H^\ell} \phi_i^n(X,\beta',\beta'') \cdot (1 - v_{i+1}^x(\beta') \cdot v_{i+1}^x(\beta'')). \tag{3}$$

By combining (3) with (2) and letting $\widehat{\phi}_i^n : \mathbb{F}^\ell \times \mathbb{F}^\ell \times \mathbb{F}^\ell \to \mathbb{F}$ denote the low-degree extension of ϕ_i^n with respect to $\mathbb{F}$, H and ℓ, one gets

$$\widehat{v}_i^x(X) = \sum_{\beta,\beta',\beta'' \in H^\ell} \mathsf{EQ}(X,\beta) \cdot \widehat{\phi}_i^n(\beta,\beta',\beta'') \cdot (1 - \widehat{v}_{i+1}^x(\beta') \cdot \widehat{v}_{i+1}^x(\beta'')). \tag{4}$$

Claim 4.4 ([GKR15, Claim 4.6]). *Let $\{C_n\}_n$ be a $\mathsf{T} := \mathsf{T}(n)$-time and $\mathsf{S} := \mathsf{S}(n)$-space uniform circuit family of size $s := s(n)$. Then, any polynomial in $\Phi := \Phi(\{C_n\}_n)$ can be evaluated by an $O(\mathsf{T} + \mathrm{poly}(s, \log|\mathbb{F}|))$-time and $O(\mathsf{S} + \log|\mathbb{F}|)$-space Turing machine, that given an input $\alpha_{\mathsf{in}} \in (\mathbb{F}^\ell)^3$ and a description $\langle 1^n, i, \mathbb{F}, H, \ell\rangle$ of a polynomial $\widehat{\phi}_i^n$ in Φ, outputs the evaluation $\widehat{\phi}_i^n(\alpha_{\mathsf{in}})$.*

We include the proof of Claim 4.4 for completeness.

Proof. Let an integer $n \in \mathbb{N}$, index $i \in [d]$, and consider the polynomial $\widehat{\phi}_i^n \in \Phi$. Since the circuit C_n is constructable in time T, space S and has size s, the adjacency function ϕ_i^n can be constructed in time $O(\mathsf{T})$ and space $O(\mathsf{S})$, and be evaluated in time $O(s)$ and space $O(\log s) \leq O(\mathsf{S})$. Together with the evaluating algorithm from Section 4.1.1, we conclude that $\widehat{\phi}_i^n$ can be evaluated in time $O(\mathsf{T}) + \mathrm{poly}(s, \log|\mathbb{F}|)$ and space $O(\mathsf{S} + \log|\mathbb{F}|)$. □

4.2 Oracle-aided IP for bounded-depth circuit classes

In this section, we present an oracle-aided interactive search system for $\mathcal{UC}irc(s,d,\mathsf{T},\mathsf{S})$, which serves as the main module of the construction. Although the system is, in fact, a proof system, we adhere to search-system terminology throughout to avoid confusion later, recalling that any proof system is a special case of a search system.

For the rest of this section, fix a set $L \in \mathcal{UC}irc(s,d,\mathsf{T},\mathsf{S})$, and let $\{C_n\}_n$ be a family of Boolean circuits of size $s := s(n)$, depth $d := (n)$ that decides L. In addition, let $V := \{\widehat{v}_i^x\}_{x,i}$ denote the polynomial family that represents the layers of the computation $C_{|x|}(x)$ for any $x \in \{0,1\}^*$ (see Section 4.1.2).

4.2.1 The main search problem that the protocol uses

The heart of the main system for deciding L is captured by a D-search problem named the next-layer problem $(Q,R)_{\mathsf{next\text{-}layer}}$, defined as follows. The next-layer problem $(Q,R)^V_{\mathsf{next\text{-}layer}}$ receives an element $\alpha_{\mathsf{in}} \in \mathbb{F}^\ell$ and a value $\zeta_{\mathsf{in}} \in \mathbb{F}$, along with $\langle x,i\rangle$ which specifies a pair of encodings $(\widehat{v}_i^x, \widehat{v}_{i+1}^x)$ in $V \times V$. The goal is to find an element $\alpha_{\mathsf{out}} \in \mathbb{F}^\ell$ and a value $\zeta_{\mathsf{out}} \in \mathbb{F}$ such that $\widehat{v}_{i+1}^x(\alpha_{\mathsf{out}}) = \zeta_{\mathsf{out}}$ if and only if

$\widehat{v}_i^x(\alpha_{\mathsf{in}}) = \zeta_{\mathsf{in}}$. In other words, the next-layer problem $(Q,R)^V_{\mathsf{next\text{-}layer}}$ reduces a true (resp. false) claim about a value of $\widehat{v}_i^x$ to a true (resp. false) claim about a value of the polynomial $\widehat{v}_{i+1}^x$. We define the qualified (resp. robust) relation $Q^V_{\mathsf{next\text{-}layer}}$ (resp. $R^V_{\mathsf{next\text{-}layer}}$) to consist of all pairs of correct (resp. incorrect) equations. Thus

$$\begin{aligned} Q^V_{\mathsf{next\text{-}layer}} &= \{(((\langle x,i\rangle,\alpha_{\mathsf{in}},\zeta_{\mathsf{in}}),(\alpha_{\mathsf{out}},\zeta_{\mathsf{out}})) : \widehat{v}_i^x(\alpha_{\mathsf{in}}) = \zeta_{\mathsf{in}} \ \wedge \ \widehat{v}_{i+1}^x(\alpha_{\mathsf{out}}) = \zeta_{\mathsf{out}}\}, \\ R^V_{\mathsf{next\text{-}layer}} &= \{(((\langle x,i\rangle,\alpha_{\mathsf{in}},\zeta_{\mathsf{in}}),(\alpha_{\mathsf{out}},\zeta_{\mathsf{out}})) : \widehat{v}_i^x(\alpha_{\mathsf{in}}) \neq \zeta_{\mathsf{in}} \ \wedge \ \widehat{v}_{i+1}^x(\alpha_{\mathsf{out}}) \neq \zeta_{\mathsf{out}}\}. \end{aligned} \tag{5}$$

Remark 4.5. *The pair $\langle x,i\rangle$, provided as part of the input to the next-layer problem $(Q,R)^V_{\mathsf{next\text{-}layer}}$, serves as a succinct representation of the encoding pair $(\widehat{v}_i^x,\widehat{v}_{i+1}^x)$. We distinguish this succinct representation $\langle x,i\rangle$ from the actual input $(\alpha_{\mathsf{in}},\zeta_{\mathsf{in}})$, since it captures the* setting *of the problem $(Q,R)^V_{\mathsf{next\text{-}layer}}$ (i.e., the polynomials that the problem refers to) rather than its actual input, similar to specifying the polynomial family V as a superscript in $(Q,R)^V_{k\mathsf{eqs}\mapsto 1\mathsf{eq}}$. We therefore refer to $\langle x,i\rangle$ as the* context *for the next-layer problem $(Q,R)^V_{\mathsf{next\text{-}layer}}$.*[33] *This succinct representation is useful for protocols that invoke the next-layer problem $(Q,R)^V_{\mathsf{next\text{-}layer}}$ as an oracle, where the explicit polynomials $\widehat{v}_0^x,\ldots,\widehat{v}_d^x$ are too costly for the searcher to compute or even read (as in the protocol for L we present below). The concept of contexts is widely used across the following sections.*

4.2.2 The protocol

Given a string $x \in \{0,1\}^n$, the searcher wants to verify that $x \in L$, which is equivalent to proving $\widehat{v}_0^x(1) = 1$. The verification process is carried out by iteratively solving the next-layer problem $(Q,R)^V_{\mathsf{next\text{-}layer}}$ via oracle access.

In the first iteration, the parties solve the next-layer problem $(Q,R)^V_{\mathsf{next\text{-}layer}}$ for the element $1 \in \mathbb{F}^\ell$, the value $1 \in \mathbb{F}$, and the context $\langle x,0\rangle$, which correspond to the assertion $\widehat{v}_0^x(1) = 1$. Then, given a solution $(\alpha_1,\zeta_1) \in \mathbb{F}^\ell \times \mathbb{F}$ as an answer, the parties again use their oracle access to $(Q,R)^V_{\mathsf{next\text{-}layer}}$ to solve the next-layer problem for the element α_1, the value ζ_1, and the context $\langle x,1\rangle$. Thus, by repeating this process for d times, each time using the solution attained in the previous iteration as an input for $(Q,R)^V_{\mathsf{next\text{-}layer}}$ (together with the context of x and the corresponding index), the initiating claim $\widehat{v}_0^x(1) = 1$ is reduced to an assertion regarding the input layer of the computation, $\widehat{v}_d^x$. We stress that in each oracle-calling step, the guide is the one submitting the queries (as in any oracle-aided system),[34] and the searcher submits the expected query length.

Finally, given an element $\alpha_d \in \mathbb{F}^\ell$ and value $\zeta_d \in \mathbb{F}$, obtained as an answer from the last oracle call to the next-layer problem $(Q,R)^V_{\mathsf{next\text{-}layer}}$, the searcher's last task is to verify that indeed $\widehat{v}_d^x(\alpha_d) = \zeta_d$. This is done by using an oracle access to the check-eval problem (which is actually a function) $(Q,R)^V_{\mathsf{check\text{-}eval}} := \{(\langle x\rangle,(\alpha,\zeta),b) : b = 1 \iff \widehat{v}_d^x(\alpha) = \zeta\}$ in the natural way. Before deciding on his output, the searcher makes a query-check, in which he verifies that the guide submitted the correct query at each oracle-calling step (by looking at the received answers, which include the corresponding query) and otherwise rejects (by outputting $\perp$). The protocol ends with the searcher outputting the

[33] The notion of contexts is implicitly used in the proof systems literature - e.g., the set or circuit a proof system is instantiated for (e.g., the set L or its corresponding circuit family $\{C_n\}_n$ in the case of [GKR15]), and the polynomials over which the sum-check protocol is executed.

[34] See Remark 4.5.

answer received from the check-eval problem $(Q,R)^V_{\mathsf{check\text{-}eval}}$.

The protocol is summarized in the following construction.

Construction 4.6 (OAIP for L)**.** *Given an input x, the parties set $\alpha_0 := 1$, $\zeta_0 := 1$ and proceed as follows.*

- *For $i = 0, \ldots, d-1$, the guide submits the query $(\langle x, i\rangle, \alpha_i, \zeta_i)$ to the next-layer problem $(Q,R)^V_{\mathsf{next\text{-}layer}}$ (and the searcher submits the query's expected length). Set $(\alpha_{i+1}, \zeta_{i+1}) \in \mathbb{F}^\ell \times \mathbb{F}$ to be the answer attained from $(Q,R)^V_{\mathsf{next\text{-}layer}}$.*
- *The guide submits (α_d, ζ_d) and the context $\langle x\rangle$ to the check-eval problem $(Q,R)^V_{\mathsf{check\text{-}eval}}$ (and the searcher submits the query's expected length). Set $b \in \{0,1\}$ to be the answer attained from $(Q,R)^V_{\mathsf{check\text{-}eval}}$.*
- Query-check: *The searcher verifies that the guide submitted the correct query at each oracle-calling step, and otherwise rejects.*
- *The searcher outputs the value b (attained from the check-eval problem $(Q,R)^V_{\mathsf{check\text{-}eval}}$).*

Proposition 4.7 (oracle-aided IP for $\mathcal{UC}irc(s,d,\mathsf{T},\mathsf{S})$)**.** *Let $L \in \mathcal{UC}irc(s,d,\mathsf{T},\mathsf{S})$ for some s, d, T and S. The protocol in Construction 4.6 is an ideal[35] oracle-aided interactive proof system for L.*

Proof. For completeness, let $x \in L$. Since $\widehat{v}^x_0(1) = 1$, the first query $(\langle x, 0\rangle, 1, 1)$ submitted to the next-layer oracle $(Q,R)^V_{\mathsf{next\text{-}layer}}$ by the prescribed guide is qualified (i.e., $(\langle x, 0\rangle, 1, 1) \in \mathrm{Dom}(Q^V_{\mathsf{next\text{-}layer}})$). By the functionality of the next-layer problem $(Q,R)^V_{\mathsf{next\text{-}layer}}$, all subsequent queries submitted to $(Q,R)^V_{\mathsf{next\text{-}layer}}$ during the protocol are qualified as well. Therefore, the element $\alpha_d \in \mathbb{F}^\ell$ and value $\zeta_d \in \mathbb{F}$ attained by the final oracle call to $(Q,R)^V_{\mathsf{next\text{-}layer}}$ satisfy $\widehat{v}^x_d(\alpha_d) = \zeta_d$, so the check-eval oracle $(Q,R)^V_{\mathsf{check\text{-}eval}}$ returns 1.

