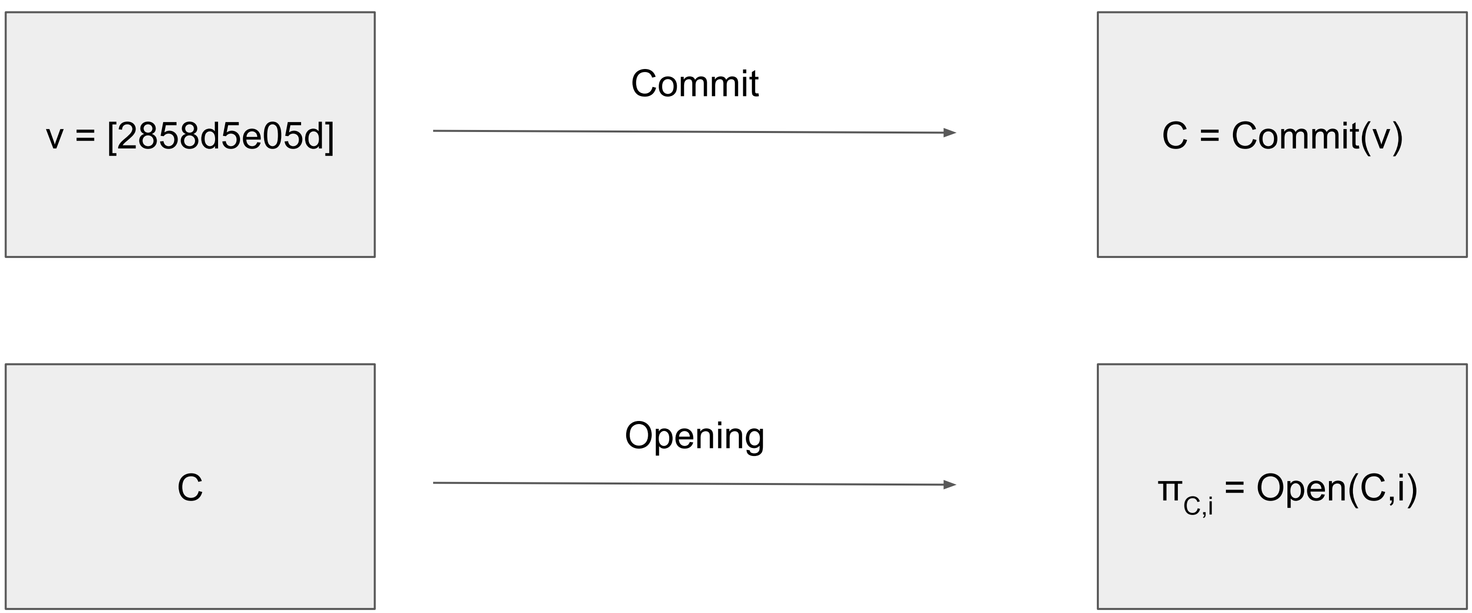
Vector commitments from univariate polynomials and their applications

PRIMES Conference
May 19th, 2019
Yiming Zheng
Alin Tomescu

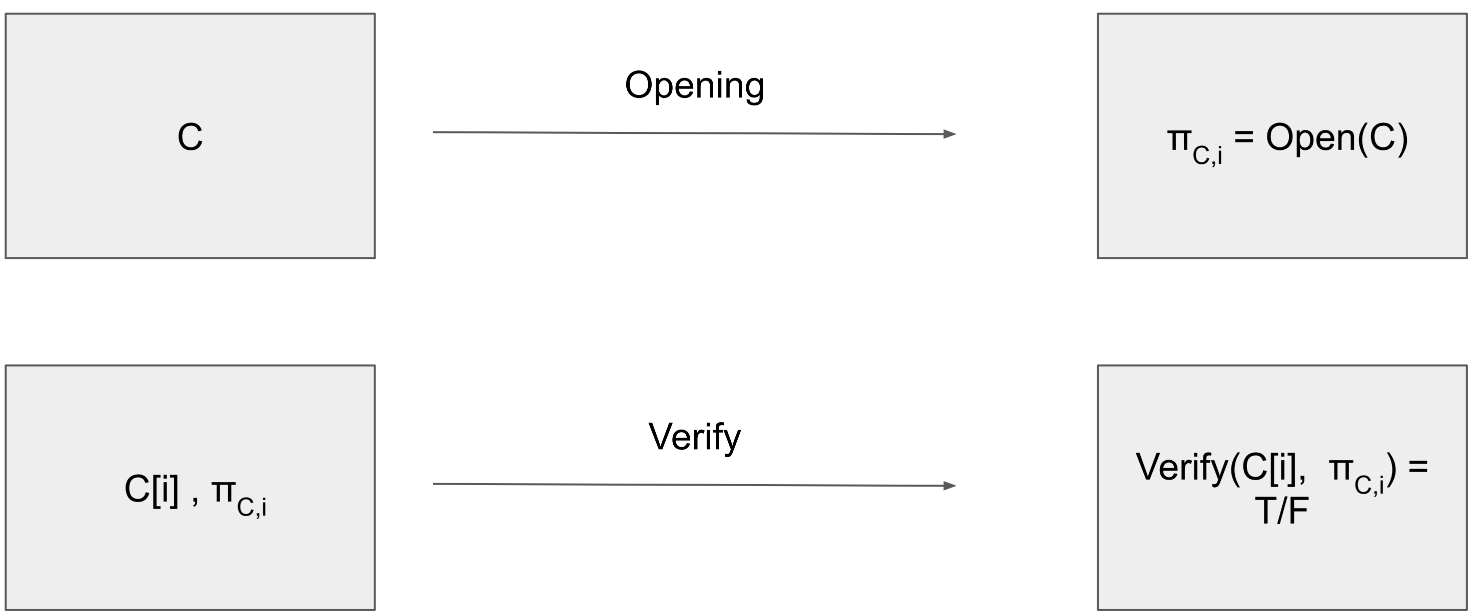
What is a vector commitment (VC) scheme?

