Lecture 6

Recap

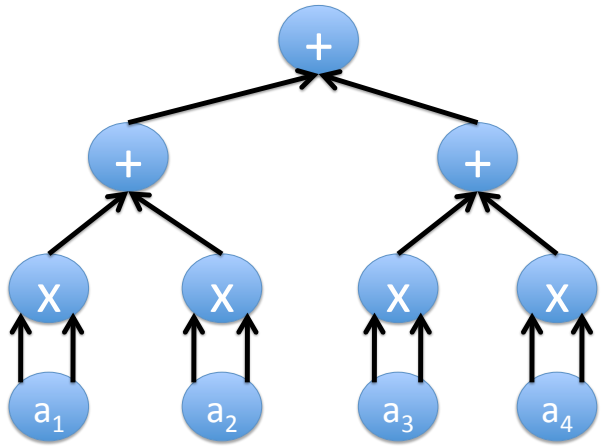
- Two lectures ago: Our first application of the sum-check protocol.
  - An IP for #SAT with a polynomial-time verifier.
  - P ran in time exponential in the input size.
  - But the fastest known algorithm for this problem requires exponential time.
    - So can't really hope for a faster prover for this problem.
- Last lecture we saw some doubly-efficient IPs.
  - V runs in **linear** time.
  - P runs in polynomial time.
    - In fact, we achieved "super-efficiency".
    - meaning P ran the fastest known algorithm for the problem, and then did a low-order amount of additional work to prove correctness.
    - Counting triangles, matrix multiplication.

Today: A General-Purpose Doubly-Efficient Interactive Proof

General-Purpose Interactive Proof and Argument Implementations

- Start with a computer program written in high-level programming language (C, Java, etc.)
- Step 1: Turn the program into an equivalent model amenable to probabilistic checking.
  - Typically some type of arithmetic circuit.
  - Called the **Front End** of the system.
- Step 2: Run an interactive proof or argument on the circuit.
  - Called the **Back End** of the system.

Front End



P and V run interactive proof or argument system (back end) on circuit

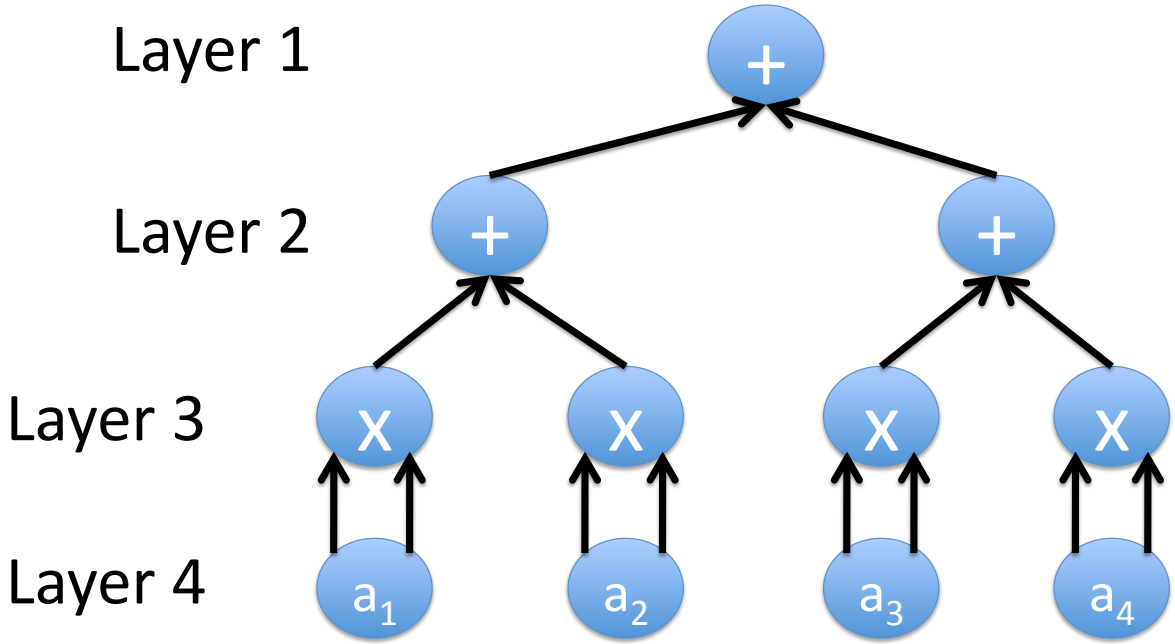
Sources of Prover Overhead in VC Systems

Source of Overhead	P Overhead vs. Native **(Crude Estimate)**	Slowdown Depends On…
Front End (overhead due to using a circuit representation of the computation)	(ratio of circuit size to number of machine steps of original program) **1x-10,000x**	• How amenable is the high-level computer program is to representation via circuits? • What type of circuits can the back-end handle?
Back-End	(ratio of P time to evaluating circuit gate-by-gate) **10x-1,000x**	• Varies by back-end and computation structure (e.g., data parallel?)

The GKR Protocol

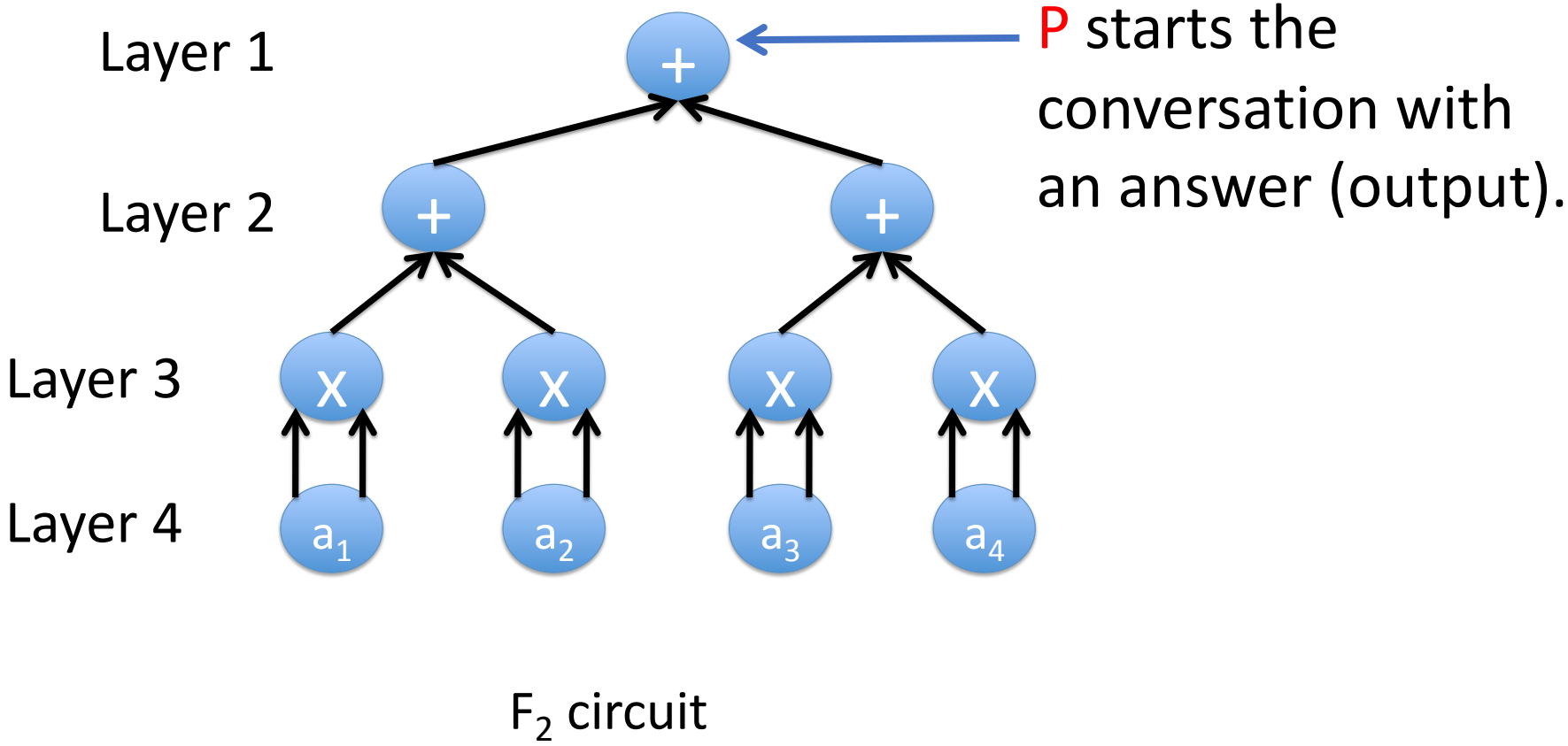
A General-Purpose Doubly-Efficient Interactive Proof