For soundness, let $x \notin L$ and fix some malicious guide strategy. Since $x \notin L$, we have $\widehat{v}^x_0(1) \neq 1$, hence the first correct query $(\langle x, 0\rangle, 1, 1)$ to the next-layer oracle $(Q,R)^V_{\mathsf{next\text{-}layer}}$ is robust (i.e., $(\langle x, 0\rangle, 1, 1)$ lies in $\mathrm{Dom}(R^V_{\mathsf{next\text{-}layer}})$). Note that submitting any other query will cause the searcher to reject at the query-check. By the functionality of the next-layer problem $(Q,R)^V_{\mathsf{next\text{-}layer}}$, all subsequent correct queries to $(Q,R)^V_{\mathsf{next\text{-}layer}}$ are robust as well. Hence, if the guide submitted the correct query at each iteration, then $\widehat{v}^x_d(\alpha_d) \neq \zeta_d$. Submitting any query to the check-eval oracle $(Q,R)^V_{\mathsf{check\text{-}eval}}$ other than the correct one $(\langle x\rangle, \alpha_d, \zeta_d)$ will cause the searcher to reject at the query-check. If instead the guide submits the correct query $(\langle x\rangle, \alpha_d, \zeta_d)$, then $(Q,R)^V_{\mathsf{check\text{-}eval}}$ outputs 0, leading the searcher to output 0, which is a robust output for x. □

We provide the complexity properties of the main system (Construction 4.6) for future reference.

Proposition 4.8 (complexity of Construction 4.6)**.** *The main system Π^L_{ucirc} of Construction 4.6 has the following properties: the system uses d sequential oracle calls to the next-layer problem $(Q,R)^V_{\mathsf{next\text{-}layer}}$ followed by a single oracle call to the check-eval problem $(Q,R)^V_{\mathsf{check\text{-}eval}}$. The prescribed (honest) guide runs in time $O(d \log |\mathbb{F}|)$. The searcher runs in time $O(d \log |\mathbb{F}|)$. The system is public coin and has no communication rounds.*

[35] Recall that an interactive search system is ideal if it has perfect completeness and perfect soundness.

4.3 Oracle-aided IS for next-layer $(Q,R)^V_{\mathsf{next\text{-}layer}}$

In this section, we present an oracle-aided interactive search system for the next-layer problem $(Q,R)^V_{\mathsf{next\text{-}layer}}$ defined in (5).

4.3.1 The search problems that the protocol uses

The oracle-aided interactive search system $\Pi^V_{\mathsf{next\text{-}layer}}$ we are about to present uses oracle access to three D-search problems, all referring to a field $\mathbb{F}$, a subset $H \subseteq \mathbb{F}$, and an integer $\ell = \lceil \log_{|H|} s \rceil$.

The sum-check problem $(Q,R)^{\mathcal{F}}_{\mathsf{sum\text{-}check}}$. For a family of polynomials $\mathcal{F} \subseteq \{f : \mathbb{F}^\ell \to \mathbb{F}\}$, the (D-search version of the) sum-check problem $(Q,R)^{\mathcal{F}}_{\mathsf{sum\text{-}check}}$ is defined as follows. Given a context $\langle f \rangle$ specifying a polynomial f in $\mathcal{F}$ and a value $\zeta_{\mathsf{in}} \in \mathbb{F}$ (which allegedly equals $\sum_{\beta \in H} f(\beta)$), the goal is to find an element $\alpha_{\mathsf{out}} \in \mathbb{F}^\ell$ and value $\zeta_{\mathsf{out}} \in \mathbb{F}$ such that $f(\alpha_{\mathsf{out}}) = \zeta_{\mathsf{out}}$ if and only if $\sum_{\beta \in H^\ell} f(\beta) = \zeta_{\mathsf{in}}$. We define the qualified (resp. robust) relation $Q^{\mathcal{F}}_{\mathsf{sum\text{-}check}}$ (resp. $R^{\mathcal{F}}_{\mathsf{sum\text{-}check}}$) to consist of all pairs of correct (resp. incorrect) equations.

$$Q^{\mathcal{F}}_{\mathsf{sum\text{-}check}} = \left\{ ((\langle f \rangle, \zeta_{\mathsf{in}}), (\alpha_{\mathsf{out}}, \zeta_{\mathsf{out}})) : \sum_{\beta \in H^\ell} f(\beta) = \zeta_{\mathsf{in}} \ \wedge \ f(\alpha_{\mathsf{out}}) = \zeta_{\mathsf{out}} \right\}, \tag{6}$$

$$R^{\mathcal{F}}_{\mathsf{sum\text{-}check}} = \left\{ ((\langle f \rangle, \zeta_{\mathsf{in}}), (\alpha_{\mathsf{out}}, \zeta_{\mathsf{out}})) : \sum_{\beta \in H^\ell} f(\beta) \neq \zeta_{\mathsf{in}} \ \wedge \ f(\alpha_{\mathsf{out}}) \neq \zeta_{\mathsf{out}} \right\}.$$

Clearly, $(Q,R)^{\mathcal{F}}_{\mathsf{sum\text{-}check}}$ formalizes the functionality of the sum-check protocol of [LFKN92] as a D-search problem. Indeed, we later use a variant of the sum-check protocol to solve $(Q,R)^{\mathcal{F}}_{\mathsf{sum\text{-}check}}$.

The equation-merging problem $(Q,R)^{\mathcal{F}}_{k\mathsf{eqs}\mapsto 1\mathsf{eq}}$. Fix a family of polynomials $\mathcal{F} \subseteq \{f : \mathbb{F}^\ell \to \mathbb{F}\}$. The second search problem is the equation-merging problem $(Q,R)^{\mathcal{F}}_{k\mathsf{eqs}\mapsto 1\mathsf{eq}}$ in which given k input-value pairs $(\alpha_1,\zeta_1),\ldots,(\alpha_k,\zeta_k) \in \mathbb{F}^\ell \times \mathbb{F}$ for some $k \in \mathbb{N}$ and a context $\langle f \rangle$, specifying a polynomial in $\mathcal{F}$, the goal is to find a *single* element $\alpha_{\mathsf{out}} \in \mathbb{F}^\ell$ and value $\zeta_{\mathsf{out}} \in \mathbb{F}$ such that $f(\alpha_{\mathsf{out}}) = \zeta_{\mathsf{out}}$ if and only if $f(\alpha_i) = \zeta_i$ for *every* $i \in [k]$. In similar to before, we define

$$\begin{aligned} Q^{\mathcal{F}}_{k\mathsf{eqs}\mapsto 1\mathsf{eq}} &:= \{(\langle f \rangle, (\alpha_1,\zeta_1),\ldots,(\alpha_k,\zeta_k)), (\alpha_{\mathsf{out}},\zeta_{\mathsf{out}}) : f(\alpha_{\mathsf{out}}) = \zeta_{\mathsf{out}} \ \wedge \ \forall i \in [k],\ f(\alpha_i) = \zeta_i\}, \\ R^{\mathcal{F}}_{k\mathsf{eqs}\mapsto 1\mathsf{eq}} &:= \{(\langle f \rangle, (\alpha_1,\zeta_1),\ldots,(\alpha_k,\zeta_k)), (\alpha_{\mathsf{out}},\zeta_{\mathsf{out}}) : \exists i \in [k],\ f(\alpha_i) \neq \zeta_i \ \wedge \ f(\alpha_{\mathsf{out}}) \neq \zeta_{\mathsf{out}}\}. \end{aligned} \tag{7}$$

Note that the equation-merging problem is a solution-merging problem, as defined in Definition 3.8.

The Φ-evaluation problem $(Q,R)^{\Phi}_{\mathsf{eval}}$. The last search problem refers to the polynomial family $\Phi := \{\widehat{\phi}^n_i\}_{n \in \mathbb{N}, i \in [d]}$ defined in Observation 4.3. Letting $\widehat{v}^x_i, \widehat{v}^x_{i+1} : \mathbb{F}^\ell \to \mathbb{F}$ be polynomials that correspond to adjacent layers of $C_{|x|}$, recall that $\widehat{\phi}^{|x|}_i : (\mathbb{F}^\ell)^3 \to \mathbb{F}$ formalizes the relation between the polynomials

$\widehat{v}_i^x$ and $\widehat{v}_{i+1}^x$, representing the i^{th} and $i+1^{\text{st}}$ layers of the circuit C_n, respectively, and satisfies

$$\widehat{v}_i^x(X) = \sum_{\beta,\beta',\beta'' \in H^\ell} \mathsf{EQ}(X,\beta) \cdot \widehat{\phi}_i^{|x|}(\beta,\beta',\beta'') \cdot (1 - \widehat{v}_{i+1}^x(\beta') \cdot \widehat{v}_{i+1}^x(\beta'')) \tag{8}$$

which is identical to (4). During the protocol, the parties need to evaluate $\widehat{\phi}_i^{|x|}$ on given elements, which we formalize as solving the Φ-evaluation problem $(Q,R)^{\Phi}_{\mathsf{eval}}$ such that

$$\begin{aligned} Q^{\Phi}_{\mathsf{eval}} &= \left\{ \left((\langle n,i \rangle, \alpha, \alpha', \alpha'') \, , \widehat{\phi}_i^n \, (\alpha, \alpha', \alpha'') \right) \right\}, \\ R^{\Phi}_{\mathsf{eval}} &= \emptyset. \end{aligned} \tag{9}$$

4.3.2 The protocol $\Pi^V_{\mathsf{next\text{-}layer}}$

Given an element $\alpha_{\mathsf{in}} \in \mathbb{F}^\ell$ and a value $\zeta_{\mathsf{in}} \in \mathbb{F}$ along with a context $\langle x,i \rangle$ specifying the polynomial $\widehat{v}_i^x$ in V, the parties interpret the input as a claim asserting that $\widehat{v}_i^x(\alpha_{\mathsf{in}}) = \zeta_{\mathsf{in}}$ and extend this equation as in (8) to obtain the equivalent claim

$$\sum_{\beta,\beta',\beta'' \in H^\ell} \mathsf{EQ}(\alpha_{\mathsf{in}},\beta) \cdot \widehat{\phi}_i^n(\beta,\beta',\beta'') \cdot (1 - \widehat{v}_{i+1}^x(\beta') \cdot \widehat{v}_{i+1}^x(\beta'')) = \zeta_{\mathsf{in}}. \tag{10}$$

Denote the polynomial nested in (10)'s summations by

$$f^x_{i,\alpha_{\mathsf{in}}}(X,X',X'') := \mathsf{EQ}(\alpha_{\mathsf{in}},X) \cdot \widehat{\phi}_i^n(X,X',X'') \cdot (1 - \widehat{v}_{i+1}^x(X') \cdot \widehat{v}_{i+1}^x(X'')), \tag{11}$$

and note that $f^x_{i,\alpha_{\mathsf{in}}} : \mathbb{F}^{3\ell} \to \mathbb{F}$ is determined by x, i and α_{in}. Set the polynomial family $\mathcal{F} := \{f^x_{i,\alpha}\}_{x,i,\alpha}$, and let $\langle x,i,\alpha \rangle$ be the context specifying the polynomial $f^x_{i,\alpha}$ in $\mathcal{F}$. The parties then solve the sum-check problem $(Q,R)^{\mathcal{F}}_{\mathsf{sum\text{-}check}}$ for the polynomial $f^x_{i,\alpha_{\mathsf{in}}}$ and the value ζ_{in} by having the guide submit the query $(\langle x,i,\alpha_{\mathsf{in}} \rangle, \zeta_{\mathsf{in}})$ to $(Q,R)^{\mathcal{F}}_{\mathsf{sum\text{-}check}}$, and the searcher submitting the expected query length.

Upon receiving an answer $(\overline{\alpha}_{\mathsf{sc}}, \zeta_{\mathsf{sc}}) \in \mathbb{F}^{3\ell} \times \mathbb{F}$ from the sum-check oracle $(Q,R)^{\mathcal{F}}_{\mathsf{sum\text{-}check}}$, the parties parse $\overline{\alpha}_{\mathsf{sc}}$ as $\overline{\alpha}_{\mathsf{sc}} =: (\alpha_{\mathsf{sc}}, \alpha'_{\mathsf{sc}}, \alpha''_{\mathsf{sc}})$ in $(\mathbb{F}^\ell)^3$, and interpret the answer $((\alpha_{\mathsf{sc}}, \alpha'_{\mathsf{sc}}, \alpha''_{\mathsf{sc}}), \zeta_{\mathsf{sc}})$ as the claim asserting that $f^x_{i,\alpha_{\mathsf{in}}}(\alpha_{\mathsf{sc}}, \alpha'_{\mathsf{sc}}, \alpha''_{\mathsf{sc}}) = \zeta_{\mathsf{sc}}$, which by the definition of $f^x_{i,\alpha}$, is equivalent to the claim

$$\mathsf{EQ}(\alpha_{\mathsf{in}}, \alpha_{\mathsf{sc}}) \cdot \widehat{\phi}_i^n(\alpha_{\mathsf{sc}}, \alpha'_{\mathsf{sc}}, \alpha''_{\mathsf{sc}}) \cdot (1 - \widehat{v}_{i+1}^x(\alpha'_{\mathsf{sc}}) \cdot \widehat{v}_{i+1}^x(\alpha''_{\mathsf{sc}})) = \zeta_{\mathsf{sc}}. \tag{12}$$

Deriving (12) is a key step towards producing an output for the next-layer protocol $\Pi^V_{\mathsf{next\text{-}layer}}$ (i.e., a solution for $(Q,R)^V_{\mathsf{next\text{-}layer}}$), since the parties managed to reduce an equation regarding the i^{th} layer (represented by $\widehat{v}_i^x$) to an equation involving $\widehat{v}_{i+1}^x$. However, (12) contains two evaluations of $\widehat{v}_{i+1}^x$ instead of one, and additionally contains evaluations of EQ and $\widehat{\phi}_i^n$.