- Commitments
  - Take a value, place it in an envelope, seal it, and put the envelope where it is visible to everyone
  - Once the envelope is sealed, the value can't be changed
  - The value remains a secret until the envelope is opened
- Vector commitments
  - A commitment to an ordered sequence of values (i.e. a vector), openings by index



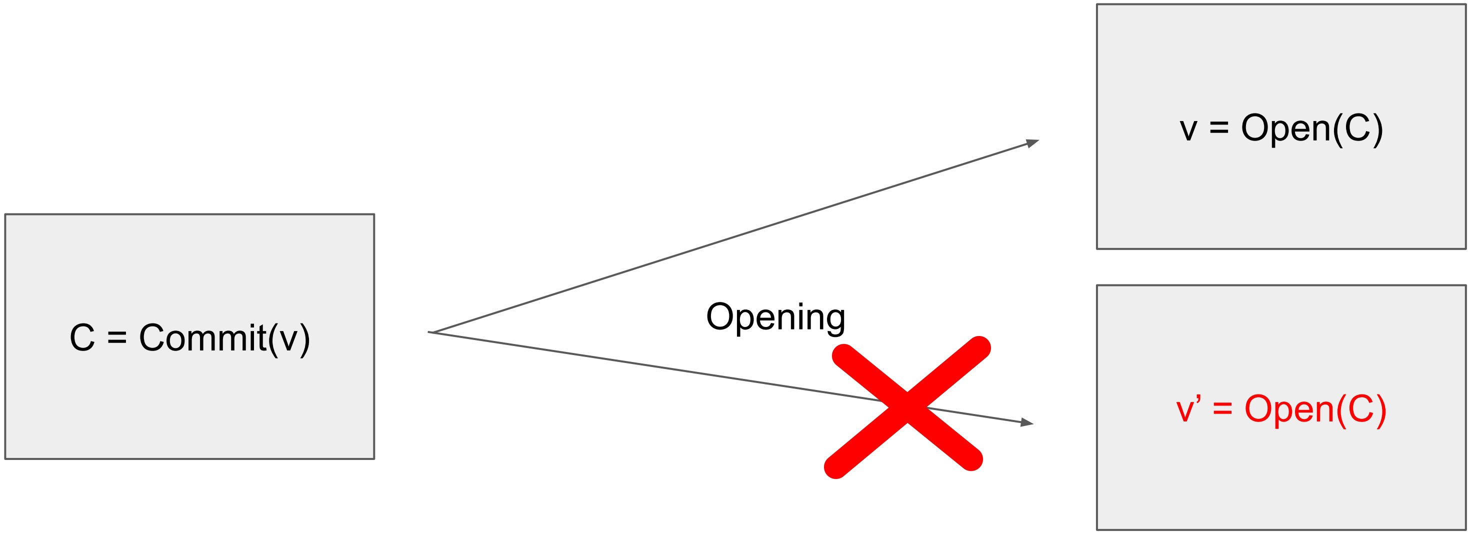
What is a vector commitment (VC) scheme?

- Commitments
  - Take a value, place it in an envelope, seal it, and put the envelope where it is visible to everyone
  - Once the envelope is sealed, the value can’t be changed
  - The value remains a secret until the envelope is opened
- Vector commitments
  - A commitment to an ordered sequence of values (i.e. a vector), openings by index



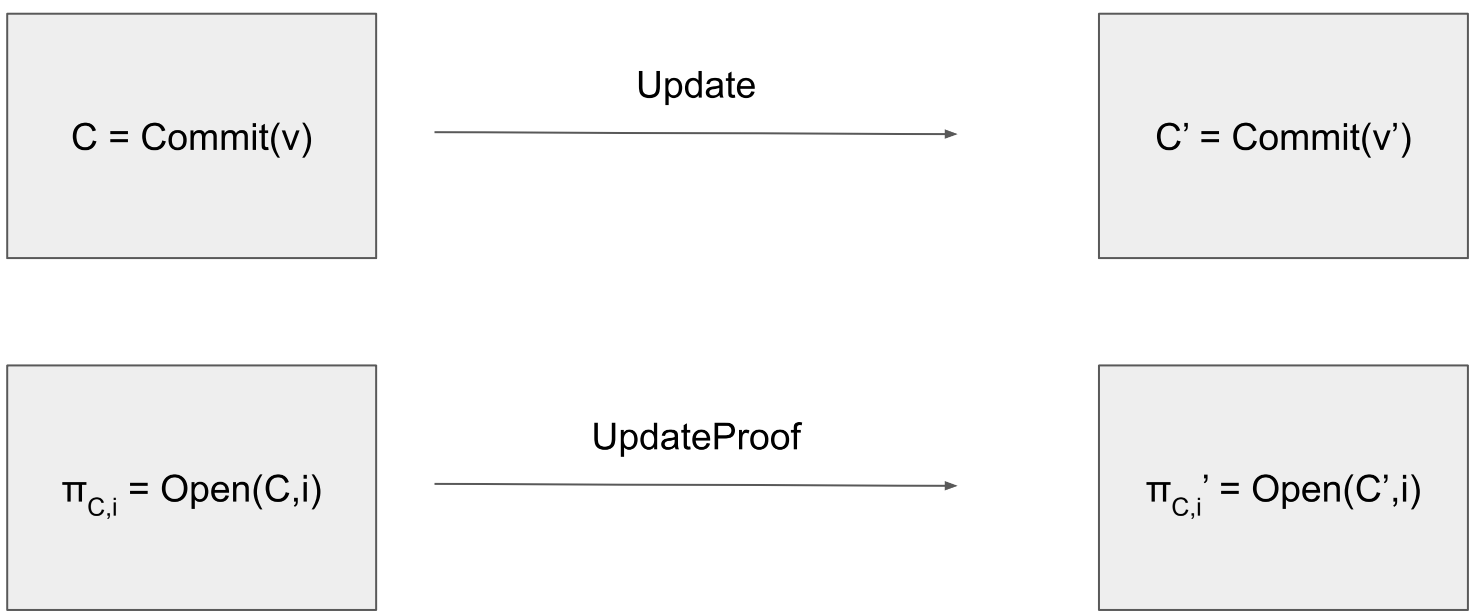
What is a vector commitment (VC) scheme?

- A commitment to an ordered sequence of values (i.e. a vector)
- Position binding
  - no openings to two distinct values at the same index
- Updatability
  - efficient updates for commitments and their proofs



What is a vector commitment (VC) scheme?

- A commitment to an ordered sequence of values (i.e. a vector)
- Position binding
    - no openings to two distinct values at the same index
- Updatability
    - efficient updates for commitments and their proofs



VCs haves many interesting applications

- Verifiable Secret Sharing (VSS)
- Distributed Key Generation (DKG)
- Stateless cryptocurrencies
  - Avoid miners having to store the full blockchain state
- Append-only Authenticated Dictionaries
  - Useful for securing HTTPS, WhatsApp, and email

Catalano & Fiore Vector Commitments

- Generate bilinear groups G_1 and G_2 of prime order p with the bilinear map $e : G_1 \times G_1 \rightarrow G_2$
- Generate a random generator g of G_1 and random integers $z_1, z_2, \ldots, z_n$
- Given a,b,c the bilinear map checks that c is the product of a and b “in the exponent”

$$e(g^a, g^b) \stackrel{?}{=} e(g^c, g)$$

Catalano & Fiore Vector Commitments

- Generate bilinear groups G_1 and G_2 of prime order p with the bilinear map $e : G_1 \times G_1 \rightarrow G_2$
- Generate a random generator g of G_1 and random integers $z_1, z_2, \ldots, z_n$
- Compute the public parameters:

$$h_i = g^{z_i}, h_{i,j} = g^{z_i z_j}, \forall i, j \in \{1, 2, \ldots, n\}$$

- There are $O(n^2)$ public parameters

Catalano & Fiore Vector Commitments

- To commit to a vector $(a_1, a_2, ..., a_n)$, compute

$$C = \prod_{i=1}^{n} h_i^{a_i}$$

- To open at index i, compute

$$\pi_i = \prod_{\substack{1 \leq j \leq n \\ j \neq i}} h_{i,j} = \left(\prod_{\substack{1 \leq j \leq n \\ j \neq i}} g^{a_j z_j} \right)^{z_i}$$