The GKR Protocol: Overview



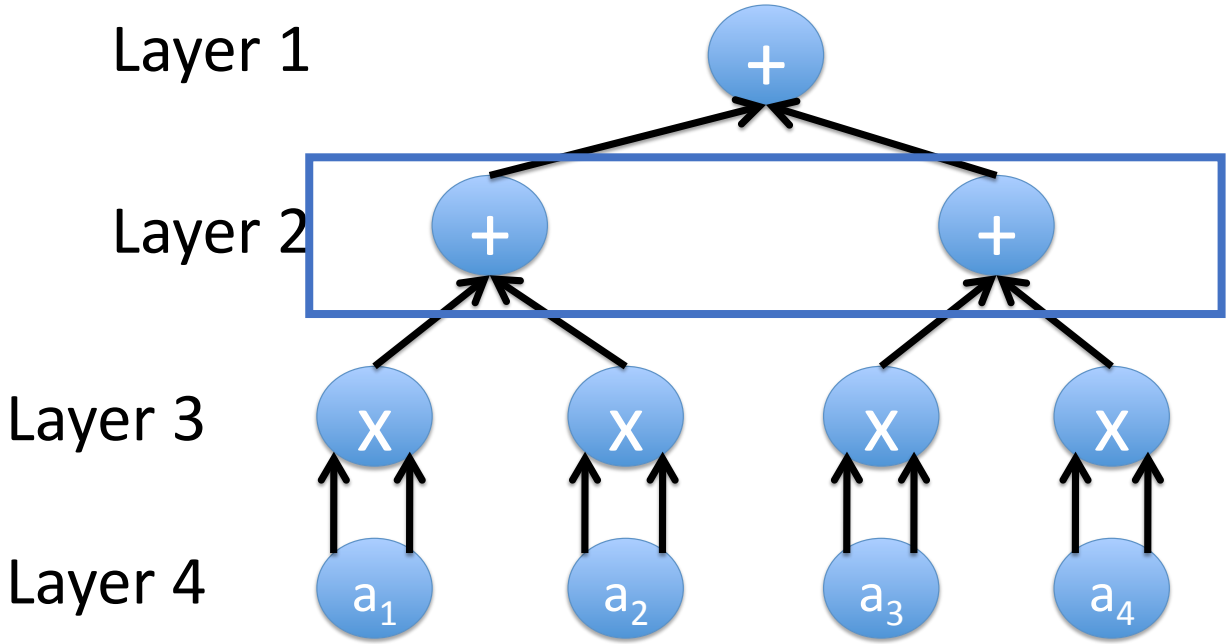
F_2 circuit

The GKR Protocol: Overview



F_2 circuit

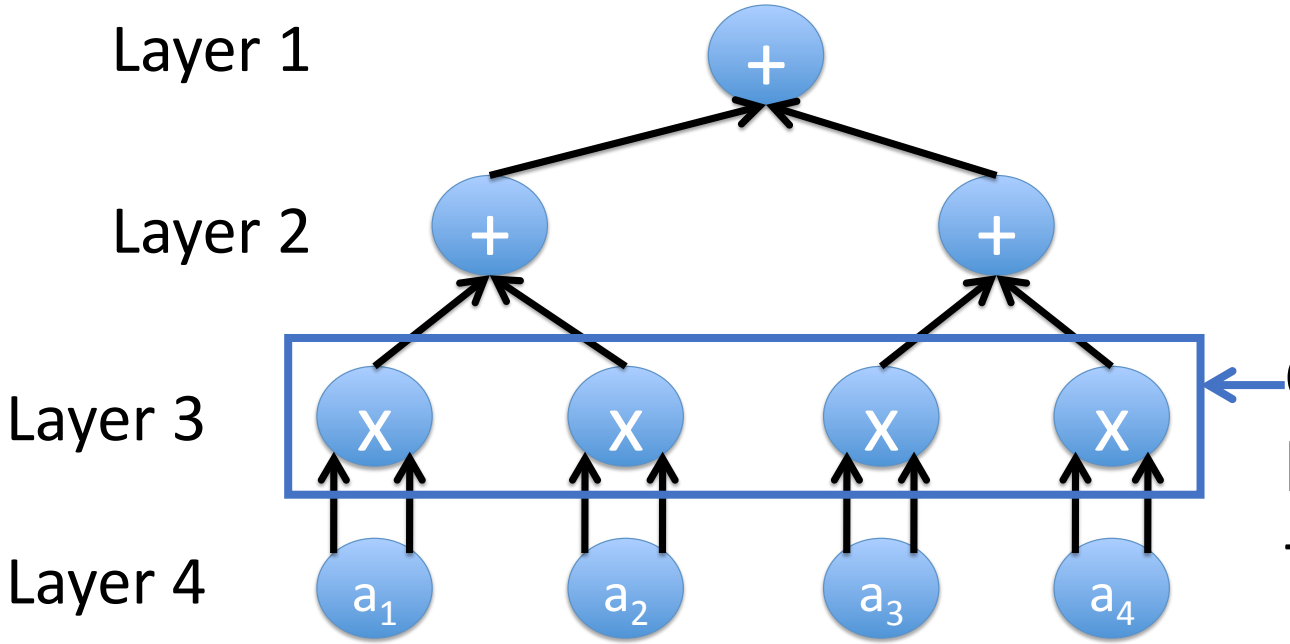
The GKR Protocol: Overview



V sends series of challenges. P responds with info about next circuit level.