The latter is solved by having the searcher compute $\mathsf{EQ}(\alpha_{\mathsf{in}}, \alpha_{\mathsf{sc}})$ by himself, and using an oracle access to the Φ-evaluation problem $(Q,R)^{\Phi}_{\mathsf{eval}}$ in the natural way (i.e., the guide submits the query $(\langle n,i \rangle, \alpha_{\mathsf{sc}}, \alpha'_{\mathsf{sc}}, \alpha''_{\mathsf{sc}})$ to $(Q,R)^{\Phi}_{\mathsf{eval}}$, and the searcher verifies the submitted query).

The guide then computes the values $\widehat{v}_{i+1}^x(\alpha'_{\mathsf{sc}})$ and $\widehat{v}_{i+1}^x(\alpha''_{\mathsf{sc}})$ and sends them to the searcher. Denote the values actually received by the searcher by $\widetilde{\zeta'}$ and $\widetilde{\zeta''}$ (where allegedly, $\widetilde{\zeta'} = \widehat{v}_i^x(\alpha'_{\mathsf{sc}})$ and $\widetilde{\zeta''} = \widehat{v}_i^x(\alpha''_{\mathsf{sc}})$). The searcher checks the received values $\widetilde{\zeta'}$ and $\widetilde{\zeta''}$ by verifying equality in (12), and rejects

otherwise. The parties then "merge" the two claims asserting that $\widehat{v}^x_{i+1}(\alpha'_{\mathsf{sc}}) = \widetilde{\zeta}'$ and $\widehat{v}^x_{i+1}(\alpha''_{\mathsf{sc}}) = \widetilde{\zeta}''$ into one by using an oracle access to the equation-merging problem: the guide submits the input-value pairs $(\alpha'_{\mathsf{sc}}, \widetilde{\zeta}')$ and $(\alpha''_{\mathsf{sc}}, \widetilde{\zeta''})$ to $(Q,R)^V_{\mathsf{2eqs}\mapsto\mathsf{1eq}}$, along with the context $\langle x, i+1\rangle$ (which specifies $\widehat{v}^x_{i+1}$), and the searcher submits the expected query-length.

The protocol ends with the searcher performing a query-check (and rejecting if any of the submitted queries is incorrect) and outputting the answer obtained from the equation-merging problem $(Q,R)^V_{\mathsf{2eqs}\mapsto\mathsf{1eq}}$.

The protocol is summarized in the following construction.

Construction 4.9 (oracle-aided IS for next-layer $(Q,R)^V_{\mathsf{next\text{-}layer}}$)**.** *Given an element $\alpha_{\mathsf{in}} \in \mathbb{F}^\ell$ and a value $\zeta_{\mathsf{in}} \in \mathbb{F}$, along with context $\langle x, i\rangle \in \{0,1\}^n \times [d]$, the parties proceed as follows.*

- *The guide submits $(\langle x, i, \alpha_{\mathsf{in}}\rangle, \zeta_{\mathsf{in}})$ to the sum-check oracle $(Q,R)^{\mathcal{F}}_{\mathsf{sum\text{-}check}}$, and the searcher submits the query's expected length. Letting $(\overline{\alpha}_{\mathsf{sc}}, \zeta_{\mathsf{sc}}) \in \mathbb{F}^{3\ell} \times \mathbb{F}$ be the answer received from the sum-check oracle $(Q,R)^{\mathcal{F}}_{\mathsf{sum\text{-}check}}$, the parties parse the first entry $\overline{\alpha}_{\mathsf{sc}}$ as $(\alpha_{\mathsf{sc}}, \alpha'_{\mathsf{sc}}, \alpha''_{\mathsf{sc}}) \in (\mathbb{F}^\ell)^3$.*

- *The searcher computes $\mathsf{EQ}(\alpha_{\mathsf{in}}, \alpha_{\mathsf{sc}})$ by himself. The guide submits the query $(\langle n, i\rangle, \alpha_{\mathsf{sc}}, \alpha'_{\mathsf{sc}}, \alpha''_{\mathsf{sc}})$ to the Φ-evaluation oracle $(Q,R)^{\Phi}_{\mathsf{eval}}$, and the searcher submits the query's expected length. Denote the oracle's answer by $\zeta_{\widehat{\phi}} \in \mathbb{F}$.*

  The guide computes the values $\widehat{v}^x_{i+1}(\alpha'_{\mathsf{sc}})$ and $\widehat{v}^x_{i+1}(\alpha''_{\mathsf{sc}})$, and sends them to the searcher.

- *Upon receiving values $\widetilde{\zeta}'$ and $\widetilde{\zeta''}$ (where allegedly, $\widetilde{\zeta}' = \widehat{v}^x_{i+1}(\alpha'_{\mathsf{sc}})$ and $\widetilde{\zeta''} = \widehat{v}^x_{i+1}(\alpha''_{\mathsf{sc}})$), the searcher checks that*

$$\mathsf{EQ}(\alpha_{\mathsf{in}}, \alpha_{\mathsf{sc}}) \cdot \zeta_{\widehat{\phi}} \cdot (1 - \widetilde{\zeta}' \cdot \widetilde{\zeta''}) = \zeta_{\mathsf{sc}}$$

  and otherwise rejects.

- *The guide submits the query $(\langle x, i+1\rangle, (\alpha'_{\mathsf{sc}}, \widetilde{\zeta}'), (\alpha''_{\mathsf{sc}}, \widetilde{\zeta''}))$ to the equation-merging oracle $(Q,R)^V_{\mathsf{2eqs}\mapsto\mathsf{1eq}}$, and the searcher submits the query's expected length. Let $(\alpha_{\mathsf{out}}, \zeta_{\mathsf{out}}) \in \mathbb{F}^\ell \times \mathbb{F}$ denote the answer attained from $(Q,R)^V_{\mathsf{2eqs}\mapsto\mathsf{1eq}}$.*

- *The searcher then performs a query-check, rejects if the query-check fails, and otherwise outputs the answer $(\alpha_{\mathsf{out}}, \zeta_{\mathsf{out}})$ received from the equation-merging oracle $(Q,R)^V_{\mathsf{2eqs}\mapsto\mathsf{1eq}}$.*

The properties of Construction 4.9 are given in the following propositions.

Proposition 4.10. *The protocol $\Pi^V_{\mathsf{next\text{-}layer}}$ of Construction 4.9 is an ideal oracle-aided interactive search system for the next-layer problem $(Q,R)^V_{\mathsf{next\text{-}layer}}$.*

Proof. Perfect completeness easily follows from the definition of $(Q,R)^V_{\mathsf{next\text{-}layer}}$ and the construction's overview (see (5) and Section 4.3.2).

For soundness, let $(\langle x, i\rangle, \alpha_{\mathsf{in}}, \zeta_{\mathsf{in}}) \in \mathrm{Dom}(R^V_{\mathsf{next\text{-}layer}})$ be a robust input and fix an arbitrary malicious guide strategy. By definition of the robust next-layer relation $R^V_{\mathsf{next\text{-}layer}}$, we have $\widehat{v}^x_i(\alpha_{\mathsf{in}}) \neq \zeta_{\mathsf{in}}$, which

(by (10) and the definition of $f^x_{i,\alpha_{\mathsf{in}}}$ in (11)) is equivalent to

$$\sum_{\beta,\beta',\beta''\in H^\ell} f^x_{i,\alpha_{\mathsf{in}}}(\beta,\beta',\beta'') \neq \zeta_{\mathsf{in}}.$$

Thus, the correct query $(\langle x,i,\alpha_{\mathsf{in}}\rangle,\zeta_{\mathsf{in}})$ for submitting to the sum-check oracle $(Q,R)^{\mathcal{F}}_{\mathsf{sum\text{-}check}}$ is robust. Together with the searcher's query-check, this implies that either the guide submits the sum-check oracle $(Q,R)^{\mathcal{F}}_{\mathsf{sum\text{-}check}}$ with a robust query (in $\mathrm{Dom}(R^{\mathcal{F}}_{\mathsf{sum\text{-}check}})$), or the searcher will later reject. If the guide indeed submitted the correct (robust) query, then, by the guarantee of the sum-check oracle $(Q,R)^{\mathcal{F}}_{\mathsf{sum\text{-}check}}$, the received answer $((\alpha_{\mathsf{sc}},\alpha'_{\mathsf{sc}},\alpha''_{\mathsf{sc}}),\zeta_{\mathsf{sc}})$ satisfies $f^x_{i,\alpha_{\mathsf{in}}}(\alpha_{\mathsf{sc}},\alpha'_{\mathsf{sc}},\alpha''_{\mathsf{sc}}) \neq \zeta_{\mathsf{sc}}$ which, by the definition of $f^x_{i,\alpha_{\mathsf{in}}}$, is equivalent to

$$\mathsf{EQ}(\alpha_{\mathsf{in}},\alpha_{\mathsf{sc}})\cdot\widehat{\phi}^n_i(\alpha_{\mathsf{sc}},\alpha'_{\mathsf{sc}},\alpha''_{\mathsf{sc}})\cdot(1-\widehat{v}^x_{i+1}(\alpha'_{\mathsf{sc}})\cdot\widehat{v}^x_{i+1}(\alpha''_{\mathsf{sc}}))\neq\zeta_{\mathsf{sc}}. \tag{13}$$

Now, recall that the searcher computes $\mathsf{EQ}(\alpha_{\mathsf{in}},\alpha_{\mathsf{sc}})$ by himself and is therefore certain of its value. In addition, if the guide does not submit the correct query $(\langle n,i\rangle,\alpha_{\mathsf{sc}},\alpha'_{\mathsf{sc}},\alpha''_{\mathsf{sc}})$ to the Φ-evaluation oracle $(Q,R)^{\Phi}_{\mathsf{eval}}$, the searcher will catch that in the query-check and reject. On the other hand, if the guide does submit the correct query to $(Q,R)^{\Phi}_{\mathsf{eval}}$, then providing the searcher with the correct values $\widetilde{\zeta'}=\widehat{v}^x_{i+1}(\alpha'_{\mathsf{sc}})$ and $\widetilde{\zeta''}=\widehat{v}^x_{i+1}(\alpha''_{\mathsf{sc}})$ would result with the searcher noticing the inequality in (13), and will therefore reject.

Otherwise, either $\widetilde{\zeta'}\neq\widehat{v}^x_{i+1}(\alpha'_{\mathsf{sc}})$ or $\widetilde{\zeta''}\neq\widehat{v}^x_{i+1}(\alpha''_{\mathsf{sc}})$, hence the correct query $(\langle x,i+1\rangle,(\alpha'_{\mathsf{sc}},\widetilde{\zeta'}),(\alpha''_{\mathsf{sc}},\widetilde{\zeta''}))$ for submitting to the equation-merging oracle $(Q,R)^V_{k\mathsf{eqs}\mapsto 1\mathsf{eq}}$ is robust (and submitting any other query leads the searcher to reject at the query-check). Soundness then followed by the functionality of the robust equation-merging relation $R^V_{k\mathsf{eqs}\mapsto 1\mathsf{eq}}$. □

The complexity features of Construction 4.9 are summarized below.

Proposition 4.11 (complexity of Construction 4.9)**.** *The next-layer system $\Pi^V_{\mathsf{next\text{-}layer}}$ of Construction 4.6 has the following properties: the system makes a single oracle call to the sum-check problem $(Q,R)^{\mathcal{F}}_{\mathsf{sum\text{-}check}}$ for the polynomial family $\mathcal{F}=\{f^x_{i,\alpha}\}$ defined in (11), a single oracle call to the Φ-evaluation problem $(Q,R)^{\Phi}_{\mathsf{eval}}$, and a single oracle call to the equation-merging problem $(Q,R)^V_{2\mathsf{eqs}\mapsto 1\mathsf{eq}}$ for $V=\{\widehat{v}^x_i\}$ and two submitted equations. The prescribed (honest) guide strategy runs in time $O(\mathsf{T}+\mathrm{poly}(s)+\ell\log|\mathbb{F}|))$, and the searcher runs in time $O(\ell|H|\mathrm{polylog}\,|\mathbb{F}|)$. The protocol is public coin, has a single communication round, and has communication complexity $O(\log|\mathbb{F}|)$.*

4.4 Oracle-aided IS for sum-check $(Q,R)_{\mathsf{sum\text{-}check}}$

We present an oracle-aided interactive search system $\Pi_{\mathsf{sum\text{-}check}}$ for the sum-check problem $(Q,R)_{\mathsf{sum\text{-}check}}$ (defined in (6)), which is based on the sum-check protocol of [LFKN92]. For simplicity, we assume that the sum-check problem $(Q,R)_{\mathsf{sum\text{-}check}}$ refers to a *single* fixed multivariate polynomial $f:\mathbb{F}^\ell\to\mathbb{F}$ rather than a polynomial family (the protocol can be trivially generalized for a set of polynomials by providing a context specifying a polynomial within the set).