Catalano & Fiore Vector Commitments

- To verify a commitment at index i given the proof $\pi_{C,i}$, check the following

$$e(\frac{C}{h_i^{a_i}}, h_i) = e(\pi_i, g)$$

- If the commitment and proofs are valid, this is equivalent to

$$e(\prod_{\substack{1 \leq j \leq n \\ j \neq i}} g^{a_j z_j}, g^{z_i}) = e((\prod_{\substack{1 \leq j \leq n \\ j \neq i}} g^{a_j z_j})^{z_i}, g)$$

Catalano & Fiore Vector Commitments

- To update the commitment of a vector as it changes from $(a_1,a_2,...a_i,...,a_n) \rightarrow (a_1,a_2,..a_i',...a_n)$, compute

$$C' = C \cdot h_i^{a_i' - a_i}$$

- To update the proof $\pi_{C,j}$ when the vector changes at index i, compute

$$\pi_j' = \pi_j \cdot (h_i^{a_i' - a_i})^{z_j} = \pi_j \cdot h_{j,i}^{a_i' - a_i}$$

Catalano & Fiore Vector Commitments

- To update the proof $\pi_{C,j}$ when the vector changes at index i, compute

$$\pi'_j = \pi_j \cdot (h_i^{a'_i - a_i})^{z_j} = \pi_j \cdot h_{j,i}^{a'_i - a_i}$$

- Updating the proof at index j requires a client to have the verification key consisting of all the $h_{i,j}$'s for fixed j, which has size O(n)

Summary

- Proof size: O(1)
- Proof update time: O(1)
- "Update key" size: O(n)
- Public parameter size: $O(n^2)$

Our scheme from Lagrange polynomials

- Represent a vector v[1,2,...,n] as a polynomial $P(x)$ where $P(i) = v_i$

Our scheme from Lagrange polynomials

- Represent a vector v[1,2,...,n] as a polynomial P(x) where P(i) = v_i
- We can use Lagrange interpolation to obtain

$$P(x) = \sum_{i=1}^{n} L_i(x) v_i$$

- Here, $L_i(x)$ is the ith *Lagrange basis polynomial*, which has the form

$$L_i(x) = \prod_{\substack{1 \le j \le n \\ j \ne i}} \frac{x - j}{i - j}$$

Our scheme from Lagrange polynomials

- Represent a vector v[1,2,...,n] as a polynomial P(x) where P(i) = v_i
- We can use Lagrange interpolation to obtain

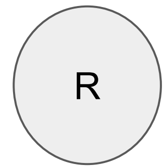
$$P(x) = \sum_{i=1}^{n} L_i(x) v_i$$

- Generate bilinear groups G_1 and G_2 of prime order p with the bilinear map e : $G_1 \times G_1 \rightarrow G_2$
- Compute the public parameters

$$g^s, g^{s^2}, \ldots, g^{s^n}, g^{L_1(s)}, g^{L_2(s)}, \ldots, g^{L_n(s)}$$

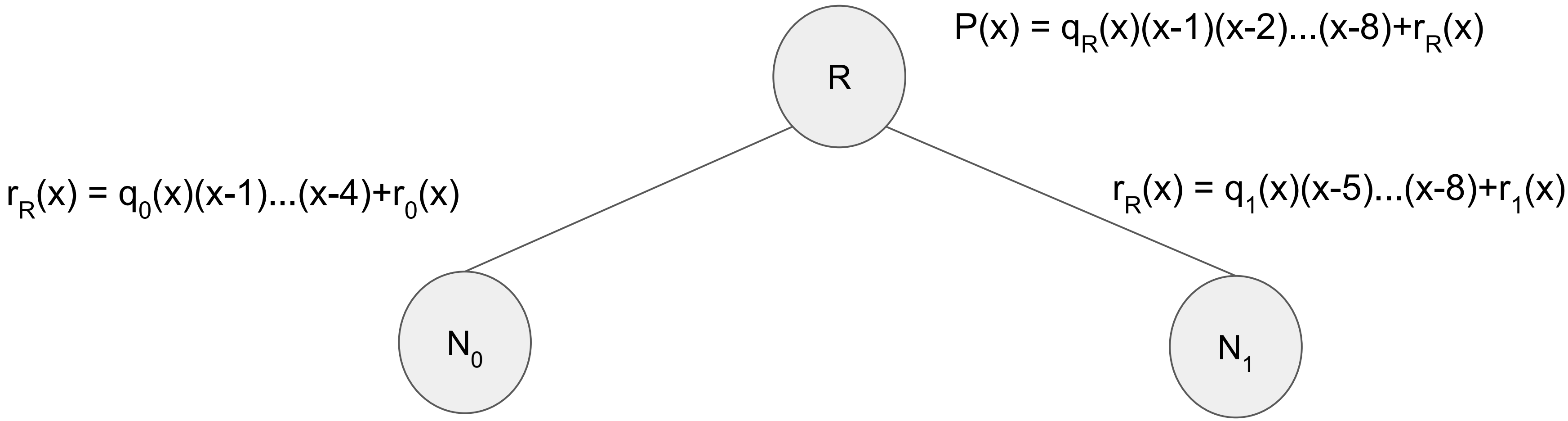
- Use these to compute the commitment to P, which is $g^{P(s)}$