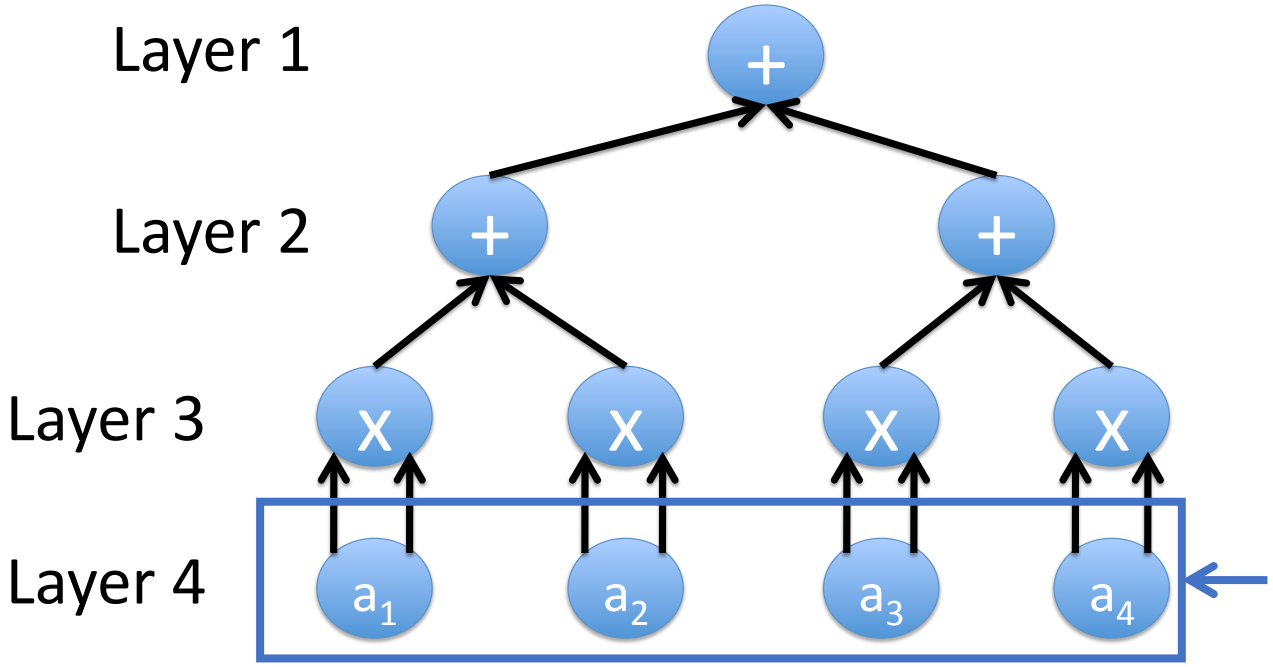
F_2 circuit

The GKR Protocol: Overview



F_2 circuit

The GKR Protocol: Overview



F_2 circuit

Finally, P says something about the (multilinear extension of the) input.

Notation

- Assume layers i and $i+1$ of C have S gates each.
  - Assign each gate a binary label ($\log S$ bits).
- Let $W_i(\boldsymbol{a})$: $\{0,1\}^{\log S} \rightarrow \boldsymbol{F}$ output the value of gate $\boldsymbol{a}$ at layer i.

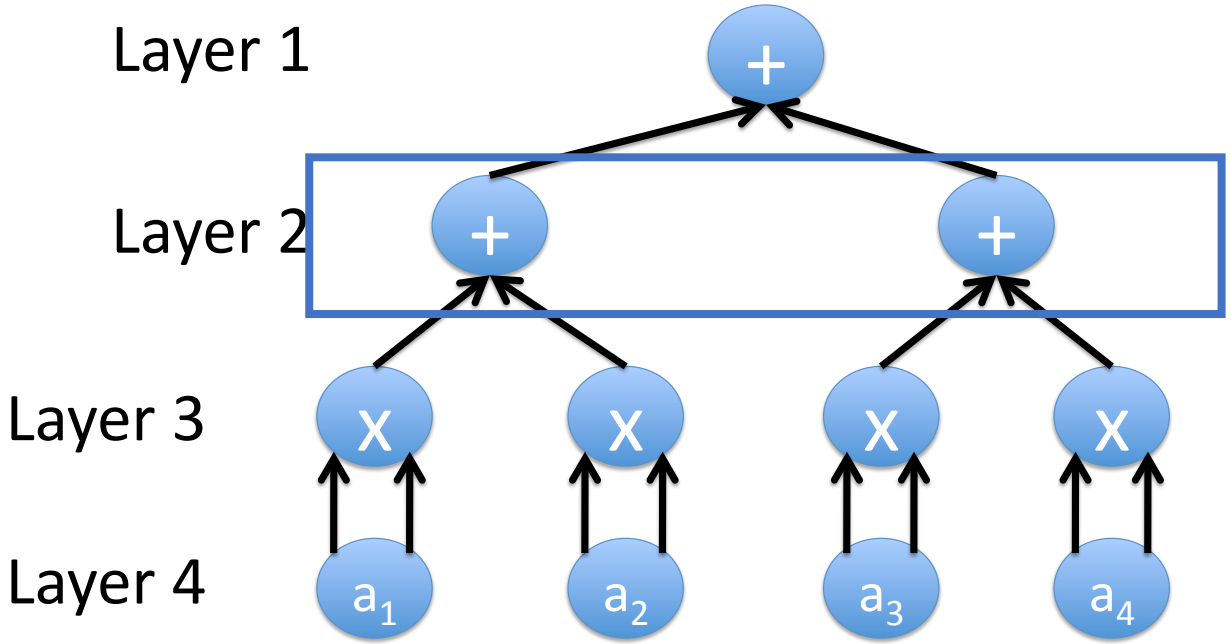
Notation

- Assume layers i and $i+1$ of C have S gates each.
  - Assign each gate a binary label ($\log S$ bits).
- Let $W_i(\boldsymbol{a})\colon \{0,1\}^{\log S} \to \boldsymbol{F}$ output the value of gate $\boldsymbol{a}$ at layer i.
- Let $\mathrm{add}_i(\boldsymbol{a},\boldsymbol{b},\boldsymbol{c})\colon \{0,1\}^{3\log S} \to \boldsymbol{F}$ output 1 iff $(\boldsymbol{b},\boldsymbol{c}) = \left(\mathrm{in}_{\boldsymbol{1}}(\boldsymbol{a}), \mathrm{in}_{\boldsymbol{2}}(\boldsymbol{a})\right)$ and gate $\boldsymbol{a}$ at layer i is an addition gate.

Notation

- Assume layers i and $i+1$ of C have S gates each.
  - Assign each gate a binary label ($\log S$ bits).
- Let $W_i(\boldsymbol{a})\colon \{0,1\}^{\log S} \to \boldsymbol{F}$ output the value of gate $\boldsymbol{a}$ at layer i.
- Let $\mathrm{add}_i(\boldsymbol{a},\boldsymbol{b},\boldsymbol{c})\colon \{0,1\}^{3\log S} \to \boldsymbol{F}$ output 1 iff $(\boldsymbol{b},\boldsymbol{c}) = \left(\mathrm{in}_{\boldsymbol{1}}(\boldsymbol{a}), \mathrm{in}_{\boldsymbol{2}}(\boldsymbol{a})\right)$ and gate $\boldsymbol{a}$ at layer i is an addition gate.
- Let $\mathrm{mult}_i(\boldsymbol{a},\boldsymbol{b},\boldsymbol{c})\colon \{0,1\}^{3\log S} \to \boldsymbol{F}$ output 1 iff $(\boldsymbol{b},\boldsymbol{c}) = \left(\mathrm{in}_{\boldsymbol{1}}(\boldsymbol{a}), \mathrm{in}_{\boldsymbol{2}}(\boldsymbol{a})\right)$ and gate $\boldsymbol{a}$ at layer i is a multiplication gate.