The system $\Pi_{\mathsf{sum\text{-}check}}$ for sum-check uses oracle access to the following D-search problem.

The H-sum$\mapsto$eval problem $(Q,R)_{H\mathsf{sum}\mapsto\mathsf{eval}}$. For a family of *univariate* polynomials $\mathcal{G}=\{g:\mathbb{F}\to\mathbb{F}\}$ and a subset $H\subseteq\mathbb{F}$, the H-sum$\mapsto$eval problem $(Q,R)^{\mathcal{G}}_{H\mathsf{sum}\mapsto\mathsf{eval}}$ receives a value $\zeta_{\mathsf{in}}\in\mathbb{F}$, along with a context $\langle g\rangle$ (specifying a polynomial $g\in\mathcal{G}$), and returns an input-value pair $(\alpha_{\mathsf{out}},\zeta_{\mathsf{out}})$ such that $g(\alpha_{\mathsf{out}})=\zeta_{\mathsf{out}}$ if and only if $\sum_{\beta\in H}g(\beta)=\zeta_{\mathsf{in}}$. As before, we define the qualified and robust relations as follows.

$$Q^{\mathcal{G}}_{H\mathsf{sum}\mapsto\mathsf{eval}}=\left\{((\langle g\rangle,\zeta_{\mathsf{in}}),(\alpha_{\mathsf{out}},\zeta_{\mathsf{out}})):\sum_{\beta\in H}g(\beta)=\zeta_{\mathsf{in}}\ \wedge\ g(\alpha_{\mathsf{out}})=\zeta_{\mathsf{out}}\right\},\tag{14}$$

$$R^{\mathcal{G}}_{H\mathsf{sum}\mapsto\mathsf{eval}}=\left\{((\langle g\rangle,\zeta_{\mathsf{in}}),(\alpha_{\mathsf{out}},\zeta_{\mathsf{out}})):\sum_{\beta\in H}g(\beta)\neq\zeta_{\mathsf{in}}\ \wedge\ g(\alpha_{\mathsf{out}})\neq\zeta_{\mathsf{out}}\right\}.$$

In the context of the sum-check protocol, we use the H-sum$\mapsto$eval problem $(Q,R)_{H\mathsf{sum}\mapsto\mathsf{eval}}$ for the polynomial family $\mathcal{G}=\{g_{\alpha_1,\ldots,\alpha_i}:\mathbb{F}\to\mathbb{F}\}_{i\in\{0,\ldots,\ell-1\},\alpha_1,\ldots,\alpha_i\in\mathbb{F}}$, where for some fixed index $i\in\{0,\ldots,\ell-1\}$, elements $\alpha_1,\ldots,\alpha_i\in\mathbb{F}$, and $f:\mathbb{F}^\ell\to\mathbb{F}$ being the polynomial considered by the sum-check system (i.e., its context), we define

$$g_{\alpha_1,\ldots,\alpha_i}(X)\coloneqq\sum_{\beta_{i+2},\ldots,\beta_\ell\in H}f(\alpha_1,\ldots,\alpha_i,X,\beta_{i+2},\ldots,\beta_\ell).\tag{15}$$

Thus, in the context of the sum-check system for $(Q,R)^f_{\mathsf{sum\text{-}check}}$, the H-sum$\mapsto$eval problem $(Q,R)^{\mathcal{G}}_{H\mathsf{sum}\mapsto\mathsf{eval}}$ is defined by

$$Q^{\mathcal{G}}_{H\mathsf{sum}\mapsto\mathsf{eval}}=\left\{((\langle\alpha_1,\ldots,\alpha_i\rangle,\zeta_{\mathsf{in}}),(\alpha_{\mathsf{out}},\zeta_{\mathsf{out}})):\sum_{\beta\in H}g_{\alpha_1,\ldots,\alpha_i}(\beta)=\zeta_{\mathsf{in}}\ \wedge\ g_{\alpha_1,\ldots,\alpha_i}(\alpha_{\mathsf{out}})=\zeta_{\mathsf{out}}\right\},$$

$$R^{\mathcal{G}}_{H\mathsf{sum}\mapsto\mathsf{eval}}=\left\{((\langle\alpha_1,\ldots,\alpha_i\rangle,\zeta_{\mathsf{in}}),(\alpha_{\mathsf{out}},\zeta_{\mathsf{out}})):\sum_{\beta\in H}g_{\alpha_1,\ldots,\alpha_i}(\beta)\neq\zeta_{\mathsf{in}}\ \wedge\ g_{\alpha_1,\ldots,\alpha_i}(\alpha_{\mathsf{out}})\neq\zeta_{\mathsf{out}}\right\},$$

where we use $\langle\alpha_1,\ldots,\alpha_i\rangle$ as a succinct representation of the polynomial $g_{\alpha_1,\ldots,\alpha_i}\in\mathcal{G}$.

4.4.1 The protocol

Given a value $\zeta_{\mathsf{in}}\in\mathbb{F}$, the sum-check system $\Pi^f_{\mathsf{sum\text{-}check}}$ for $(Q,R)^f_{\mathsf{sum\text{-}check}}$ is carried by gradually stripping summations out of the equation

$$\sum_{\beta_1,\ldots,\beta_\ell\in H}f(\beta_1,\ldots,\beta_\ell)\stackrel{?}{=}\zeta_{\mathsf{in}},$$

while maintaining its correctness. This gradual stripping is done in ℓ iterations, where in each iteration a single summation is being removed by using the H-sum$\mapsto$eval problem $(Q,R)^{\mathcal{G}}_{H\mathsf{sum}\mapsto\mathsf{eval}}$. The protocol is formalized in the following construction.

Construction 4.12 (oracle-aided IS for $(Q,R)^f_{\mathsf{sum\text{-}check}}$)**.** *Given a value $\zeta_{\mathsf{in}}\in\mathbb{F}$, the parties set $\zeta_0\coloneqq\zeta_{\mathsf{in}}$ and proceed as follows.*

- *For $i = 0, \ldots, \ell - 1$,*
  - *The guide submits the value ζ_i and the context $\langle \alpha_1, \ldots, \alpha_i \rangle$ (which specifies $g_{\alpha_1,\ldots,\alpha_i} \in \mathcal{G}$) to the H-sum $\mapsto$ eval oracle $(Q, R)^{\mathcal{G}}_{H\mathsf{sum}\mapsto\mathsf{eval}}$ (and the searcher submits the query's expected length). Let $(\alpha_{i+1}, \zeta_{i+1}) \in \mathbb{F} \times \mathbb{F}$ denote the oracle's answer (where allegedly, $g_{\alpha_1,\ldots,\alpha_i}(\alpha_{i+1}) = \zeta_{i+1}$).*
- *The searcher performs a query-check and rejects if it fails. Otherwise, the searcher outputs $(\alpha_1, \ldots, \alpha_\ell) \in \mathbb{F}^\ell$ and the value $\zeta_\ell \in \mathbb{F}$.*

Proposition 4.13. *The protocol $\Pi^f_{\mathsf{sum\text{-}check}}$ of Construction 4.12 is an ideal oracle-aided interactive search system for the sum-check problem $(Q, R)^f_{\mathsf{sum\text{-}check}}$.*

Proof. Completeness follows easily for any (actually, the unique) qualified input in $\mathrm{Dom}(Q^f_{\mathsf{sum\text{-}check}}) = \{\sum_{\beta\in\mathbb{F}^\ell} f(\beta)\}$.

For soundness, let $\zeta_{\mathsf{in}} \in \mathrm{Dom}(R^f_{\mathsf{sum\text{-}check}})$ be a robust input (hence $\sum_{\beta\in\mathbb{F}^\ell} f(\beta) \neq \zeta_{\mathsf{in}}$) and fix an arbitrary malicious guide strategy. We prove by induction on $i \in \{0, \ldots, \ell\}$ that if the guide submits the correct query in the first i iterations, then

$$\sum_{\beta_i,\ldots,\beta_\ell} f(\alpha_1, \ldots, \alpha_i, \beta_{i+1}, \ldots, \beta_\ell) \neq \zeta_i.$$

Soundness immediately follows by using $i = \ell$. The base case $i = 0$ trivially holds by $\zeta_0 = \zeta_{\mathsf{in}} \in \mathrm{Dom}(R^f_{\mathsf{sum\text{-}check}})$. For the induction step, assume the claim holds for some $i \in [\ell - 1]$, and we prove it also holds for $i + 1$. By the induction hypothesis, if the guide submits the correct query in each of the first i iterations, then

$$\sum_{\beta\in H} g_{\alpha_1,\ldots,\alpha_i}(\beta) = \sum_{\beta\in H} \sum_{\beta_{i+2}\ldots,\beta_\ell\in H} f(\alpha_1, \ldots, \alpha_i, \beta, \beta_{i+2}, \ldots, \beta_\ell) \neq \zeta_i. \tag{16}$$

Therefore, the correct query $(\langle \alpha_1, \ldots, \alpha_i \rangle, \zeta_i)$ for the H-sum$\mapsto$eval problem $(Q, R)^{\mathcal{G}}_{H\mathsf{sum}\mapsto\mathsf{eval}}$ is in the domain of the *robust* relation $R^{\mathcal{G}}_{H\mathsf{sum}\mapsto\mathsf{eval}}$, and if the guide submits any other query, the searcher will notice that during his query-check and reject. By the functionality of the H-sum $\mapsto$ eval problem $(Q, R)^{\mathcal{G}}_{H\mathsf{sum}\mapsto\mathsf{eval}}$, the received answer $(\alpha_{i+1}, \zeta_{i+1})$ must satisfy $g_{\alpha_1,\ldots,\alpha_i}(\alpha_{i+1}) \neq \zeta_{i+1}$, which implies

$$g_{\alpha_1,\ldots,\alpha_i}(\alpha_{i+1}) = \sum_{\beta_{i+2}\ldots,\beta_\ell\in H} f(\alpha_1, \ldots, \alpha_{i+1}, \beta_{i+2}, \ldots, \beta_\ell) \neq \zeta_{i+1},$$

completing the proof. □

The complexity features of Construction 4.12 are summarized below.

Proposition 4.14 (complexity of Construction 4.12)**.** *For a polynomial f of arity ℓ, the sum-check system $\Pi_{\mathsf{sum\text{-}check}}$ of Construction 4.12 has the following properties. The system uses ℓ sequential oracle call to the H-sum $\mapsto$ eval problem $(Q, R)^{\mathcal{G}}_{H\mathsf{sum}\mapsto\mathsf{eval}}$ for the polynomial family $\mathcal{G} = \{g_{\alpha_1,\ldots,\alpha_i} : \mathbb{F} \to \mathbb{F}\}_{i\in\{0,\ldots,\ell-1\},\alpha_1,\ldots,\alpha_i\in\mathbb{F}}$ defined in (15). The prescribed (honest) guide strategy runs in time*

$O(\ell|H|\log|\mathbb{F}|)$, *and the searcher runs in time* $O(\ell|H|\log|\mathbb{F}|)$. *The protocol is public coin and has no communication rounds.*

4.5 Interactive search system for H-sum$\mapsto$eval $(Q,R)_{H\mathsf{sum}\mapsto\mathsf{eval}}$

We present an oracle-aided interactive search system for the H-sum$\mapsto$eval problem $(Q,R)_{H\mathsf{sum}\mapsto\mathsf{eval}}$, defined in (14). The system uses oracle access to the equation-merging problem $(Q,R)_{k\mathsf{eqs}\mapsto 1\mathsf{eq}}$, presented and defined in (7). As before, we present the H-sum$\mapsto$eval protocol for some fixed single polynomial $g:\mathbb{F}\to\mathbb{F}$, where the construction can be generalized for any family of univariate polynomials.

Construction 4.15 (IS for $(Q,R)^g_{H\mathsf{sum}\mapsto\mathsf{eval}}$)**.** *Given an input* $\zeta_{\mathsf{in}}\in\mathbb{F}$*, the parties proceed as follows.*

- *The guide evaluates the polynomial* g *over the entire set* H*, and sends the resulting input-evaluation sequence* $(\beta,g(\beta))_{\beta\in H}$ *to the searcher.*
- *Upon receiving a sequence of paired values* $(\beta,\widetilde{\zeta}_\beta)_{\beta\in H}$ *(where allegedly* $\widetilde{\zeta}_\beta=g(\beta)$ *for each* $\beta\in H$*), the searcher verifies that* $\sum_{\beta\in H}\widetilde{\zeta}_\beta=\zeta_{\mathsf{in}}$ *and otherwise rejects.*
- *The guide submits the sequence* $(\beta,\widetilde{\zeta}_\beta)_{\beta\in H}$ *as a query to the equation-merging problem* $(Q,R)^g_{|H|\mathsf{eqs}\mapsto 1\mathsf{eq}}$ *(and the searcher submits the query's expected length).*

  The searcher then performs a query-check, rejects if the query-check fails, and otherwise outputs the answer received from $(Q,R)^g_{|H|\mathsf{eqs}\mapsto 1\mathsf{eq}}$.

The properties of Construction 4.15 are given below.