Multipoint Evaluation Trees

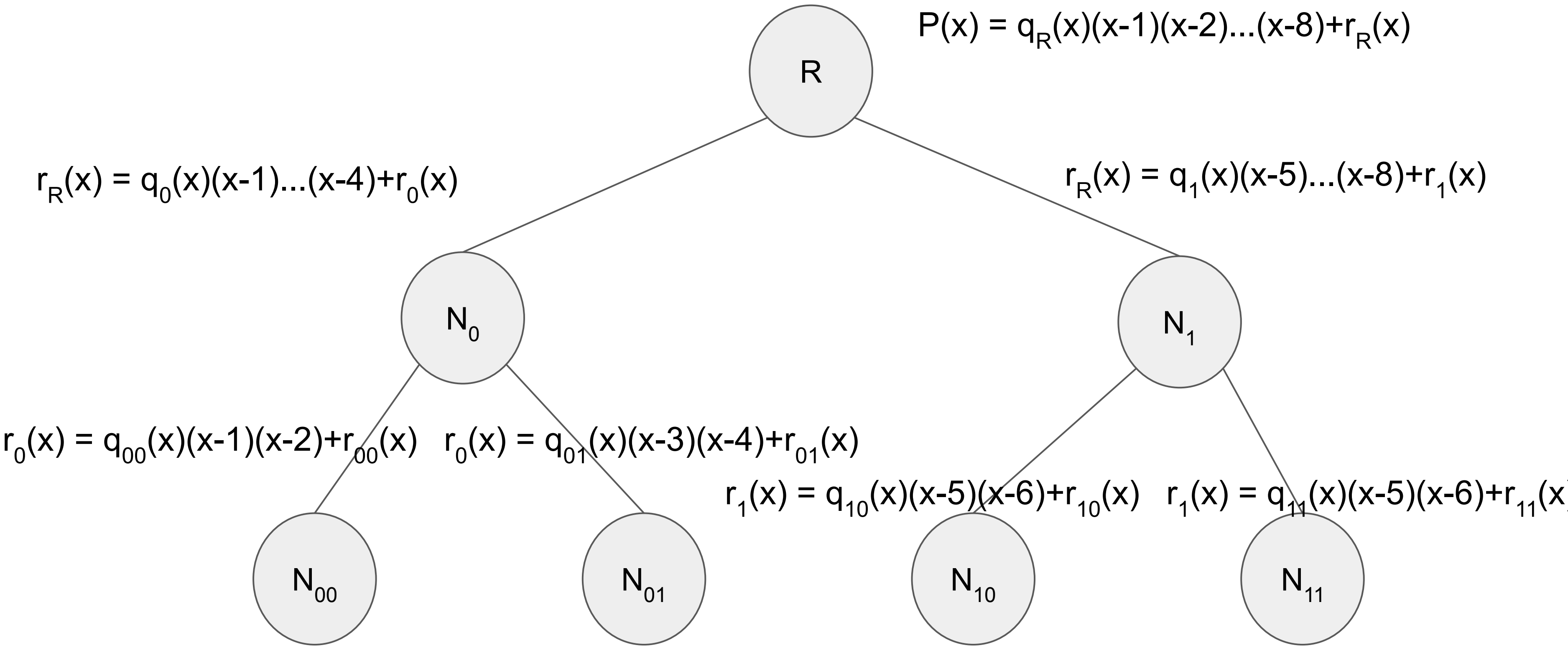


$P(x) = q_R(x)(x-1)(x-2)...(x-8)+r_R(x)$

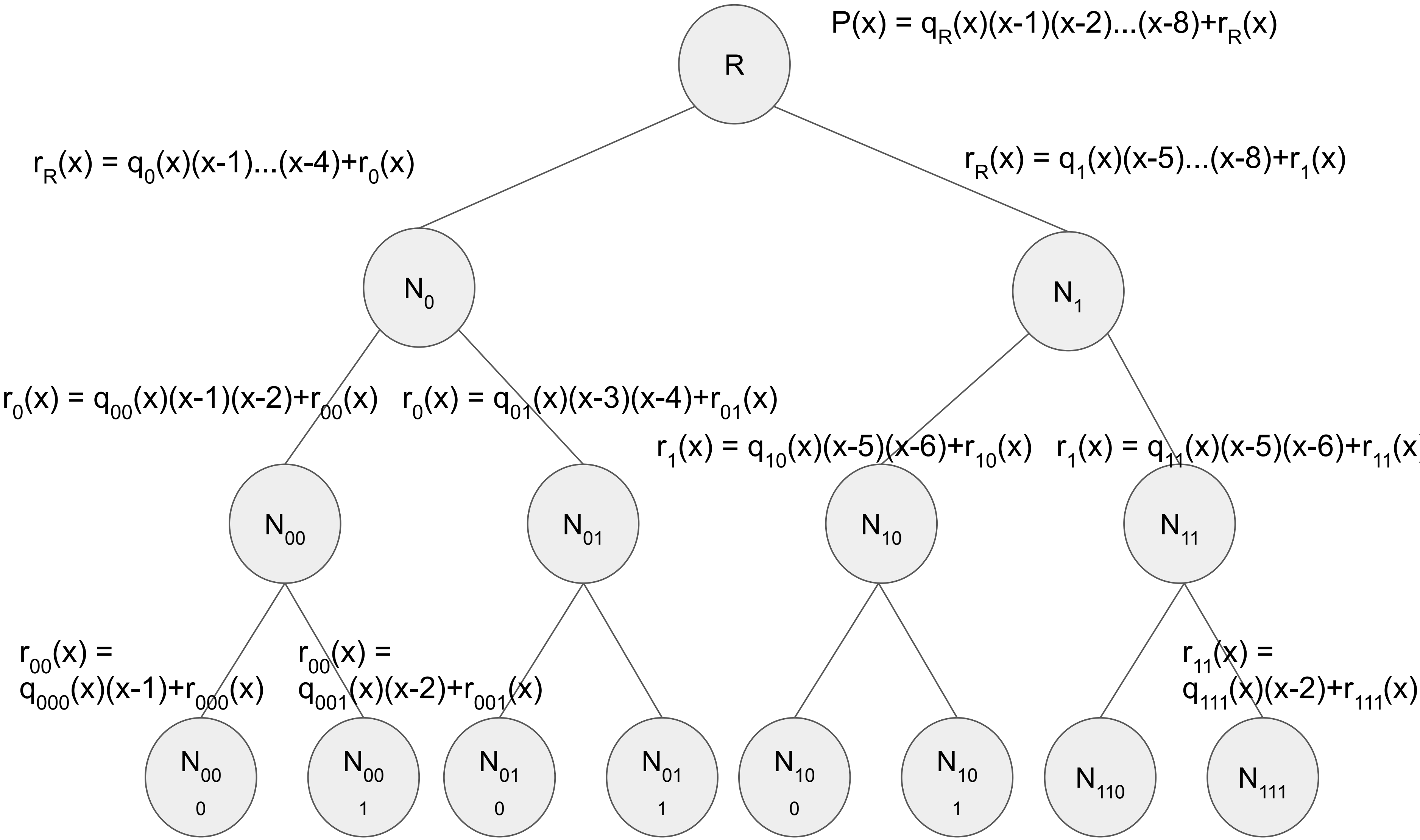
Multipoint Evaluation Trees



Multipoint Evaluation Trees



Multipoint Evaluation Trees

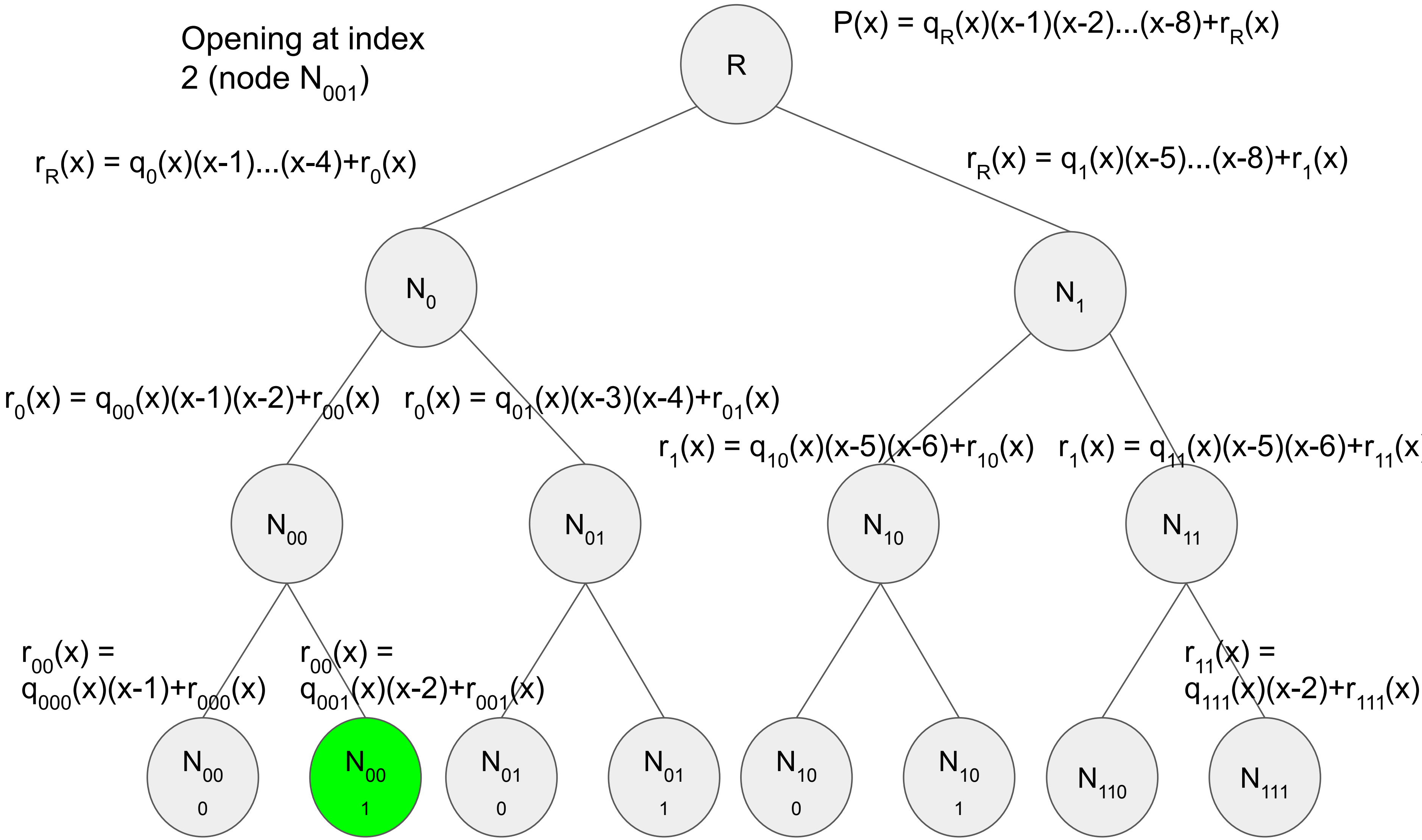