The GKR Protocol: Overview

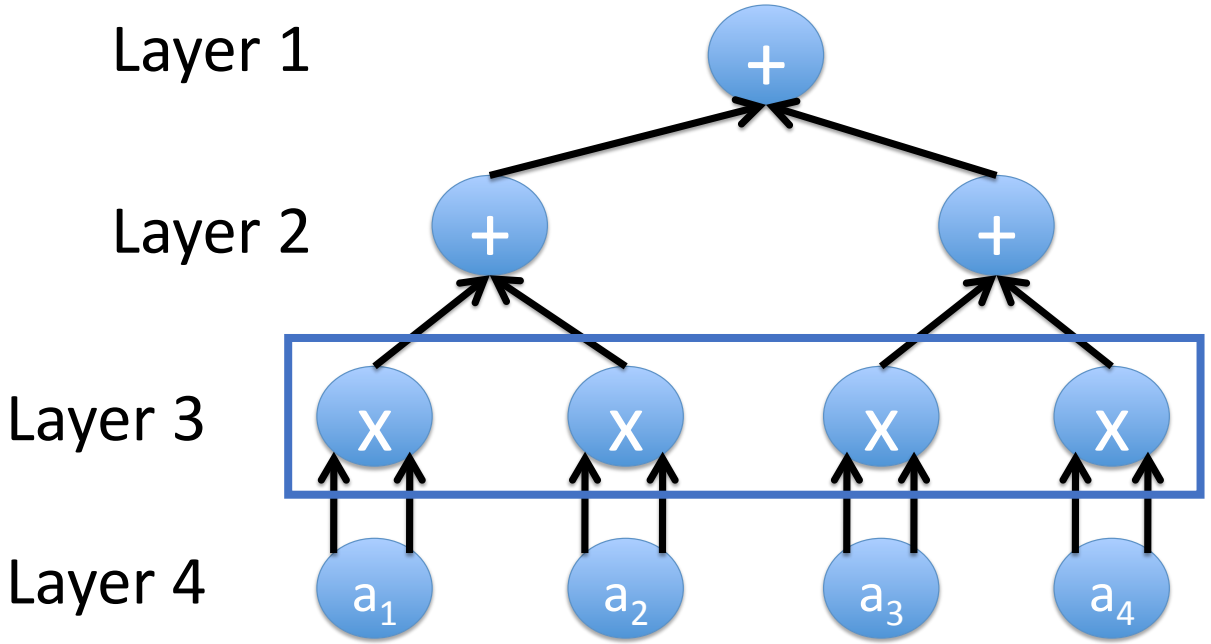


$\mathrm{add}_2(0,(0,0),(0,1)) = 1$

$\mathrm{add}_2(1,(1,0),(1,1)) = 1$

F_2 circuit

The GKR Protocol: Overview



F_2 circuit

$$\text{mult}_3\big((0,0),(0,0),(0,0)\big) = 1$$
$$\text{mult}_3\big((0,1),(0,1),(0,1)\big) = 1$$
$$\text{mult}_3\big((1,0),(1,0),(1,0)\big) = 1$$
$$\text{mult}_3\big((1,1),(1,1),(1,1)\big) = 1$$

GKR Protocol: Goal of Iteration i

- Iteration i starts with a claim from P about $\widetilde{W}_i(\boldsymbol{r}_1)$ for a random point $\boldsymbol{r}_1 \in \boldsymbol{F}^{\log S}$.
- Goal: Reduce this to a claim about $\widetilde{W}_{i+1}(\boldsymbol{r}_2)$ for a random point $\boldsymbol{r}_2 \in \boldsymbol{F}^{\log S}$.
- Observation: $W_i(\boldsymbol{a}) =$

$$\sum_{\boldsymbol{b},\boldsymbol{c}\in\{0,1\}^{\log s}}[\mathrm{add}_i(\boldsymbol{a},\boldsymbol{b},\boldsymbol{c})(W_{i+1}(\boldsymbol{b})+W_{i+1}(\boldsymbol{c}))+\mathrm{mult}_i(\boldsymbol{a},\boldsymbol{b},\boldsymbol{c})(W_{i+1}(\boldsymbol{b})\cdot W_{i+1}(\boldsymbol{c}))]$$

- Hence, the following equality holds as formal polynomials:

$$\widetilde{W}_i(\boldsymbol{a}) =$$

$$\sum_{\boldsymbol{b},\boldsymbol{c}\in\{0,1\}^{\log s}}[\widetilde{\mathrm{add}}_i(\boldsymbol{a},\boldsymbol{b},\boldsymbol{c})(\widetilde{W}_{i+1}(\boldsymbol{b})+\widetilde{W}_{i+1}(\boldsymbol{c}))+\widetilde{\mathrm{mult}}_i(\boldsymbol{a},\boldsymbol{b},\boldsymbol{c})(\widetilde{W}_{i+1}(\boldsymbol{b})\cdot\widetilde{W}_{i+1}(\boldsymbol{c}))]$$

GKR Protocol: Goal of Iteration i

- So V applies sum-check protocol to compute
- $\widetilde{W}_i(\boldsymbol{r}_1) = \sum_{\boldsymbol{b},\boldsymbol{c}\in\{0,1\}^{\log s}} g(\boldsymbol{b},\boldsymbol{c})$, where:

$$g(\boldsymbol{b},\boldsymbol{c}) = \widetilde{\mathrm{add}}_i(\boldsymbol{r}_1,\boldsymbol{b},\boldsymbol{c})(\widetilde{W}_{i+1}(\boldsymbol{b}) + \widetilde{W}_{i+1}(\boldsymbol{c})) + \widetilde{\mathrm{mult}}_i(\boldsymbol{r}_1,\boldsymbol{b},\boldsymbol{c})(\widetilde{W}_{i+1}(\boldsymbol{b}) \cdot \widetilde{W}_{i+1}(\boldsymbol{c}))$$

GKR Protocol: Goal of Iteration i

- So V applies sum-check protocol to compute
- $\widetilde{W}_i(\boldsymbol{r}_1) = \sum_{\boldsymbol{b},\boldsymbol{c}\in\{0,1\}^{\log s}} g(\boldsymbol{b},\boldsymbol{c})$, where:

$$g(\boldsymbol{b},\boldsymbol{c}) = \widetilde{\mathrm{add}}_i(\boldsymbol{r}_1,\boldsymbol{b},\boldsymbol{c})(\widetilde{W}_{i+1}(\boldsymbol{b}) + \widetilde{W}_{i+1}(\boldsymbol{c})) + \widetilde{\mathrm{mult}}_i(\boldsymbol{r}_1,\boldsymbol{b},\boldsymbol{c})(\widetilde{W}_{i+1}(\boldsymbol{b}) \cdot \widetilde{W}_{i+1}(\boldsymbol{c}))$$

- At end of sum-check protocol, V must evaluate $g(\boldsymbol{r}_2,\boldsymbol{r}_3)$.