Proposition 4.16. *Construction 4.15 is an ideal oracle-aided interactive search system for the* H*-sum*$\mapsto$*eval problem* $(Q,R)^g_{H\mathsf{sum}\mapsto\mathsf{eval}}$*. Moreover, the prescribed (honest) guide strategy runs in time* $\deg(g)\cdot\mathrm{polylog}(|\mathbb{F}|)$ *and the searcher runs in time* $O(|H|\log|\mathbb{F}|)$*. The system is public coin and has a single communication round.*

Proof. Completeness easily follows for the unique qualified input $\zeta_{\mathsf{in}}=\sum_{\beta\in H}g(\beta)$.

For soundness, let $\zeta_{\mathsf{in}}\in\mathrm{Dom}(R^g_{H\mathsf{sum}\mapsto\mathsf{eval}})$ be a robust input (hence $\zeta_{\mathsf{in}}\neq\sum_{\beta\in H}g(\beta)$) and fix some malicious guide strategy. If the guide sends the searcher the correct evaluations $(\beta,g(\beta))_{\beta\in H}$, then the searcher notices the inequality

$$\sum_{\beta\in H}\widetilde{\zeta}_\beta=\sum_{\beta\in H}g(\beta)\neq\zeta_{\mathsf{in}}$$

and rejects. Otherwise, there exists some $\beta^*\in H$ such that $g(\beta^*)\neq\widetilde{\zeta}_{\beta^*}$. Thus, the correct query for the equation-merging oracle $(Q,R)^g_{|H|\mathsf{eqs}\mapsto 1\mathsf{eq}}$ is robust (that is, $(\beta,\widetilde{\zeta}_\beta)_{\beta\in H}\in\mathrm{Dom}(R^g_{|H|\mathsf{eqs}\mapsto 1\mathsf{eq}})$, and if the guide submits any other query, the searcher will reject during the query-check. Soundness then follows from the functionality of the equation-merging robust relation $R_{k\mathsf{eqs}\mapsto 1\mathsf{eq}}$. □

4.6 Interactive search system for equation-merging $(Q,R)_{k\mathsf{eqs}\mapsto 1\mathsf{eq}}$

We present a plain (i.e., non-oracle-aided) interactive search system for the equation-merging problem $(Q,R)_{k\mathsf{eqs}\mapsto 1\mathsf{eq}}$, defined in (14). The system is based on techniques implicitly used in [LFKN92] (for univariate polynomials) and explicitly in [ALM+98] and [LS91] (for multivariate polynomials).

As before, we present the equation-merging system $\Pi^g_{k\mathsf{eqs}\mapsto 1\mathsf{eq}}$ for some fixed polynomial $g : \mathbb{F}^\ell \to \mathbb{F}$, and the construction can be easily generalized for a polynomial family.

Remark 4.17. *The protocol for the equation-merging problem $(Q, R)^g_{k\mathsf{eqs}\mapsto 1\mathsf{eq}}$ requires the searcher to know the individual degree of the polynomial g in question, as well as its arity, and we assume that these parameters are known to the searcher. In addition, letting k denote the number of submitted equations to the system, we abuse notation and treat $[k]$ as a subset of $\mathbb{F}$ (where we can assume without loss of generality that $k \leq |\mathbb{F}|$, and therefore, there exists an embedding $[k] \mapsto \mathbb{F}$).*

4.6.1 The protocol

Construction 4.18 (IS for $(Q, R)^g_{k\mathsf{eqs}\mapsto 1\mathsf{eq}}$)**.** *Let $g : \mathbb{F}^\ell \to \mathbb{F}$ be a polynomial of individual degree Δ_g. Given an input $((\alpha_1, \zeta_1), \ldots, (\alpha_k, \zeta_k)) \in (\mathbb{F}^\ell \times \mathbb{F})^k$ for some $k \leq |\mathbb{F}|$, the parties proceed as follows.*

- *The guide computes the unique polynomial mapping $\mathfrak{m} : \mathbb{F} \to \mathbb{F}^\ell$ of degree at most $(k-1)$ that interpolates on $\alpha_1, \ldots, \alpha_k$ (that is, $\mathfrak{m}(i) = \alpha_i$ for each $i \in [k]$). In addition, the guide computes the unique univariate polynomial $h : \mathbb{F} \to \mathbb{F}$ of degree at most $\Delta_g \cdot \ell \cdot (k-1)$ defined by*

$$h(X) := g(\mathfrak{m}(X)), \tag{17}$$

  and sends h to the searcher.

- *Upon receiving a polynomial $\widetilde{h}$ (where allegedly, $\widetilde{h} = h$), the searcher checks that $\widetilde{h}$ is a univariate polynomial of degree at most $\Delta_g \cdot \ell \cdot (k-1)$ and that $\widetilde{h}(i) = \zeta_i$ for each $i \in [k]$ (and otherwise rejects).*

  The searcher computes the polynomial mapping $\mathfrak{m} : \mathbb{F} \to \mathbb{F}^\ell$ defined above, selects a uniformly random element $\alpha \in \mathbb{F}$, and outputs $(\mathfrak{m}(\alpha), \widetilde{h}(\alpha))$.

The properties of Construction 4.18, followed by a proof, are given below.

Proposition 4.19. *The system $\Pi^g_{k\mathsf{eqs}\mapsto 1\mathsf{eq}}$ of Construction 4.18 is a perfectly complete and $\left(1 - \frac{\Delta_g \cdot \ell \cdot (k-1)}{|\mathbb{F}|}\right)$-sound interactive search system for the equation-merging problem $(Q, R)^g_{k\mathsf{eqs}\mapsto 1\mathsf{eq}}$. Moreover, the prescribed (honest) guide runs in time $\mathrm{poly}(\ell, k, \Delta_g, \log|\mathbb{F}|)$ and the searcher runs in time $\Delta_g \cdot k \cdot \mathrm{poly}(\ell, \log|\mathbb{F}|)$. The protocol is public coin and has a single communication round.*

Proof. The complexity features of Construction 4.18 easily follow from the construction.

For completeness, let $((\alpha_1, \zeta_1), \ldots, (\alpha_k, \zeta_k)) \in \mathrm{Dom}(Q^g_{k\mathsf{eqs}\mapsto 1\mathsf{eq}})$ be a qualified input and let h be the polynomial sent by the guide. Then

$$h(i) = g(\mathfrak{m}(i)) = g(\alpha_i) = \zeta_i$$

for each $i \in [k]$, implying that the searcher does not reject during his checks. Completeness then follows by $g(\mathfrak{m}(\alpha)) = h(\alpha)$.

For soundness, let $((\alpha_1, \zeta_1), \ldots, (\alpha_k, \zeta_k)) \in \mathrm{Dom}(R^g_{k\mathsf{eqs}\mapsto 1\mathsf{eq}})$ be a robust input , and fix some malicious guide strategy. If the sent polynomial $\widetilde{h}$ is not a univariate polynomial of degree at most

$\Delta_g \cdot \ell \cdot (k-1)$, or if there exists an index $i \in [k]$ such that $\widetilde{h}(i) \neq \zeta_i$, then the searcher rejects and we are done. Otherwise, it must be the case that $\widetilde{h} \neq h$. This is since, by robustness, there exists some index $\tilde{\imath} \in [k]$ such that $g(\alpha_{\tilde{\imath}}) \neq \zeta_{\tilde{\imath}}$, and therefore

$$h(\tilde{\imath}) = g(\mathfrak{m}(\tilde{\imath})) = g(\alpha_{\tilde{\imath}}) \neq \zeta_{\tilde{\imath}} = \widetilde{h}(\alpha_{\tilde{\imath}}).$$

Therefore, $\widetilde{h}$ may agree with the correct polynomial h on at most $\Delta_g \cdot \ell \cdot (k-1)$ elements. It follows that the probability that the searcher fails (i.e., outputs $(\mathfrak{m}(\alpha), \widetilde{h}(\alpha))$ for which $g(\mathfrak{m}(\alpha)) = \widetilde{h}(\alpha)$) is at most $\frac{\Delta_g \cdot \ell \cdot (k-1)}{|\mathbb{F}|}$. □

4.7 Interactive search system for Φ-evaluation $(Q,R)^{\Phi}_{\mathsf{eval}}$

This section presents an interactive search system for the Φ-evaluation problem $(Q,R)^{\Phi}_{\mathsf{eval}}$, where recall that $\Phi = \{\widehat{\phi}^n_i\}_{n,i}$ is the polynomial family describing the structure of bounded-uniform circuits (see Section 4.1.2 and (9)).[36] The protocol is derived from [RRR16].

Proposition 4.20 (IS for $(Q,R)^{\Phi}_{\mathsf{eval}}$)**.** *Let $\{C_n\}_n$ be a $\mathsf{T} := \mathsf{T}(n)$-time and $\mathsf{S} := \mathsf{S}(n)$-space bounded-uniform circuit family of size $s := s(n)$ and depth $d := d(n)$, where $n \leq \mathsf{T} \leq \exp(n)$ and $\log(\mathsf{T}) \leq \mathsf{S} \leq \mathrm{poly}(n)$. Fix $\mu \in (0, \frac{1}{2})$ a small enough constant, and let Φ be the polynomial family describing the structure of $\{C_n\}_n$ (see Observation 4.3). Then, the Φ-evaluation problem $(Q,R)^{\Phi}_{\mathsf{eval}}$ has a perfectly complete and $\left(1 - \frac{1}{4d}\right)$-sound[37] interactive search system with the following properties. The prescribed (honest) guide strategy runs in time $(\mathsf{T} + \mathrm{poly}(s, \log|\mathbb{F}|))^{1+O(\mu)} \cdot \mathrm{poly}(\mathsf{S}, \log|\mathbb{F}|)$. The searcher runs in time $O(n + |H|) \cdot \mathrm{polylog}(\mathsf{T} + \log|\mathbb{F}|) + (\mathsf{T} + \mathrm{poly}(s, \log|\mathbb{F}|))^{O(\mu)} \cdot \mathrm{poly}(\mathsf{S}, \log|\mathbb{F}|)$. Moreover, the protocol is public coin and has a constant number of rounds.*

Proposition 4.20 is derived from [RRR16]'s interactive proof system for space- and time-bounded computations (given in Theorem 4.21), when reducing its soundness error from $\frac{1}{2}$ to $\frac{1}{\Omega(d)}$ by parallel repetitions.

Theorem 4.21 ([RRR16, Corollary 8])**.** *Let $\mu \in (0, \frac{1}{2})$ and let $\mathsf{T} := \mathsf{T}(n)$, $\mathsf{S} := \mathsf{S}(n)$ be integer functions such that $n \leq \mathsf{T} \leq \exp(n)$ and $\mathrm{poly}(\frac{1}{\mu}) \leq \log(\mathsf{T}) \leq \mathsf{S} \leq \mathrm{poly}(n)$. Let L be a set decidable by a T-time and S-space Turing machine. Then L has a perfectly complete and $\frac{1}{2}$-sound interactive proof system. The prescribed (honest) guide runs in time $\mathsf{T}^{1+O(\mu)} \cdot \mathrm{poly}(\mathsf{S})$ and the searcher runs in time $n \cdot \mathrm{polylog}(\mathsf{T}) + \mathsf{T}^{O(\mu)} \cdot \mathrm{poly}(\mathsf{S})$. Moreover, the protocol is public coin and has round complexity of $\mu^{-O\left(\frac{1}{\mu}\right)}$.*

4.8 Interactive search system for check-eval $(Q,R)^{V}_{\mathsf{check\text{-}eval}}$

Recall that the check-eval problem $(Q,R)^{V}_{\mathsf{check\text{-}eval}}$ for the polynomial family $V_d = \{\widehat{v}^x_d\}_x$ (where $\widehat{v}^x_d$ corresponds to the lowest layer of the computation $C_{|x|}(x)$) is defined by

$$(Q,R)^{V}_{\mathsf{check\text{-}eval}} = \{(\langle x\rangle, (\alpha_{\mathsf{in}}, \zeta_{\mathsf{in}}), b) : b = 1 \iff \widehat{v}^x_d(\alpha_{\mathsf{in}}) = \zeta_{\mathsf{in}}\},$$

[36] This is where the uniformity of the class $\mathcal{UC}irc$ is being used.

[37] We set a soundness error of $\frac{1}{4d}$ because this is what required later on. This can be improved by parallel repetition.

where the qualified relation $Q^{V_d}_{\mathsf{check\text{-}eval}}$ consists of all correct evaluations $(\langle x\rangle, (\alpha_{\mathsf{in}}, \widehat{v}^x_d(\alpha_{\mathsf{in}})), 1)$, and $R^{V_d}_{\mathsf{check\text{-}eval}}$ consists of the incorrect evaluations in the natural way. Further recall that $\widehat{v}^x_d : \mathbb{F}^\ell \to \mathbb{F}$ is the low-degree extension of the function $v^x_d = x0^{s-n}$ with respect to a field $\mathbb{F}$, subset $H \subseteq \mathbb{F}$, and integer $\ell = O(\log_{|H|} s)$ (see Section 4.2).