Commitment Proofs

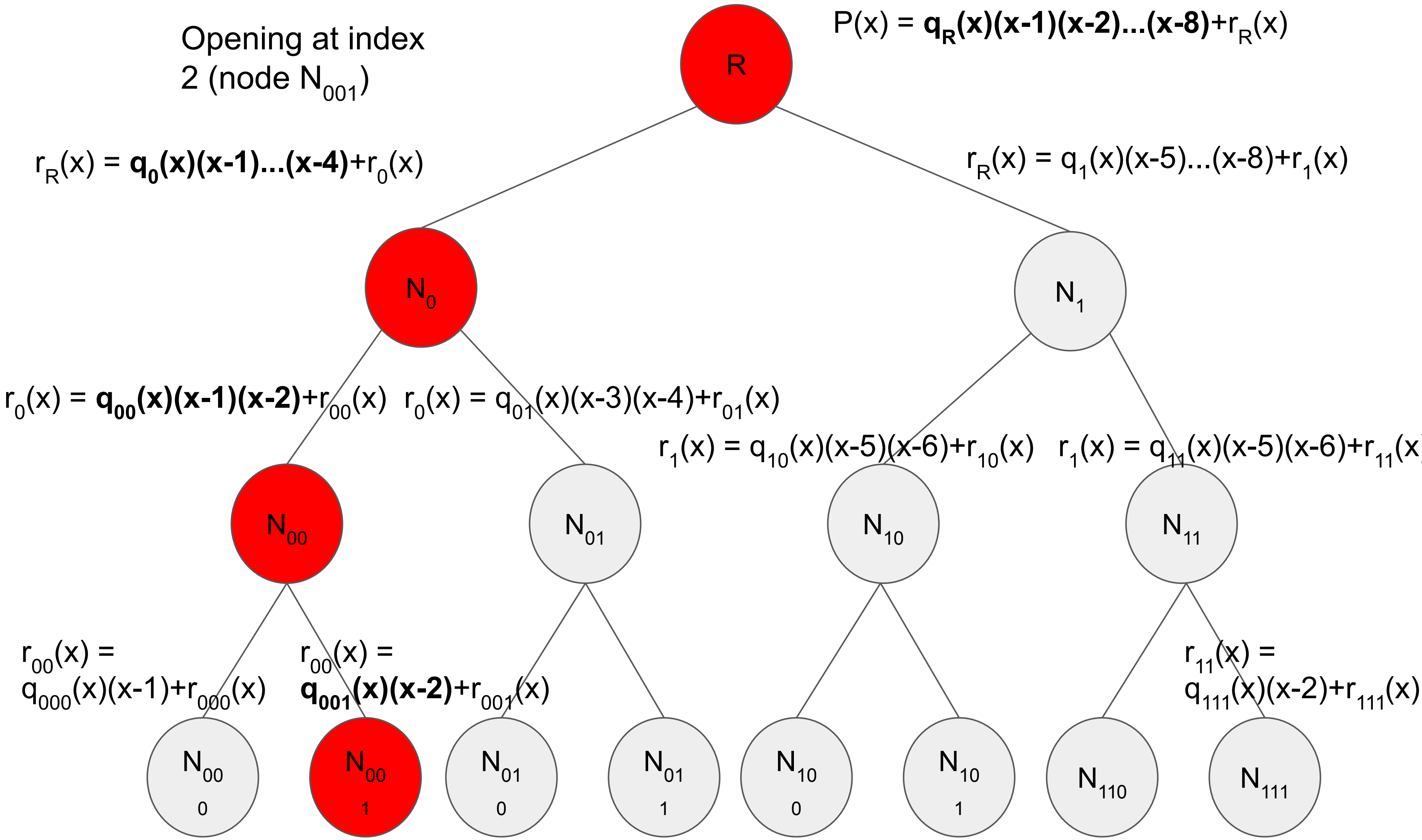
- The opening $\pi_{C,i}$ consists of the commitments to all the quotients and accumulators in the path from the root to the leaf corresponding to index i
- This is because the following equation