GKR Protocol: Goal of Iteration i

- So V applies sum-check protocol to compute
- $\widetilde{W}_i(\boldsymbol{r}_1) = \sum_{\boldsymbol{b},\boldsymbol{c}\in\{0,1\}^{\log s}} g(\boldsymbol{b},\boldsymbol{c})$, where:

$$g(\boldsymbol{b},\boldsymbol{c}) = \widetilde{\mathrm{add}}_i(\boldsymbol{r}_1,\boldsymbol{b},\boldsymbol{c})(\widetilde{W}_{i+1}(\boldsymbol{b}) + \widetilde{W}_{i+1}(\boldsymbol{c})) + \widetilde{\mathrm{mult}}_i(\boldsymbol{r}_1,\boldsymbol{b},\boldsymbol{c})(\widetilde{W}_{i+1}(\boldsymbol{b}) \cdot \widetilde{W}_{i+1}(\boldsymbol{c}))$$

- At end of sum-check protocol, V must evaluate $g(\boldsymbol{r}_2,\boldsymbol{r}_3)$.
- Let us assume V can compute $\widetilde{\mathrm{add}}_i(\boldsymbol{r}_1,\boldsymbol{r}_2,\boldsymbol{r}_3)$ and $\widetilde{\mathrm{mult}}_i(\boldsymbol{r}_1,\boldsymbol{r}_2,\boldsymbol{r}_3)$ unaided in time $\mathrm{polylog}(n)$.
- Then V only needs to know $\widetilde{W}_{i+1}(\boldsymbol{r}_2)$ and $\widetilde{W}_{i+1}(\boldsymbol{r}_3)$ to complete this check.

GKR Protocol: Goal of Iteration i

- So V applies sum-check protocol to compute
- $\widetilde{W}_i(\boldsymbol{r}_1) = \sum_{\boldsymbol{b},\boldsymbol{c}\in\{0,1\}^{\log s}} g(\boldsymbol{b},\boldsymbol{c})$, where:

$$g(\boldsymbol{b},\boldsymbol{c}) = \widetilde{\mathrm{add}}_i(\boldsymbol{r}_1,\boldsymbol{b},\boldsymbol{c})(\widetilde{W}_{i+1}(\boldsymbol{b}) + \widetilde{W}_{i+1}(\boldsymbol{c})) + \widetilde{\mathrm{mult}}_i(\boldsymbol{r}_1,\boldsymbol{b},\boldsymbol{c})(\widetilde{W}_{i+1}(\boldsymbol{b}) \cdot \widetilde{W}_{i+1}(\boldsymbol{c}))$$

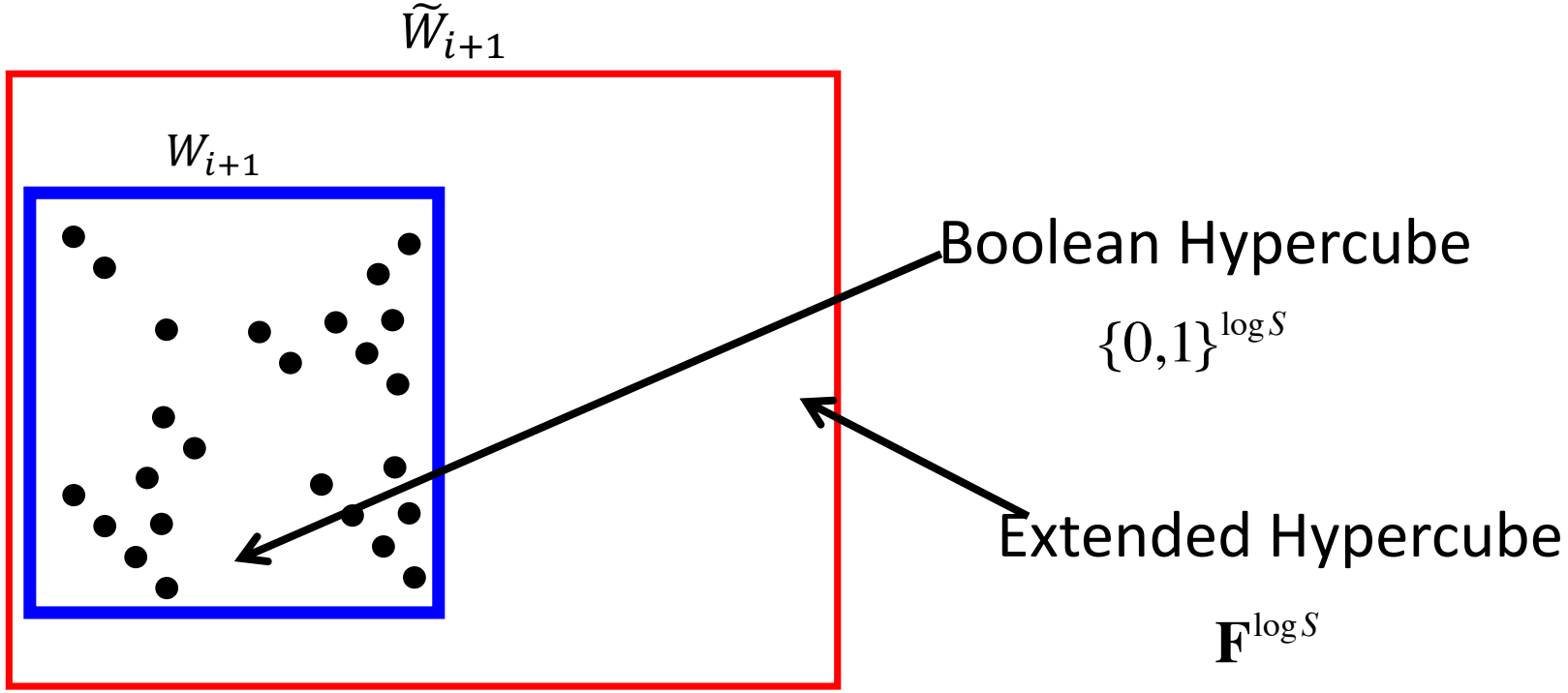
- At end of sum-check protocol, V must evaluate $g(\boldsymbol{r}_2,\boldsymbol{r}_3)$.
- Let us assume V can compute $\widetilde{\mathrm{add}}_i(\boldsymbol{r}_1,\boldsymbol{r}_2,\boldsymbol{r}_3)$ and $\widetilde{\mathrm{mult}}_i(\boldsymbol{r}_1,\boldsymbol{r}_2,\boldsymbol{r}_3)$ unaided in time $\mathrm{polylog}(n)$.
- Then V only needs to know $\widetilde{W}_{i+1}(\boldsymbol{r}_2)$ and $\widetilde{W}_{i+1}(\boldsymbol{r}_3)$ to complete this check.
- Iteration $i+1$ is devoted to computing these values.

Remaining Issue: Reducing to Verification of a Single Point

- There is one remaining problem: we don't want to have to separately verify both $\widetilde{W}_{i+1}(\boldsymbol{r}_2)$ and $\widetilde{W}_{i+1}(\boldsymbol{r}_3)$ in iteration i +1.
- Solution: Reduce verifying both of the above values to verifying $\widetilde{W}_{i+1}(\boldsymbol{r}_4)$ for a single point $\boldsymbol{r}_4$.