The searcher can solve the check-eval problem $(Q, R)^V_{\mathsf{check\text{-}eval}}$ directly by evaluating $\widehat{v}^x_d$ by himself, since unlike the intermediate layers $v^x_2, \ldots, v^x_{d-1}$ of the computation $C_n(x)$, the lowest layer v^x_d is determined entirely by the input x. However, a direct application of the evaluation algorithm from Section 4.1.1 would require time $|H|^{\ell+1} \cdot \mathrm{polylog}\,|\mathbb{F}|$, exceeding the $O(s)$ time needed to compute $C_n(x)$ (hence making the entire interactive search system for $\mathcal{UC}irc(s, d, \mathsf{T}, \mathsf{S})$ redundant (when $\mathsf{T} = O(s)$)).

This can be overcome in the current case as follows. Note that the string $v^x_d = x0^{s-n}$ is sparse, as only the first n entries contain actual data. Hence, letting $\ell' = \lceil \log_{|H|} n \rceil$, we view v^x_d as a function from $H^{\ell'} \times \{1\}^{\ell-\ell'}$ to $\{0,1\}$ defined by $v^x_d(\beta' \| 1^{\ell-\ell'}) = x_{\beta'}$ for any $\beta' \in H^{\ell'}$ and otherwise 0. Therefore, for every $\alpha \in \mathbb{F}^\ell$, evaluating $\widehat{v}^x_d$ over α amounts to computing

$$\widehat{v}^x_d(\alpha) = \sum_{\beta' \in H^{\ell'}} \mathsf{EQ}(\alpha, \beta' \| 1^{\ell-\ell'}) \cdot v^x_d(\beta' \| 1^{\ell-\ell'}),$$

which contains only $|H|^{\ell'} = \Theta(n)$ summands. Consequently, $\widehat{v}^x_d$ can be evaluated in time $n \cdot \ell \cdot |H| \cdot \mathrm{polylog}\,|\mathbb{F}|$, which is affordable for the searcher.

We therefore obtain a (fictitious) "interactive" search system for the check-evolution problem $(Q, R)^V_{\mathsf{check\text{-}eval}}$ by having the searcher evaluate $\widehat{v}^x_d$ on the given input element α_{in} by himself, and accepting if and only if the given value ζ_{in} matches his computation. The properties of the search system are summarized below for future reference.

Proposition 4.22. *The check-eval problem $(Q, R)^V_{\mathsf{check\text{-}eval}}$ has an ideal interactive search system with no interaction, where the searcher runs in time $n \cdot \ell \cdot |H| \cdot \mathrm{polylog}\,|\mathbb{F}|$.*

4.9 Concluding an IP for bounded-depth circuits

Finally, we obtain a (plain) interactive proof system for the class $\mathcal{UC}irc(s, d, \mathsf{T}, \mathsf{S})$ by composing the protocols from the previous sections. This completes the proof of Theorem 4.1.

For convenience, Figure 2 provides a roadmap of the construction presented in this section.

Proof of Theorem 4.1. Fix a set $L \in \mathcal{UC}irc(s, d, \mathsf{T}, \mathsf{S})$ and let $\{C_n\}_n$ be the circuit family deciding L. In addition, fix a field $\mathbb{F}$, a subset $H \subseteq \mathbb{F}$, and integer ℓ such that $|H| = O(d + \log s)$, $\ell = \lceil \log_{|H|} s \rceil$, and $|\mathbb{F}| = \mathrm{poly}(|H|)$ large enough. A (plain) interactive proof system for L as in Theorem 4.1 is derived by iteratively composing (bottom-up) the constructions given above, using Theorem 2.8, while using the fixed field $\mathbb{F}$, subset H and integer ℓ as above when required by the inner problems. □

Remark 4.23. *Note that the interactive proof system of Theorem 4.1 can easily be generalized to an interactive search system for functions $\Psi : \{0,1\}^n \to \{0,1\}^{m(n)}$, for any integer $m \geq 1$, computable by bounded-depth circuit families as follows. Given an input x, the guide sends the searcher a candidate solution $\widetilde{y}$ for x (where allegedly $\widetilde{y} = \Psi(x)$). The parties then carry the interactive proof system of Theorem 4.1 for the set $L := \{(x, \Psi(x)) : x \in \{0,1\}^n\}$.*

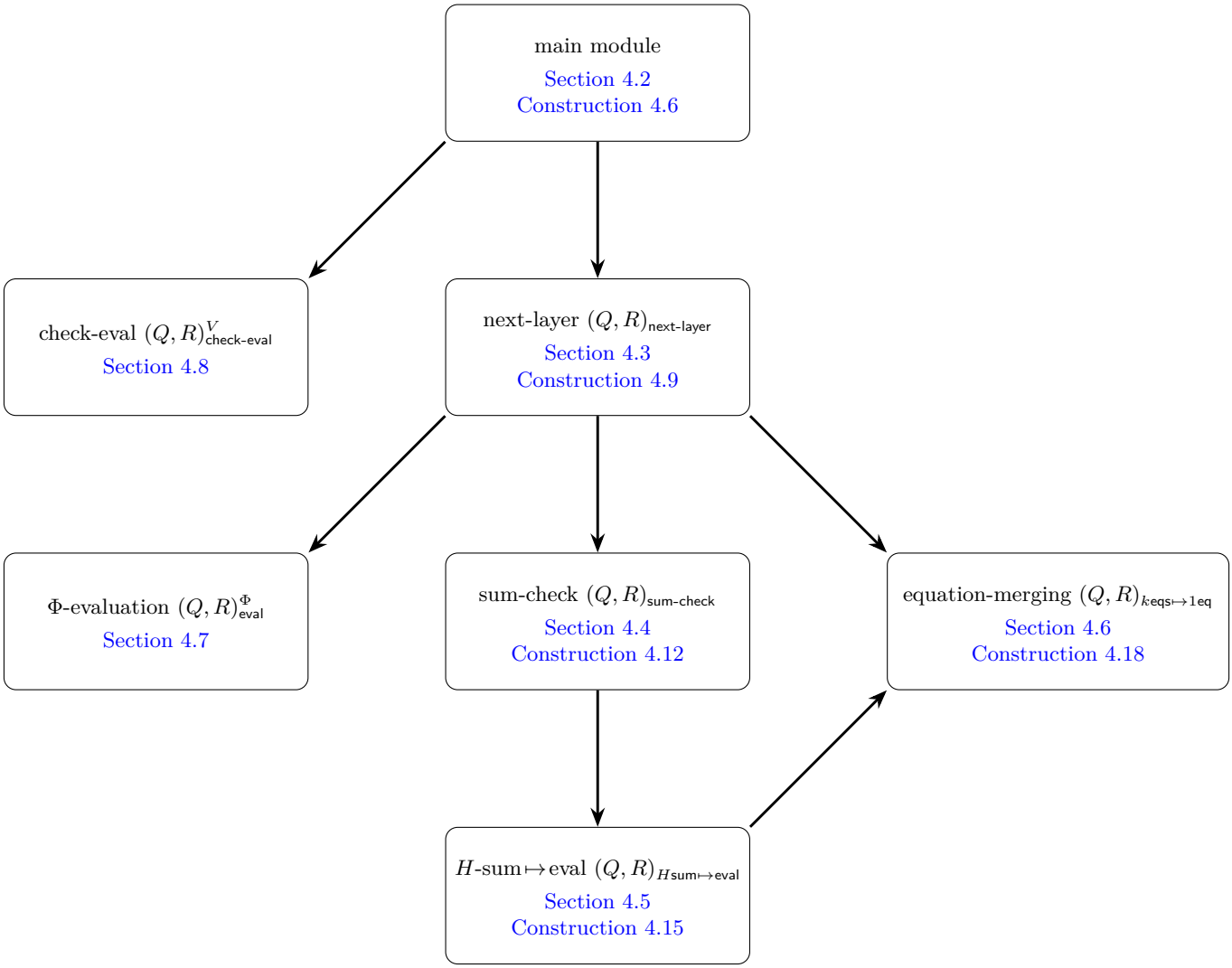

Figure 2: The modular interactive proof construction for $\mathcal{UC}irc(s, d, \mathsf{T}, \mathsf{S})$. Each node represents a D-search problem that appears in the construction, labeled with the construction or section in which its (oracle-aided) interactive search system is given, along with the properties of the constitution. A directed edge from node A to node B indicates that the system for A uses oracle access to B. Nodes with no outgoing edges, therefore, correspond to plain (non-oracle-aided) systems.

5 An SMR system for bounded-depth circuit classes

In this section, we extend the construction of Section 4 into an SMR system for sets decidable by uniform-bounded depth circuits of bounded size and depth (the class $\mathcal{UC}irc(s,d,\mathsf{T},\mathsf{S})$) using the tools developed in the previous sections. Specifically, our modular approach for constructing interactive search systems.[38] Formally, we prove the following theorem.

Theorem 5.1. *Let $s := s(n)$, $d := d(n)$, $\mathsf{T} := \mathsf{T}(n)$ and $\mathsf{S} := \mathsf{S}(n)$ integer functions such that $n \leq \mathsf{T} \leq \exp(n)$ and $\log(\mathsf{T}) \leq \mathsf{S} \leq \mathrm{poly}(n)$, and let a constant $\mu \in (0, \frac{1}{2})$. Every set $L \in \mathcal{UC}irc(s,d,\mathsf{T},\mathsf{S})$ has a perfectly complete and $\frac{1}{2}$-sound[39] SMR-IS system with the following properties. The (honest) guide runs in time $O(\mathsf{T})^{1+O(\mu)} \cdot \mathrm{poly}(s,\mathsf{S})$. The searcher runs in time $n \cdot \big(\mathrm{poly}(d,\log \mathsf{T}) + (\mathsf{T}+\mathrm{poly}(s))^{O(\mu)} \cdot \mathrm{poly}(d,\log s,\mathsf{S})\big)$. The system is public coin and has $d \cdot \mathrm{polylog}(s)$ rounds.*

As a corollary, we obtain a *doubly-efficient* SMR-IS for sets decidable by $\mathrm{poly}(n)$-time and $\mathrm{polylog}(n)$-space uniform circuits of size $\mathrm{poly}(n)$ and depth $\mathrm{polylog}(n)$.

Corollary 5.2. *Let $\mathsf{T} = \mathrm{poly}(n)$, $\mathsf{S} = \mathrm{polylog}(n)$, $s = \mathrm{poly}(n)$ and $d = \mathrm{polylog}(n)$ such that $\log(\mathsf{T}) \leq \mathsf{S}$. Then each set $L \in \mathcal{UC}irc(s,d,\mathsf{T},\mathsf{S})$ has a perfectly complete and $\frac{1}{2}$-sound* doubly-efficient *SMR-IS system in which the (honest) guide runs in time $\mathrm{poly}(n)$, and the searcher runs in time $\widetilde{O}(n)$. The system is public coin and has $\mathrm{polylog}(n)$ rounds.*

For the rest of this section, fix a set $L \in \mathcal{UC}irc(s,d,\mathsf{T},\mathsf{S})$ as in Theorem 5.1 and let $\{C_n\}_n$ be a circuit family that decides L ala Theorem 5.1. For a finite field $\mathbb{F}$, a subset $H \subseteq \mathbb{F}$ and an integer $\ell = O(\log_{|H|} s)$, let $V = \{\widehat{v}_i^x : \mathbb{F}^\ell \to \mathbb{F}\}_{x,i}$ be the polynomial family of degree at most $|H|-1$ that describes computations in $\{C_n\}_n$ (see Section 4.1.2).

5.1 Construction overview

We construct an SMR-IS for deciding L by exploiting the modular approach developed in earlier sections: we "decompose" the problem of deciding L into simpler subtasks (as in Section 4) for which constructing an SMR-IS is easier. We then construct an SMR-IS for each subtask and use the SMR-IS composition theorem (Theorem 3.4) to compose the SMR-ISs (for these subtasks) together, yielding a (plain) SMR-IS for L. The SMR-ISs for the subtasks are not constructed from scratch, but rather obtained by transforming their IS-analogs from Section 4 into SMR-ISs via the IS$\mapsto$SMR-IS transformations Constructions 3.9 and 3.12.

The construction proceeds in two main steps. The first step constructs an "outer" *oracle-aided* SMR-IS for L that uses oracle access to the next-layer problem $(Q,R)^V_{\mathsf{next\text{-}layer}}$ and the check-eval problem $(Q,R)^V_{\mathsf{check\text{-}eval}}$ (see Figure 2 for a roadmap of the IS contraction given in Section 4.2). This reduces the task of constructing a (plain) SMR-IS for L to constructing an SMR-IS for each of the D-search problems that the resulting outer system uses as subroutines. Obtaining an SMR-IS for

[38] A PCIS (probabilistically-checkable interactive search) systems for $\mathcal{UC}irc(s,d,\mathsf{T},\mathsf{S})$ can be easily derived from Theorem 5.1 using standard methods.

[39] The constant $\frac{1}{2}$-soundness error is arbitrary. One can achieve an arbitrarily small constant (and even $\mathrm{poly}(d,\log \mathsf{T})^{-1}$-soundness error) without deteriorating other parameters.

all but one subroutine (and their nested subroutines) is straightforward, since these subroutines have constant-round interactive search systems (already given in Section 4), which can be directly used as SMR-ISs with $O(1)$ reading complexity (by simply reading the entire interaction).