$$P(x) = P(i) + \sum_{\omega \in \mathrm{path}(i)} q_w(x) a_w(x)$$

Commitment Proofs



Commitment Proofs



Commitment Verification

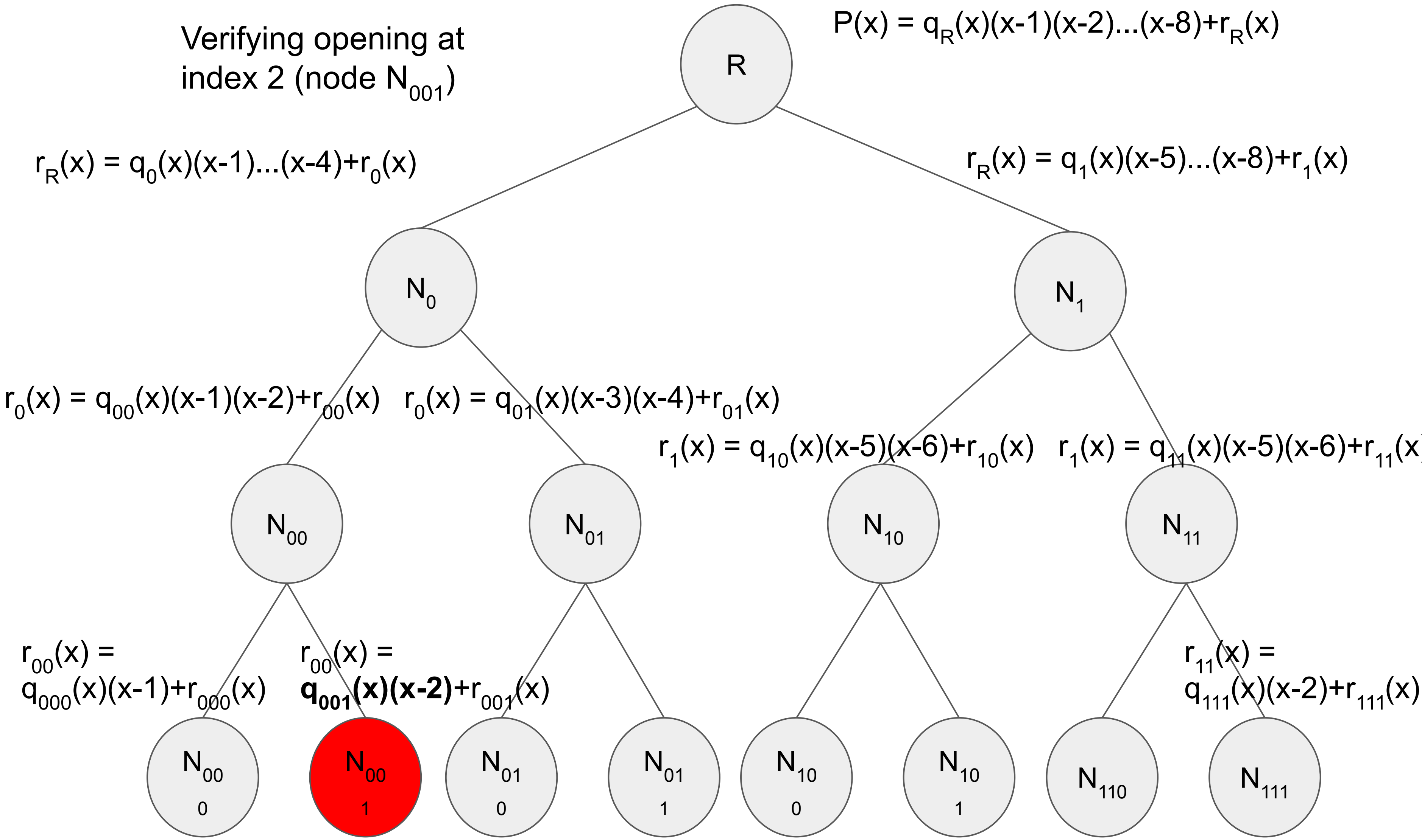
- To verify the correctness of the opening at index i given the opening $\pi_{C,i}$, use bilinear maps to check the opening equation is true "in the exponent"

$$e(g^{P(s)}, g) \stackrel{?}{=} e(g^{P(i)}, g) \cdot \prod_{\omega \in \text{path}(i)} e(g^{q_w(s)}, g^{a_w(s)})$$

Commitment Verification

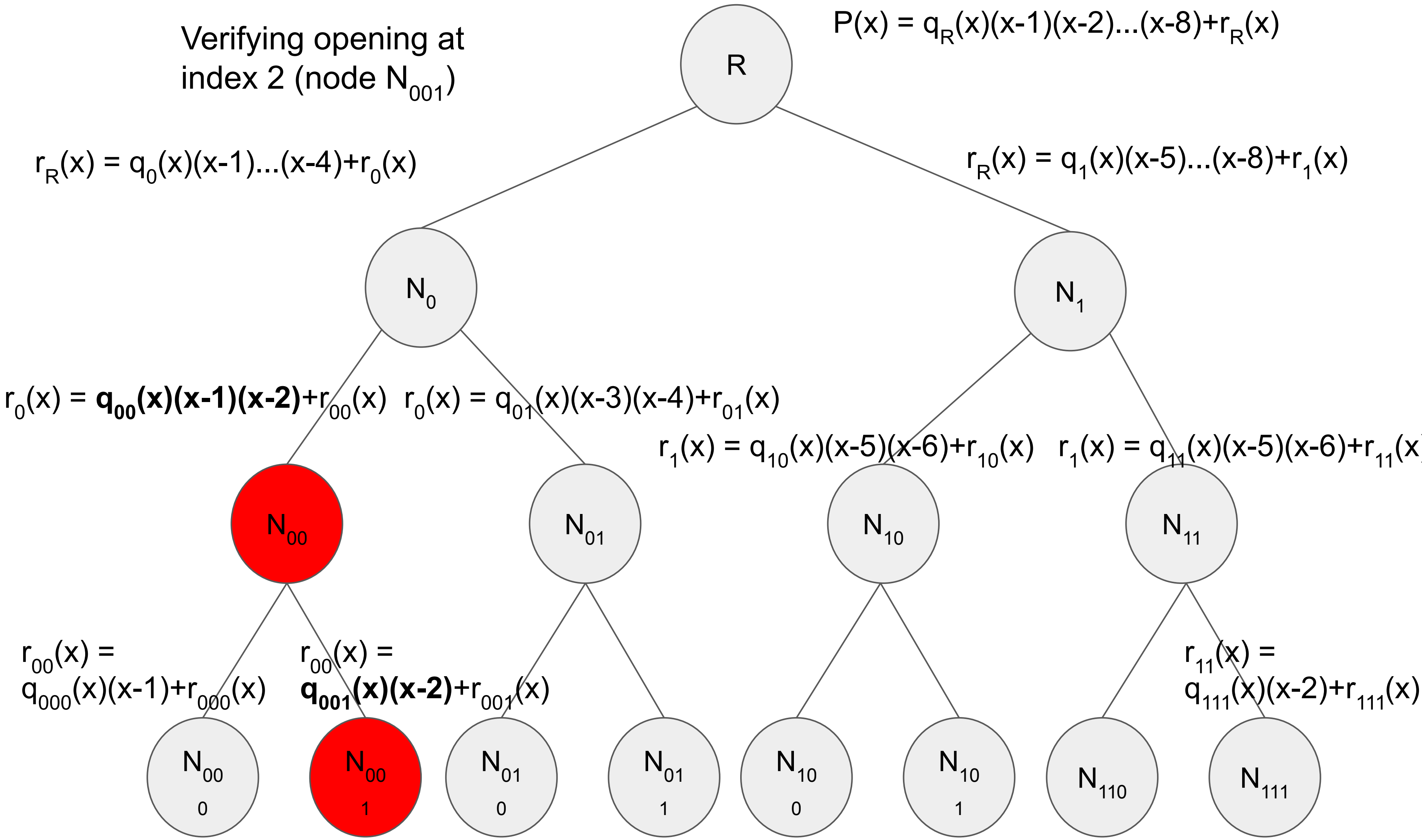
$$e(g^{P(s)}, g) \stackrel{?}{=} e(g^{P(i)}, g) \cdot \prod_{\omega \in \text{path}(i)} e(g^{q_w(s)}, g^{a_w(s)})$$