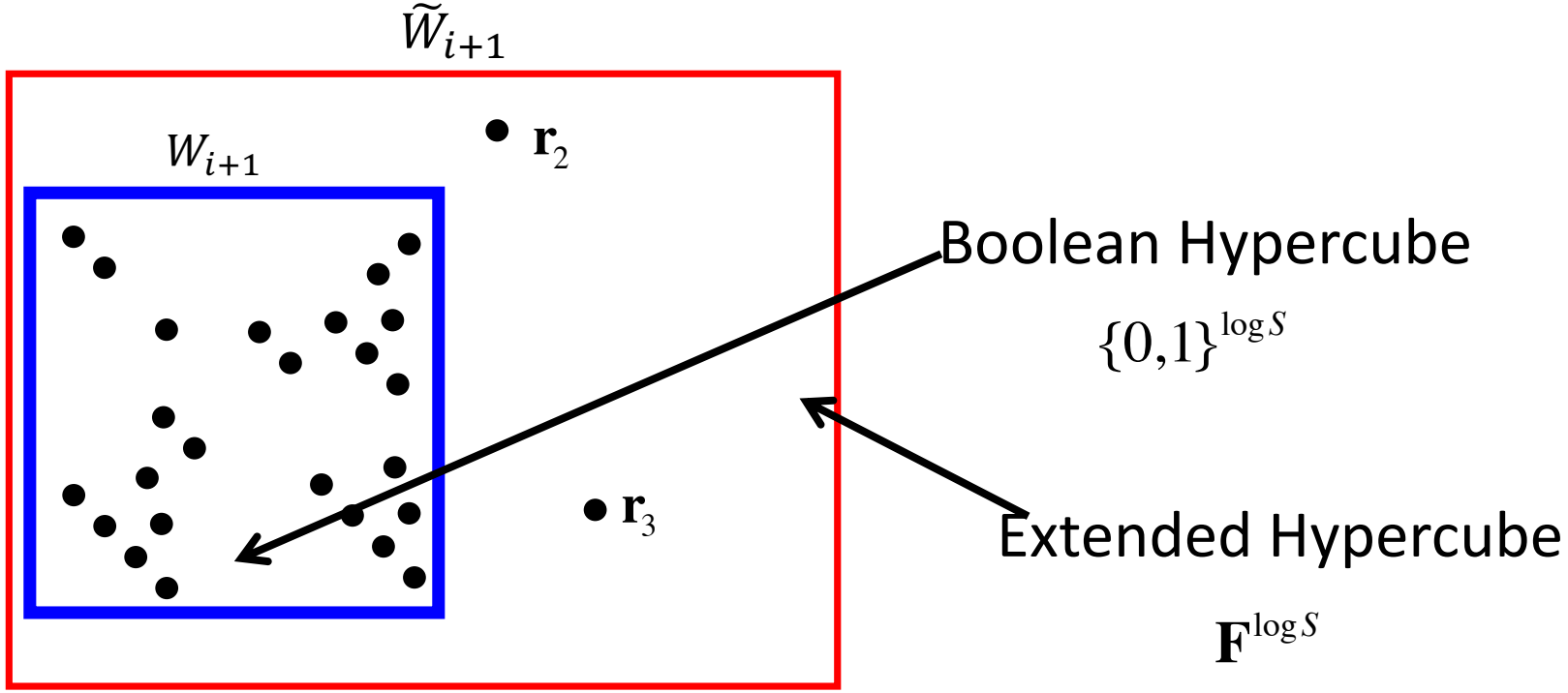
Remaining Issue: Reducing to Verification of a Single Point

- There is one remaining problem: we don't want to have to separately verify both $\widetilde{W}_{i+1}(\mathbf{r}_2)$ and $\widetilde{W}_{i+1}(\mathbf{r}_3)$ in iteration i +1.
- Solution: Reduce verifying both of the above values to verifying $\widetilde{W}_{i+1}(\mathbf{r}_4)$ for a single point $\mathbf{r}_4 \in \mathbf{F}^{\log S}$.



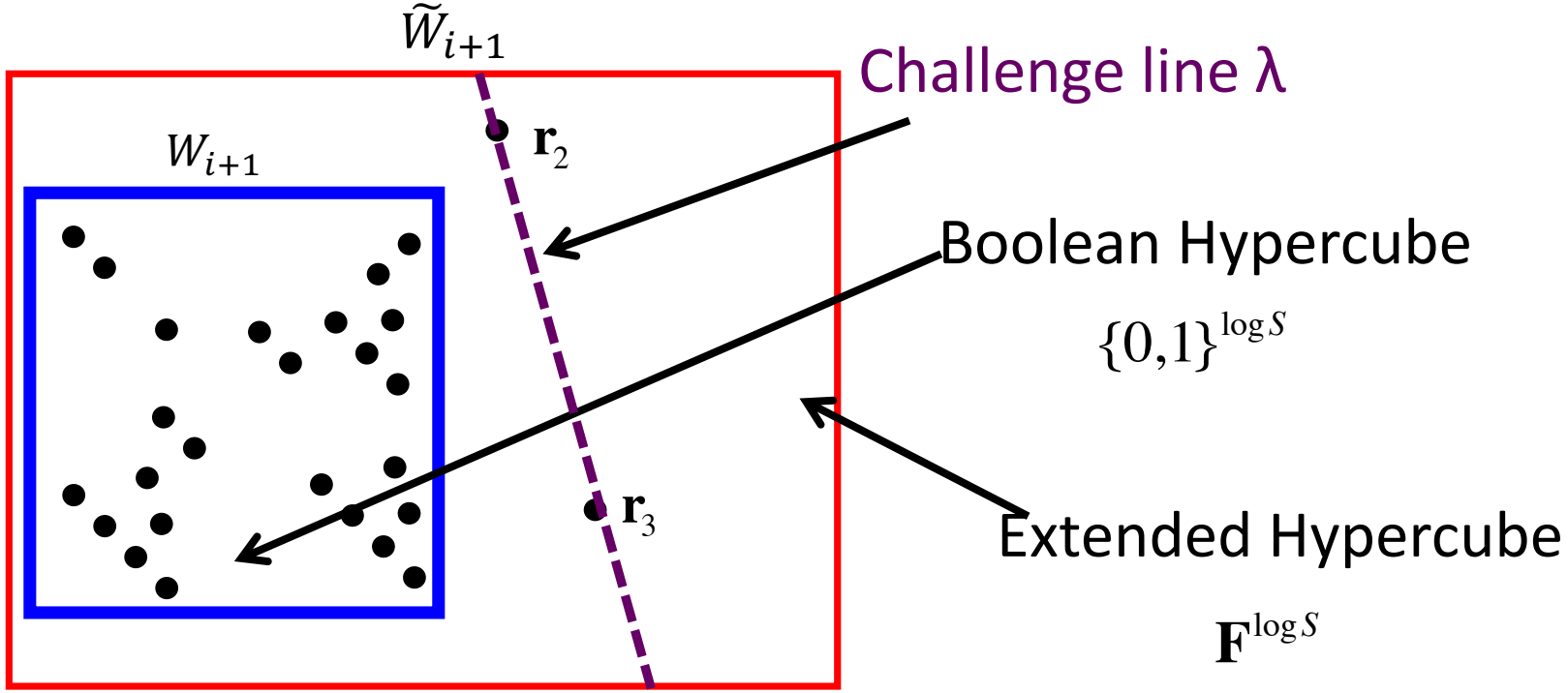
Remaining Issue: Reducing to Verification of a Single Point

- There is one remaining problem: we don't want to have to separately verify both $\widetilde{W}_{i+1}(\mathbf{r}_2)$ and $\widetilde{W}_{i+1}(\mathbf{r}_3)$ in iteration i +1.
- Solution: Reduce verifying both of the above values to verifying $\widetilde{W}_{i+1}(\mathbf{r}_4)$ for a single point $\mathbf{r}_4 \in \mathbf{F}^{\log S}$.