Specifically, a plain SMR for the check-eval problem $(Q,R)^V_{\mathsf{check\text{-}eval}}$ is trivial (requiring no communication), and a constant-round oracle-aided SMR for the next-layer problem $(Q,R)^V_{\mathsf{next\text{-}layer}}$ is obtained directly from the constant-round IS $\Pi^V_{\mathsf{next\text{-}layer}}$ of Construction 4.9. Recall that $\Pi^V_{\mathsf{next\text{-}layer}}$ makes a single oracle call to each of the following D-search problems: sum-check problem $(Q,R)_{\mathsf{sum\text{-}check}}$, the equation-merging problem $(Q,R)_{k\mathsf{eqs}\mapsto 1\mathsf{eq}}$, and the Φ-evaluation problem $(Q,R)^{\Phi}_{\mathsf{eval}}$. The only forgoing D-search problem that lacks a trivial SMR-IS is the sum-check problem $(Q,R)_{\mathsf{sum\text{-}check}}$, and constructing an SMR-IS for $(Q,R)_{\mathsf{sum\text{-}check}}$ constitutes the second main step of the construction.

Recall that [ACY22b] already constructed an SMR for sum-check; nevertheless, rather than using their construction as a black box, we carry out this second step ourselves for two reasons. First, the sum-check SMR of [ACY22b] has constant soundness error, whereas our application requires soundness error at most $O(d)^{-1}$. Second, we carry out this explicit construction in order to demonstrate our methodology for deriving SMR systems via composition.

On using the equation-merging problem $(Q,R)_{k\mathsf{eqs}\mapsto 1\mathsf{eq}}$ as a merging mechanism. As mentioned above, constructing SMR-IS systems for the different subroutines is done via the IS$\mapsto$SMR-IS transformations Constructions 3.9 and 3.12. Recall that both transformations require certain solution-merging mechanisms as an ingredient. Since the different subroutines extensively use polynomials and claims about their evaluations, the equation-merging problem $(Q,R)_{k\mathsf{eqs}\mapsto 1\mathsf{eq}}$ (defined in (7)) serves as a natural merging mechanism in many of the SMR-ISs constructed in this section. We recall the functionality of $(Q,R)_{k\mathsf{eqs}\mapsto 1\mathsf{eq}}$: for a fixed function f, the equation-merging problem $(Q,R)^f_{k\mathsf{eqs}\mapsto 1\mathsf{eq}}$ receives multiple claims of the form $f(\alpha)\stackrel{?}{=}\zeta$, and outputs a pair $(\alpha_{\mathsf{out}},\zeta_{\mathsf{out}})$ such that $f(\alpha_{\mathsf{out}})=\zeta_{\mathsf{out}}$ if and only if equality holds in each input claim.

5.2 Oracle-aided SMR-IS for L

We begin by constructing an "outer" *oracle-aided* SMR-IS for the set L. To this end, let Π^L_{ucirc} denote the main oracle-aided interactive proof system for L given in Construction 4.6. We use the *efficient*[40] IS$\mapsto$SMR-IS transformation (Construction 3.12) to transform Π^L_{ucirc} into an SMR-IS as follows. First, note that the main system Π^L_{ucirc} can be easily modified to be *pure* ala Definition 3.11.[41] Indeed, recall that given an input x, the main system Π^L_{ucirc} proceeds by sequentially using the next-layer problem

[40] The efficient transformation is required because the main system Π^L_{ucirc} has d oracle-calling steps, which may exceed $\frac{1}{2}\log n + o(\log n)$. Note that applying the basic IS$\mapsto$SMR-IS transformation for larger values of d, even $d=O(\log n)$, which is seemingly feasible by inducing a polynomial-time searcher, renders the construction superfluous: Theorem 5.1 is only meaningful for sets decidable by poly(n)-time-bounded uniform circuits, since larger running times result in a super-polynomial-time searcher. When $d=\Omega(\log n)$, however, the searcher's work in the SMR-IS resulting from the basic transformation already requires polynomial time, at which point the searcher could simply evaluate $C_n(x)$ directly at no greater cost. (We comment that considering $d=c\log n$ for some constant $c>\frac{1}{2}$ may still be meaningful in the fine-grained complexity setting, if evaluating $C_n(x)$ requires time greater than 2^{2d} (but still polynomial). The basic transformation is therefore meaningful (outside the fine-grained setting) only when $d=\frac{1}{2}\log n+o(\log n)$.

[41] The required modification is merely technical, and amounts to adjusting the input and outputs of the different oracles to adhere to the requirements of Definition 3.11 (for example, set the oracle answers to contain the context for the following oracle call).

$(Q,R)^{\widehat{v}_i^x,\widehat{v}_{i+1}^x}_{\mathsf{next\text{-}layer}}$ (see (5)) for $i=0,\ldots,d-1$, followed by a single oracle call to the check-eval problem $(Q,R)^{\widehat{v}_d^x}_{\mathsf{check\text{-}eval}}$, where at each iteration it feeds the answer received from the last oracle call (along with the corresponding context) as a query for the following oracle call. Second, using the efficient IS $\mapsto$ SMR-IS transformation requires merging mechanisms for the next-layer problem $(Q,R)^{\widehat{v}_i^x,\widehat{v}_{i+1}^x}_{\mathsf{next\text{-}layer}}$, and the check-eval problem $(Q,R)^{\widehat{v}_d^x}_{\mathsf{check\text{-}eval}}$.

Note that for each $i\in\{0,\ldots,d-1\}$, the equation-merging problem $(Q,R)^{\widehat{v}_{i+1}^x}_{k\mathsf{eqs}\mapsto 1\mathsf{eq}}$ (used for the polynomial $\widehat{v}_{i+1}^x$) serves as a natural merging mechanism for the next-layer problem $(Q,R)^{\widehat{v}_i^x,\widehat{v}_{i+1}^x}_{\mathsf{next\text{-}layer}}$. A merging mechanism for the check-eval problem $(Q,R)^{\widehat{v}_d^x}_{\mathsf{check\text{-}eval}}$ is trivially obtained since $(Q,R)^{\widehat{v}_d^x}_{\mathsf{check\text{-}eval}}$ is a decision problem (hence, merging amounts to outputting 1 if and only if all the given solutions are 1).

By Proposition 3.13, applying the efficient IS $\mapsto$ SMR-IS transformation to the main system Π^L_{ucirc} (and using the above merging mechanisms) for $\ell_{\mathsf{SMR}}=3d$ yields:

Proposition 5.3. *Any set $L\in\mathcal{UC}irc(s,d,\mathsf{T},\mathsf{S})$ has a perfectly complete and $\frac{2}{3}$-sound[42] oracle-aided SMR-IS system with reading complexity $O(1)$.[43] The system has $O(d)$ oracle-calling steps dedicated to advancing the computation, each consisting of $O(d)$ parallel oracle calls to the next-layer problem $(Q,R)^V_{\mathsf{next\text{-}layer}}$ along with a single oracle call to the check-eval problem $(Q,R)^V_{\mathsf{check\text{-}eval}}$. There are an additional $O(d)$ oracle-calling steps dedicated to solution-merging, each consisting of $O(d)$ parallel oracle calls to the equation-merging problem $(Q,R)_{k\mathsf{eqs}\mapsto 1\mathsf{eq}}$ along with a single oracle call to the trivial merging mechanism for $(Q,R)^V_{\mathsf{check\text{-}eval}}$.*

Moreover, the guide runs in time $O(\mathsf{T})+\mathrm{poly}(d)\cdot O(s+\ell\log|\mathbb{F}|)$ and the searcher runs in time $\mathrm{poly}(d)\cdot O(\ell\log|\mathbb{F}|)$. The system is public coin and has $O(d)$ rounds.

5.3 Oracle-aided SMR-IS for sum-check $(Q,R)_{\mathsf{sum\text{-}check}}$

We now present an oracle-aided SMR-IS for the sum-check problem $(Q,R)_{\mathsf{sum\text{-}check}}$, defined in (6). Fix a polynomial $f:\mathbb{F}^\ell\to\mathbb{F}$ for some finite field $\mathbb{F}$, and let Δ_f denote the individual degree of f. In addition, fix a subset $H\subseteq\mathbb{F}$, and let $\Pi^f_{\mathsf{sum\text{-}check}}$ denote the oracle-aided interactive search system for the sum-check problem $(Q,R)^f_{\mathsf{sum\text{-}check}}$, given in Construction 4.12.

We again use the *efficient*[44] IS $\mapsto$ SMR-IS transformation (Construction 3.12) to transform the sum-check system $\Pi^f_{\mathsf{sum\text{-}check}}$ into SMR-IS. This can be done since the sum-check system $\Pi^f_{\mathsf{sum\text{-}check}}$ is pure,[45] proceeding by sequentially using the H-sum $\mapsto$ eval problem $(Q,R)^{\mathcal{G}}_{H\mathsf{sum}\mapsto\mathsf{eval}}$ (14) for ℓ rounds, referring to the polynomial family $\mathcal{G}:=\mathcal{G}(f)$ reminded below (18). In addition, the H-sum $\mapsto$ eval problem $(Q,R)^{\mathcal{G}}_{H\mathsf{sum}\mapsto\mathsf{eval}}$ admits a merging mechanism, given again by the equation-merging prob-

[42] The constant $\frac{2}{3}$ is chosen arbitrarily over $\frac{1}{2}$. The soundness error can be reduced to be arbitrarily small (but noticeable) by increasing the number of epochs in the efficient IS $\mapsto$ SMR-IS transformation.

[43] Recall that the reading complexity of an oracle-aided SMR-IS is the number of oracle-calling steps and guide messages that the searcher reads.

[44] On top of the reason mention in Footnote 40, the efficient transformation is required here (when considering the sum-check system in the context of Proposition 5.3) for another reason. To conclude an SMR-IS for L, we must obtain an $(1-O(d)^{-1})$-sound SMR-IS for the sum-check problem $(Q,R)_{\mathsf{sum\text{-}check}}$ (which the basic transformation cannot achieve for circuits of superconstant size). This is because, while realizing the main oracle-aided SMR-IS for L of Proposition 5.3 (that is, implementing all of its oracle calling steps with actual protocol), the resulting system performs d parallel oracle calls to the sum-check problem. Hence, the sum-check SMR-IS used to implement these oracles must have soundness error $O(d)^{-1}$ to "absorb" a multiplicative factor of d.

[45] Up to naive modifications as in Footnote 41.

lem $(Q,R)_{k\mathsf{eqs}\mapsto 1\mathsf{eq}}$. However, some care is required: as noted above, the equation-merging problem $(Q,R)_{k\mathsf{eqs}\mapsto 1\mathsf{eq}}$ merges equations all referring to the *same fixed polynomial*. However, in the context of the sum-check system, the polynomial considered by $(Q,R)_{H\mathsf{sum}\mapsto\mathsf{eval}}$ depends on the partial transcript obtained so far, hence (seemingly) forces us to merge different polynomials. Indeed, recall that the polynomial used at iteration i of the sum-check system is determined by $i-1$ elements $\alpha_1,\ldots,\alpha_{i-1}\in\mathbb{F}$ obtained in the previous iterations, and that different elements induce different polynomials in $\mathcal{G}$.

This can be overcome in the current case, since all polynomials that may appear at the i^{th} iteration of $\Pi^f_{\mathsf{sum\text{-}check}}$ are derived from a *single* polynomial. Specifically, recall that the polynomial family $\mathcal{G}$ is defined by

$$\mathcal{G}=\left\{g_{\alpha_1,\ldots,\alpha_i}(X):=\sum_{\beta_{i+2},\ldots,\beta_\ell\in H} f(\alpha_1,\ldots,\alpha_i,X,\beta_{i+2},\ldots,\beta_\ell)\right\}_{i\in\{0,\ldots,\ell-1\},\,\alpha_1,\ldots,\alpha_i\in\mathbb{F}}. \tag{18}$$

Crucially, for each $i\in\{0,\ldots,\ell-1\}$, the $i+1^{\text{st}}$ iteration of $\Pi^f_{\mathsf{sum\text{-}check}}$ always invokes the H-sum$\mapsto$eval oracle $(Q,R)^{\mathcal{G}}_{H\mathsf{sum}\mapsto\mathsf{eval}}$ with a polynomial $g_{\alpha_1,\ldots,\alpha_i}\in\mathcal{G}$ having *exactly* i fixed elements $\alpha_1,\ldots,\alpha_i\in\mathbb{F}$. Moreover, the subset of all such polynomials (that may be used at the $i+1^{\text{th}}$ iteration of the sum-check system) can be represented by a *single* $(i+1)$-variate polynomial $g^*_i:\mathbb{F}^{i+1}\to\mathbb{F}$, defined by

$$g^*_i(X_1,\ldots,X_i,X):=\sum_{\beta_{i+2},\ldots,\beta_\ell\in H} f(X_1,\ldots,X_i,X,\beta_{i+2},\ldots,\beta_\ell). \tag{19}$$

We therefore generalize the H-sum$\mapsto$eval problem to handle multivariate polynomials as follows. For a polynomial $g^*_i:\mathbb{F}^{i+1}\to\mathbb{F}$, the generalized H-sum$\mapsto$eval problem $(Q,R)^{g^*_i}_{H\mathsf{sum}\mapsto\mathsf{eval}}$ receives as input a context $\overline{\alpha}^i_{\mathsf{in}}\in\mathbb{F}^i$ along with a value $\zeta_{\mathsf{in}}\in\mathbb{F}$, and outputs $(\overline{\alpha}^i_{\mathsf{out}},\alpha_{\mathsf{out}},\zeta_{\mathsf{out}})\in\mathbb{F}^i\times\mathbb{F}\times\mathbb{F}$ such that

$$g^*_i(\overline{\alpha}^i_{\mathsf{out}},\alpha_{\mathsf{out}})=\zeta_{\mathsf{out}}\iff\sum_{\beta\in H}g^*_i(\overline{\alpha}^i_{\mathsf{in}},\beta)=\zeta_{\mathsf{in}}.$$

We then modify the sum-check system $\Pi^f_{\mathsf{sum\text{-}check}}$ to use this generalized H-sum$\mapsto$eval problem, which introduces no operational change. Applying the efficient IS$\mapsto$SMR-IS transformation to this modified system requires a merging mechanism for $(Q,R)^{g^*_i}_{H\mathsf{sum}\mapsto\mathsf{eval}}$ for each $i\in\{0,\ldots,\ell-1\}$, which is naturally provided by the equation-merging problem $(Q,R)^{g^*_i}_{k\mathsf{eqs}\mapsto 1\mathsf{eq}}$.