Verifying opening at index 2 (node N_{001})



Commitment Verification

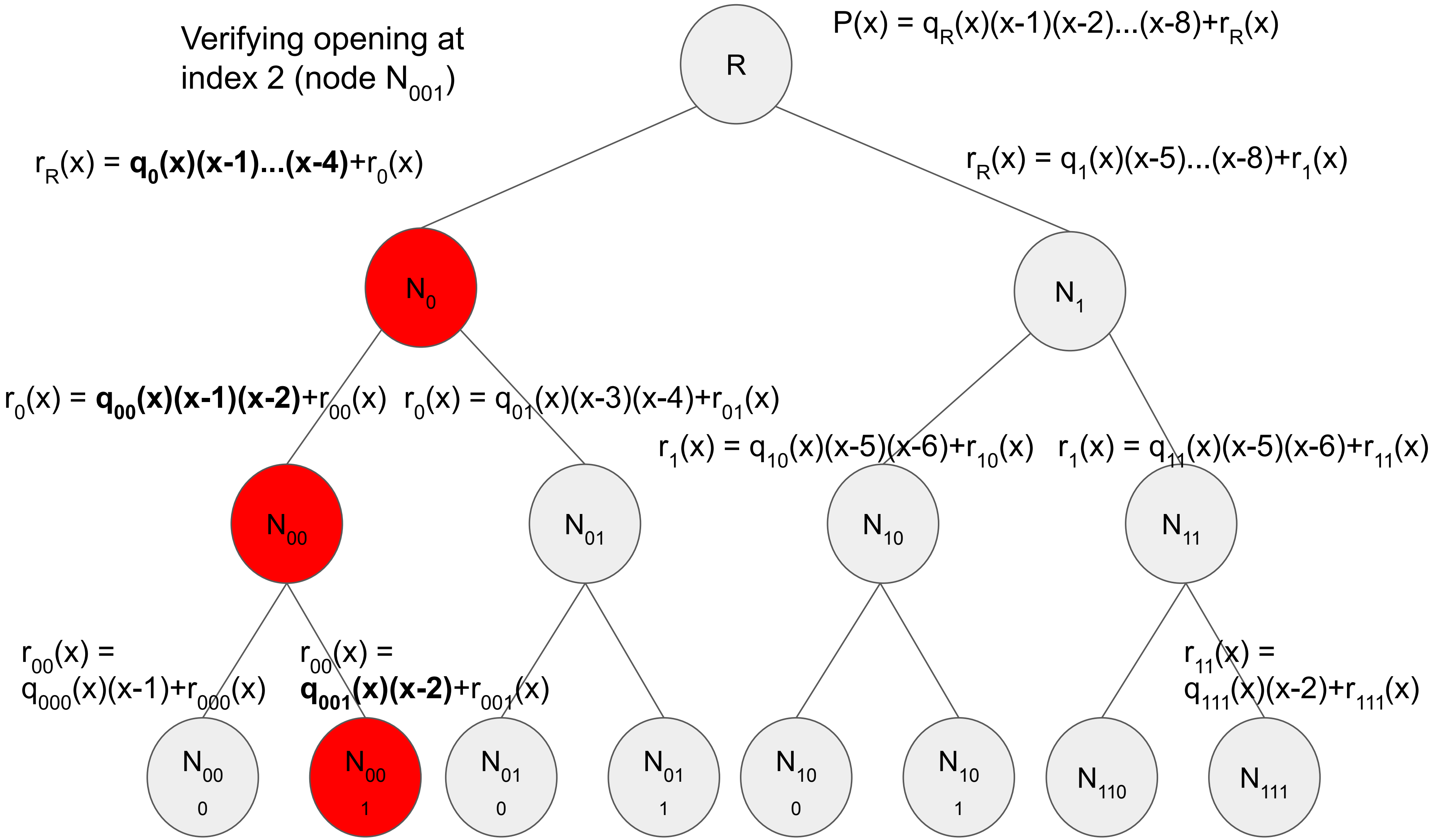
$$e(g^{P(s)}, g) \stackrel{?}{=} e(g^{P(i)}, g) \cdot \prod_{\omega \in \text{path}(i)} e(g^{q_w(s)}, g^{a_w(s)})$$