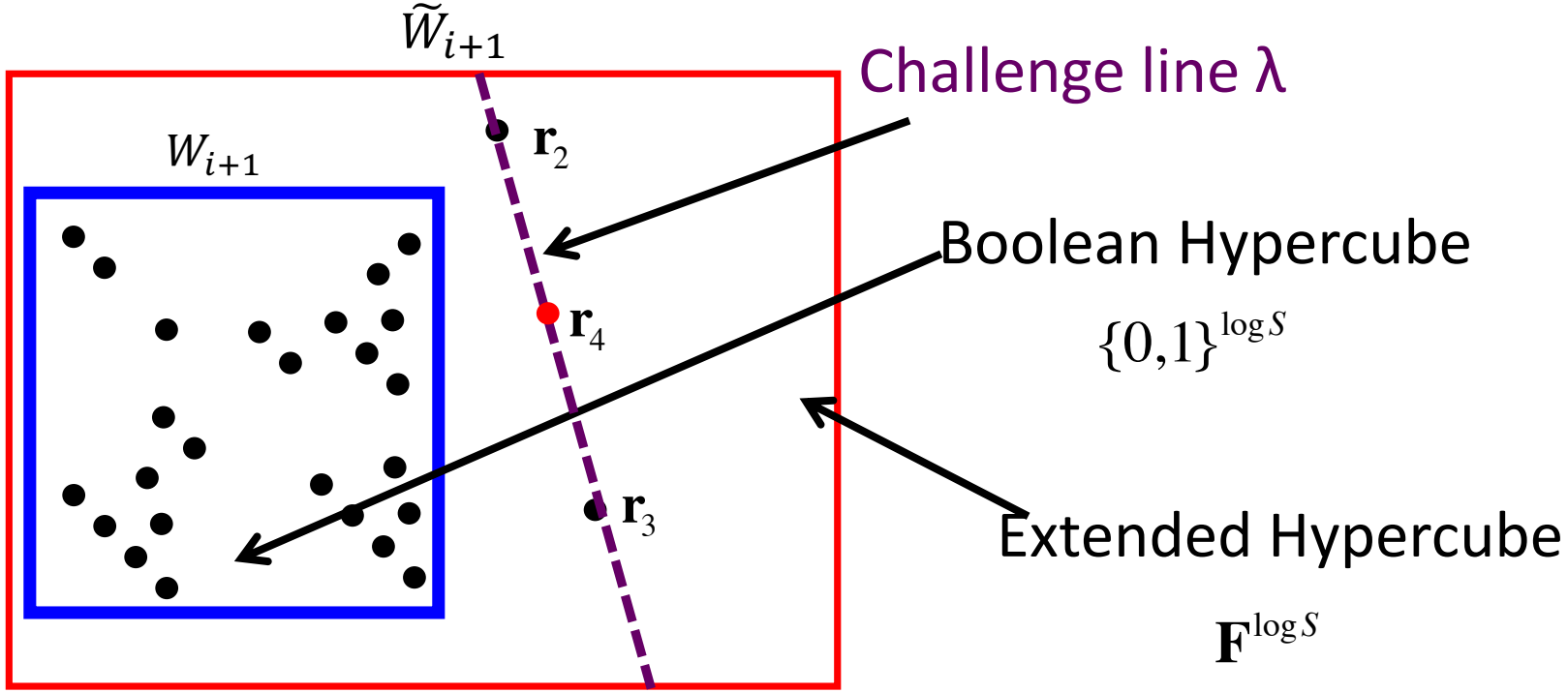
Remaining Issue: Reducing to Verification of a Single Point

- There is one remaining problem: we don't want to have to separately verify both $\widetilde{W}_{i+1}(\mathbf{r}_2)$ and $\widetilde{W}_{i+1}(\mathbf{r}_3)$ in iteration i +1.
- Solution: Reduce verifying both of the above values to verifying $\widetilde{W}_{i+1}(\mathbf{r}_4)$ for a single point $\mathbf{r}_4 \in \mathbf{F}^{\log S}$.



Remaining Issue: Reducing to Verification of a Single Point

- There is one remaining problem: we don't want to have to separately verify both $\widetilde{W}_{i+1}(\mathbf{r}_2)$ and $\widetilde{W}_{i+1}(\mathbf{r}_3)$ in iteration i +1.
- Solution: Reduce verifying both of the above values to verifying $\widetilde{W}_{i+1}(\mathbf{r}_4)$ for a single point $\mathbf{r}_4 \in \mathbf{F}^{\log S}$.



Costs of the GKR protocol

- V time is $O(n + D \log S)$ where n is input size, D is circuit depth, and S is circuit size.
  - Assumes V can compute $\widetilde{\text{add}}_i(\boldsymbol{r}_1, \boldsymbol{r}_2, \boldsymbol{r}_3)$ and $\widetilde{\text{mult}}_i(\boldsymbol{r}_1, \boldsymbol{r}_2, \boldsymbol{r}_3)$ unaided in time $\text{polylog}(n)$
- Communication cost is $O(D \log S)$.

Costs of the GKR protocol

- V time is $O(n + D \log S)$ where n is input size, D is circuit depth, and S is circuit size.
  - Assumes V can compute $\widetilde{\text{add}}_i(\boldsymbol{r}_1, \boldsymbol{r}_2, \boldsymbol{r}_3)$ and $\widetilde{\text{mult}}_i(\boldsymbol{r}_1, \boldsymbol{r}_2, \boldsymbol{r}_3)$ unaided in time $\text{polylog}(n)$
- Communication cost is $O(D \log S)$.

- P time is $O(S)$.
  - A naïve implementation of P takes $\Omega(S^3)$ time, where S is circuit size.
  - A sequence of works has brought this down to $O(S)$, for arbitrary circuits [CMT12, Thaler13, WJBSTWW17, XZZPS19]

GKR Prover Runtime: Details

Recall: Core of the GKR protocol is applying sum-check to compute $\sum_{\boldsymbol{b},\boldsymbol{c}\in\{0,1\}^{\log s}} g(\boldsymbol{b},\boldsymbol{c})$ where

$$g(\boldsymbol{b},\boldsymbol{c}) = \widetilde{\mathrm{add}}_i(\boldsymbol{r}_1,\boldsymbol{b},\boldsymbol{c})(\widetilde{W}_{i+1}(\boldsymbol{b}) + \widetilde{W}_{i+1}(\boldsymbol{c})) + \widetilde{\mathrm{mult}}_i(\boldsymbol{r}_1,\boldsymbol{b},\boldsymbol{c})(\widetilde{W}_{i+1}(\boldsymbol{b}) \cdot \widetilde{W}_{i+1}(\boldsymbol{c}))$$

GKR Prover Runtime: Details

Recall: Core of the GKR protocol is applying sum-check to compute $\sum_{\boldsymbol{b},\boldsymbol{c}\in\{0,1\}^{\log S}} g(\boldsymbol{b},\boldsymbol{c})$ where

$$g(\boldsymbol{b},\boldsymbol{c}) = \widetilde{\mathrm{add}}_i(\boldsymbol{r}_1,\boldsymbol{b},\boldsymbol{c})(\widetilde{W}_{i+1}(\boldsymbol{b}) + \widetilde{W}_{i+1}(\boldsymbol{c})) + \widetilde{\mathrm{mult}}_i(\boldsymbol{r}_1,\boldsymbol{b},\boldsymbol{c})(\widetilde{W}_{i+1}(\boldsymbol{b}) \cdot \widetilde{W}_{i+1}(\boldsymbol{c}))$$

- A naïve implementation of P takes $\Omega(S^3)$ time, where S is circuit size.
  - Same idea as prover implementation for #SAT protocol.
  - i.e., P evaluates g in each round of sum-check at all $O(S^2/2^i)$ necessary points $\boldsymbol{z}$, taking $O(S)$ time per point.