Following the lines described above yields an oracle-aided SMR-IS for the sum-check problem $(Q,R)^f_{\mathsf{sum\text{-}check}}$, which by the properties of the efficient IS$\mapsto$SMR-IS transformation used for $\ell_{\mathsf{SMR}}=\mathrm{poly}(\ell)$ epochs (see Proposition 3.13), and the properties of the sum-check system $\Pi^f_{\mathsf{sum\text{-}check}}$ (Propositions 4.13 and 4.14), has the following properties:

Proposition 5.4. *Let a polynomial $f:\mathbb{F}^\ell\to\mathbb{F}$ over a finite field $\mathbb{F}$. In addition, fix a subset $H\subseteq\mathbb{F}$, an integer $\ell_{\mathsf{SMR}}=\mathrm{poly}(\ell)$, and let $\mathcal{G}^*:=\{g^*_i\}_i$ denote the family of polynomials defined in (19). The sum-check problem $(Q,R)^f_{\mathsf{sum\text{-}check}}$ has a perfectly complete and $\left(1-\frac{\ell}{\ell_{\mathsf{SMR}}}\right)$-sound oracle-aided SMR-IS with reading complexity $O(1)$.*

Moreover, the system has $O(\ell_{\mathsf{SMR}})$ oracle-calling steps (dedicated to "advancing the computation"), each consisting of $O(\ell)$ parallel oracle calls to the H-sum$\mapsto$eval problem $(Q,R)_{H\mathsf{sum}\mapsto\mathsf{eval}}$. There are an

additional $O(\ell_{\mathsf{SMR}})$ oracle-calling steps (dedicated to solution-merging), each consisting of $O(\ell)$ parallel oracle calls to the equation-merging problem $(Q,R)_{k\mathsf{eqs}\mapsto 1\mathsf{eq}}$.

As a corollary, we obtain a (plain) SMR-IS system (that is, with no oracle-calling steps) for the sum-check problem $(Q,R)^{f}_{\mathsf{sum\text{-}check}}$.

Corollary 5.5. *Let a polynomial $f:\mathbb{F}^{\ell}\to\mathbb{F}$ of individual degree at most Δ_f, and fix an integer $\ell_{\mathsf{SMR}}=\mathrm{poly}(\ell)$. The sum-check problem $(Q,R)^{f}_{\mathsf{sum\text{-}check}}$ has a perfectly complete and $\left(1-\frac{\ell}{\ell_{\mathsf{SMR}}}-\frac{(\Delta_f+|H|)\cdot\mathrm{poly}(\ell)}{|\mathbb{F}|}\right)$-sound (plain) SMR-IS system with reading complexity $O(1)$.*

Moreover, the guide runs in time $|H|\cdot\mathrm{poly}(\Delta_f,\ell,\log|\mathbb{F}|)$ and the searcher runs in time $\Delta_f\cdot|H|\cdot\mathrm{poly}(\ell,\log|\mathbb{F}|)$. The system is public coin and has $O(\ell_{\mathsf{SMR}})$ rounds.

Proof of Corollary 5.5. Let $\Sigma^{f}_{\mathsf{sum\text{-}check}}$ denote the oracle-aided sum-check SMR of Proposition 5.4. The proof proceeds in two steps. We first obtain a plain SMR-IS for the H-sum $\mapsto$ eval problem: let $\Pi^{\mathcal{G}^*}_{H\mathsf{sum}\mapsto\mathsf{eval}}$ denote the oracle-aided IS for the H-sum $\mapsto$ eval problem, given in Construction 4.15. Composing the H-sum $\mapsto$ eval system $\Pi^{\mathcal{G}^*}_{H\mathsf{sum}\mapsto\mathsf{eval}}$ with the equation-merging (plain) system of Construction 4.18, via Theorem 3.4, yields a constant-round plain SMR for H-sum $\mapsto$ eval. Since the H-sum $\mapsto$ eval system $\Pi^{\mathcal{G}^*}_{H\mathsf{sum}\mapsto\mathsf{eval}}$ uses the equation-merging problem to merge $|H|$ equations in a polynomial from $\mathcal{G}^*$, the resulting H-sum $\mapsto$ eval SMR has perfect completeness and soundness error at most $\frac{\mathsf{deg}(f)\cdot(|H|-1)}{|\mathbb{F}|}$ (recalling that each polynomial in $\mathcal{G}^*$ has the same individual degree as f).

For the second step, we compose the sum-check SMR $\Sigma^{f}_{\mathsf{sum\text{-}check}}$ with the H-sum $\mapsto$ eval SMR from the first step, and with the equation-merging system of Construction 4.18, again via Theorem 3.4. The resulting plain SMR for sum-check has constant reading complexity and perfect completeness. Its soundness error is obtained by combining: the $\frac{\ell}{\ell_{\mathsf{SMR}}}$ soundness error of the sum-check SMR; the $\frac{\mathsf{deg}(f)\cdot\ell\cdot(|H|-1)}{|\mathbb{F}|}\cdot O(\ell_{\mathsf{SMR}})$ soundness error accumulated over $O(\ell_{\mathsf{SMR}})$ oracle-calling steps, each consisting of $O(\ell)$ parallel invocations of the H-sum $\mapsto$ eval SMR; and the $\frac{\mathsf{deg}(f)\cdot O(\ell^2)}{|\mathbb{F}|}\cdot O(\ell_{\mathsf{SMR}})$ soundness error accumulated over $O(\ell_{\mathsf{SMR}})$ oracle-calling steps, each consisting of $O(\ell)$ parallel invocations of the equation-merging system, where each invocation merges $O(\ell)$ equations in a polynomial from $\mathcal{G}^*$. □

5.4 Concluding an SMR-IS for bounded-depth circuits

We are now ready to prove Theorem 5.1, thereby obtaining a (plain) SMR-IS system for the class $\mathcal{UC}irc(s,d,\mathsf{T},\mathsf{S})$.

Proof of Theorem 5.1. Let a set $L\in\mathcal{UC}irc(s,d,\mathsf{T},\mathsf{S})$ as in Theorem 5.1. Fix a field $\mathbb{F}$, a subset $H\subseteq\mathbb{F}$ and an integer ℓ such that $|H|=O(d+\log s)$, $|\mathbb{F}|=\mathrm{poly}(|H|)$ sufficiently large, and set $\ell=O(\log_{|H|}s)$. The plain SMR-IS Σ_L for L is obtained in two steps. We first obtain a plain SMR-IS $\Sigma_{\mathsf{next\text{-}layer}}$ for the next-layer problem $(Q,R)^{V}_{\mathsf{next\text{-}layer}}$, by applying the composition theorem for SMR-ISs (Theorem 3.4) to the (constant round) oracle-aided IS for the next-layer problem of Construction 4.9, while implementing the oracle call for the sum-check problem $(Q,R)_{\mathsf{sum\text{-}check}}$ with the sum-check SMR-IS of Corollary 5.5 using $\ell_{\mathsf{SMR}}=O(d\ell)$ epochs, and implementing the oracle calling steps for the Φ-evaluation problem $(Q,R)^{\Phi}_{\mathsf{eval}}$ and the equation-merging problem $(Q,R)_{k\mathsf{eqs}\mapsto 1\mathsf{eq}}$ with the corresponding (constant round) interactive search systems from Section 4. The second step is again using Theorem 3.4 to compose the

main oracle-aided SMR-IS for L from Proposition 5.3 with the resulted next-layer SMR-IS $\Sigma_{\mathsf{next\text{-}layer}}$, and the (non-interactive) system for the check-eval problem $(Q, R)^V_{\mathsf{check\text{-}eval}}$ from Section 4.8. □

References

[ACY22a] Gal Arnon, Alessandro Chiesa, and Eylon Yogev. A PCP Theorem for Interactive Proofs and Applications. In *Proceedings of the 41st Annual International Conference on the Theory and Applications of Cryptographic Techniques* (*EUROCRYPT '22*), pages 64–94, Cham, 2022. Springer International Publishing.

[ACY22b] Gal Arnon, Alessandro Chiesa, and Eylon Yogev. Hardness of Approximation for Stochastic Problems via Interactive Oracle Proofs. In *Proceedings of the 37th Computational Complexity Conference* (*CCC '22*), volume 234 of *Leibniz International Proceedings in Informatics* (*LIPIcs*), pages 24:1–24:16, Dagstuhl, Germany, 2022. Schloss Dagstuhl – Leibniz-Zentrum für Informatik.

[ALM+98] Sanjeev Arora, Carsten Lund, Rajeev Motwani, Madhu Sudan, and Mario Szegedy. Proof Verification and the Hardness of Approximation Problems. *J. ACM*, 45(3):501–555, 1998.

[BB18] Ben Berger and Zvika Brakerski. Brief Announcement: Zero-Knowledge Protocols for Search Problems. In *Proceedings of the 45th International Colloquium on Automata, Languages, and Programming* (*ICALP '18*), pages 105–1. Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 2018.

[BFLS91] László Babai, Lance Fortnow, Leonid A. Levin, and Mario Szegedy. Checking Computations in Polylogarithmic Time. In *Proceedings of the 23rd Annual ACM Symposium on Theory of Computing* (*STOC '91*), page 21–32, New York, NY, USA, 1991. Association for Computing Machinery.

[BSCS16] Eli Ben-Sasson, Alessandro Chiesa, and Nicholas Spooner. Interactive Oracle Proofs. In Martin Hirt and Adam Smith, editors, *Theory of Cryptography* (*TCC '16*), pages 31–60, Berlin, Heidelberg, 2016. Springer Berlin Heidelberg.

[Can00] Ran Canetti. Security and Composition of Multiparty Cryptographic Protocols. *Journal of Cryptology*, 13(1):143–202, 2000.

[FGL+96] Uriel Feige, Shafi Goldwasser, Laszlo Lovász, Shmuel Safra, and Mario Szegedy. Interactive Proofs and the Hardness of Approximating Cliques. *J. ACM*, 43(2):268–292, March 1996.

[GKR15] Shafi Goldwasser, Yael Tauman Kalai, and Guy N. Rothblum. Delegating Computation: Interactive Proofs for Muggles. *J. ACM*, 62(4), 2015.

[GMR85] Shafi Goldwasser, Silvio Micali, and Chales Rackoff. The Knowledge Complexity of Interactive Proof Systems. In *Proceedings of the 17th Annual ACM Symposium on Theory of Computing* (*STOC '85*), page 291–304, New York, NY, USA, 1985. Association for Computing Machinery.

[Gol04] Oded Goldreich. *Foundations of Cryptography, Volume 2*. Cambridge university press Cambridge, 2004.

[Gol18] Oded Goldreich. On Doubly-Efficient Interactive Proof Systems. *Foundations and Trends in Theoretical Computer Science*, 13(3):158–246, April 2018.

[LFKN92] Carsten Lund, Lance Fortnow, Howard Karloff, and Noam Nisan. Algebraic Methods for Interactive Proof Systems. *J. ACM*, 39(4):859–868, October 1992.

[LS91] Dror Lapidot and Adi Shamir. Fully Parallelized Multi Prover Protocols for NEXP-Time. In *Proceedings of the 32nd Annual Symposium of Foundations of Computer Science* (*FOCS '91*), pages 13–18. IEEE, 1991.

[RRR16] Omer Reingold, Guy N. Rothblum, and Ron D. Rothblum. Constant-Round Interactive Proofs for Delegating Computation. In *Proceedings of the 48th Annual ACM Symposium on Theory of Computing* (*STOC '16*), page 49–62, New York, NY, USA, 2016. Association for Computing Machinery.

ECCC ISSN 1433-8092

https://eccc.weizmann.ac.il