GKR Prover Runtime: Details

Recall: Core of the GKR protocol is applying sum-check to compute $\sum_{\boldsymbol{b},\boldsymbol{c}\in\{0,1\}^{\log S}} g(\boldsymbol{b},\boldsymbol{c})$ where

$$g(\boldsymbol{b},\boldsymbol{c}) = \widetilde{\mathrm{add}}_i(\boldsymbol{r}_1,\boldsymbol{b},\boldsymbol{c})(\widetilde{W}_{i+1}(\boldsymbol{b}) + \widetilde{W}_{i+1}(\boldsymbol{c})) + \widetilde{\mathrm{mult}}_i(\boldsymbol{r}_1,\boldsymbol{b},\boldsymbol{c})(\widetilde{W}_{i+1}(\boldsymbol{b}) \cdot \widetilde{W}_{i+1}(\boldsymbol{c}))$$

- A naïve implementation of P takes $\Omega(S^3)$ time, where S is circuit size.
  - Same idea as prover implementation for #SAT protocol.
  - i.e., P evaluates g in each round of sum-check at all $O(S^2/2^i)$ necessary points $\boldsymbol{z}$, taking $O(S)$ time per point.
- [CMT12]: P time is $O(S \log S)$.
  - Exploit structure in **multilinear** extensions $\widetilde{\mathrm{add}}_i$ and $\widetilde{\mathrm{mult}}_i$. Ensures that each gate of C contributes to $g(\boldsymbol{z})$ for $O(1)$ relevant points $\boldsymbol{z}$ in each round.

GKR Prover Runtime: Details

Recall: Core of the GKR protocol is applying sum-check to compute $\sum_{\boldsymbol{b},\boldsymbol{c}\in\{0,1\}^{\log S}} g(\boldsymbol{b},\boldsymbol{c})$ where

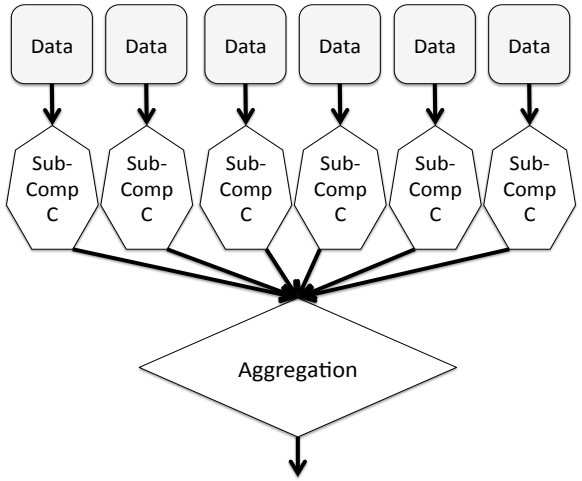
$$g(\boldsymbol{b},\boldsymbol{c}) = \widetilde{\mathrm{add}}_i(\boldsymbol{r}_1,\boldsymbol{b},\boldsymbol{c})(\widetilde{W}_{i+1}(\boldsymbol{b}) + \widetilde{W}_{i+1}(\boldsymbol{c})) + \widetilde{\mathrm{mult}}_i(\boldsymbol{r}_1,\boldsymbol{b},\boldsymbol{c})(\widetilde{W}_{i+1}(\boldsymbol{b}) \cdot \widetilde{W}_{i+1}(\boldsymbol{c}))$$

- A naïve implementation of P takes $\Omega(S^3)$ time, where S is circuit size.
  - Same idea as prover implementation for #SAT protocol.
  - i.e., P evaluates g in each round of sum-check at all $O(S^2/2^i)$ necessary points $\boldsymbol{z}$, taking $O(S)$ time per point.
- [CMT12]: P time is $O(S \log S)$.
  - Exploit structure in **multilinear** extensions $\widetilde{\mathrm{add}}_i$ and $\widetilde{\mathrm{mult}}_i$. Ensures that each gate of C contributes to $g(\boldsymbol{z})$ for $O(1)$ relevant points $\boldsymbol{z}$ in each round.
- All subsequent works seek to bring "Approach 3" to bear on the GKR protocol, letting P reuse work across rounds.

GKR Prover Runtime: Details

- [Thaler13]:
  1. P time $O(S)$ for circuits with "nice" wiring patterns.
  2. P time $O(S \log S')$ for data parallel circuits

     i.e., that apply the same subcomputation (of size S') independently to different pieces of data



GKR Prover Runtime: Details

- [Thaler13]:
  1. P time $O(S)$ for circuits with "nice" wiring patterns.
  2. P time $O(S \log S')$ for data parallel circuits

     i.e., that apply the same subcomputation (of size S') independently to different pieces of data
- [WJBSTWW17] improved the data parallel time to $O(S + S' \log S')$.

GKR Prover Runtime: Details

- [Thaler13]:
  1. P time $O(S)$ for circuits with "nice" wiring patterns.
  2. P time $O(S \log S')$ for data parallel circuits

     i.e., that apply the same subcomputation (of size S') independently to different pieces of data
- [WJBSTWW17] improved the data parallel time to $O(S + S' \log S')$.
- [ZGKPP18] extends to data parallel computations where each subcomputation may not be the same.

GKR Prover Runtime: Details

- [Thaler13]:
  1. P time $O(S)$ for circuits with "nice" wiring patterns.
  2. P time $O(S \log S')$ for data parallel circuits

     i.e., that apply the same subcomputation (of size S') independently to different pieces of data
- [WJBSTWW17] improved the data parallel time to $O(S + S' \log S')$.
- [ZGKPP18] extends to data parallel computations where each subcomputation may not be the same.
- [XZZPS19] achieved $O(S)$ time for general circuits.

Rumination on Generality Vs. Efficiency

Generality vs. Efficiency

- The GKR protocol for circuit evaluation has now been rendered optimally efficient for P (up to constant factors).
- **Any** computation can be represented as a circuit evaluation (or satisfiability) problem.
  - But this can introduce tremendous overheads.
  - The GKR protocol **forces** the prover to compute the output in a prescribed manner, which may be far from optimal (gate-by-gate evaluation of a circuit).
- To achieve scalability, the gold standard is really **super-efficiency**.
  - i.e., P computed the right answer directly using the fastest known algorithm, and did a low-order amount of extra work to prove correctness