Commitment Verification

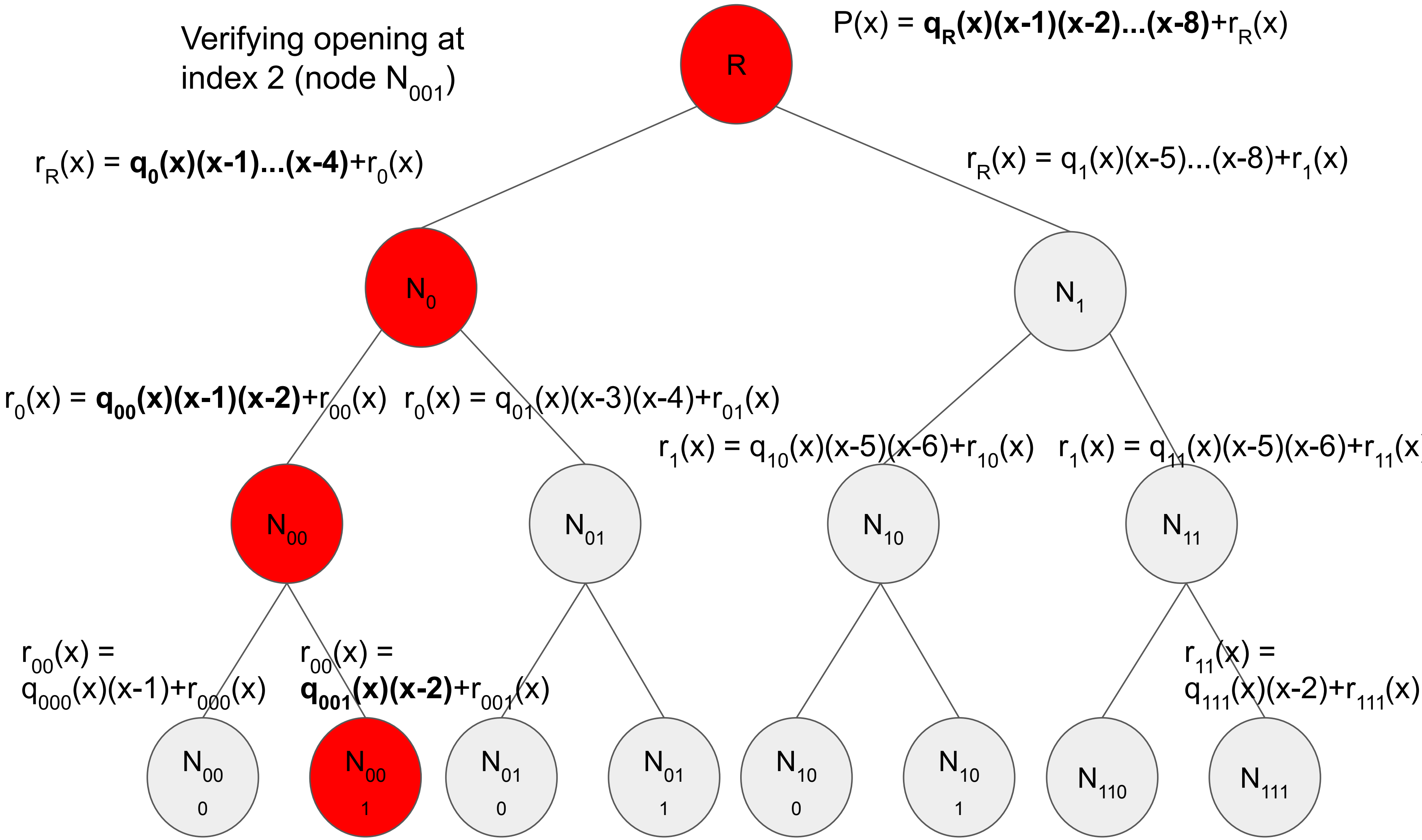
$$e(g^{P(s)}, g) \stackrel{?}{=} e(g^{P(i)}, g) \cdot \prod_{\omega \in \text{path}(i)} e(g^{q_w(s)}, g^{a_w(s)})$$

Verifying opening at index 2 (node N_{001})



Commitment Verification

$$e(g^{P(s)}, g) \stackrel{?}{=} e(g^{P(i)}, g) \cdot \prod_{\omega \in \mathrm{path}(i)} e(g^{q_w(s)}, g^{a_w(s)})$$



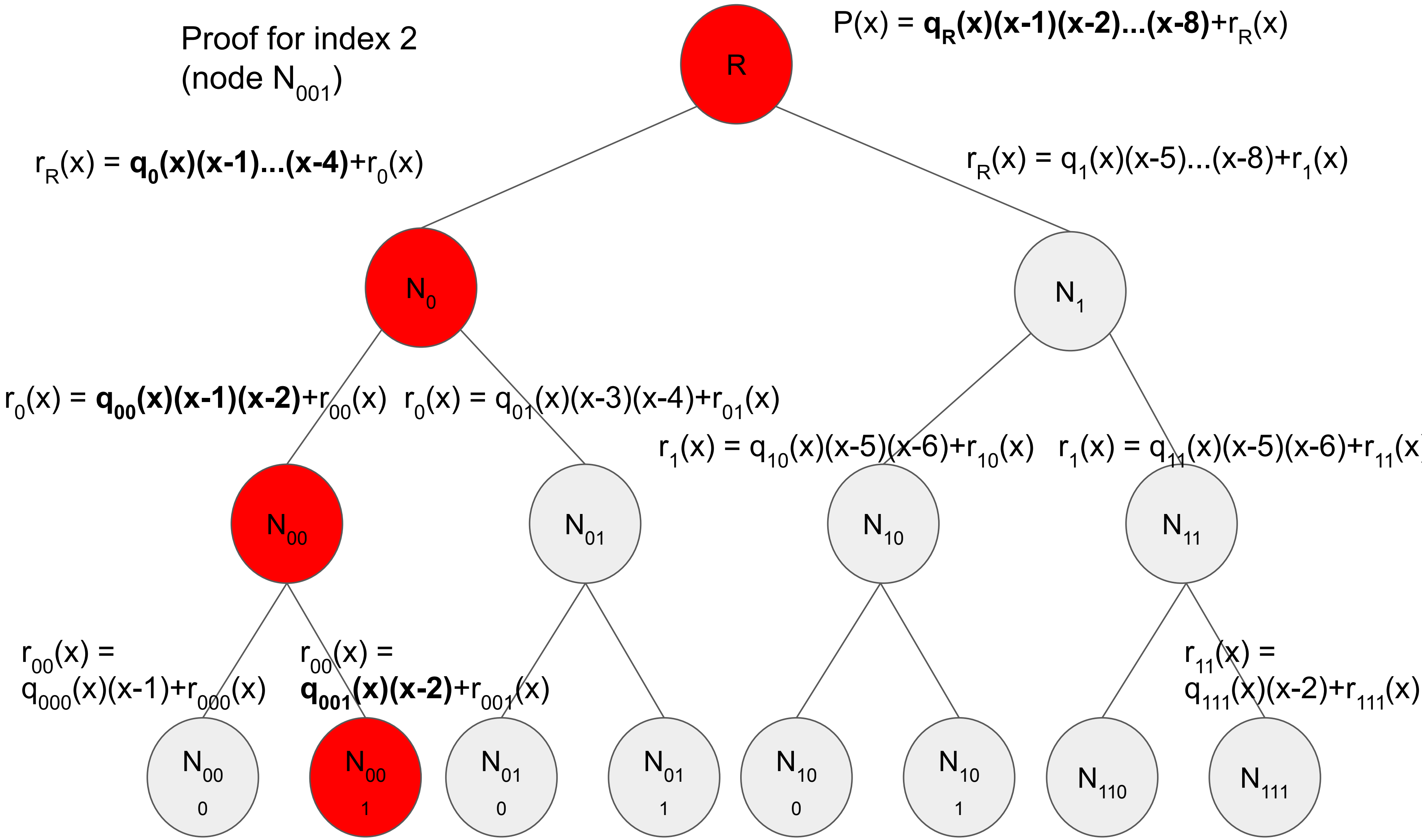
Updating Commitments & Proofs

- To update the commitment of a vector as it changes from $(a_1,a_2,...a_i,...,a_n) \rightarrow (a_1,a_2,..a_i',...a_n)$, compute

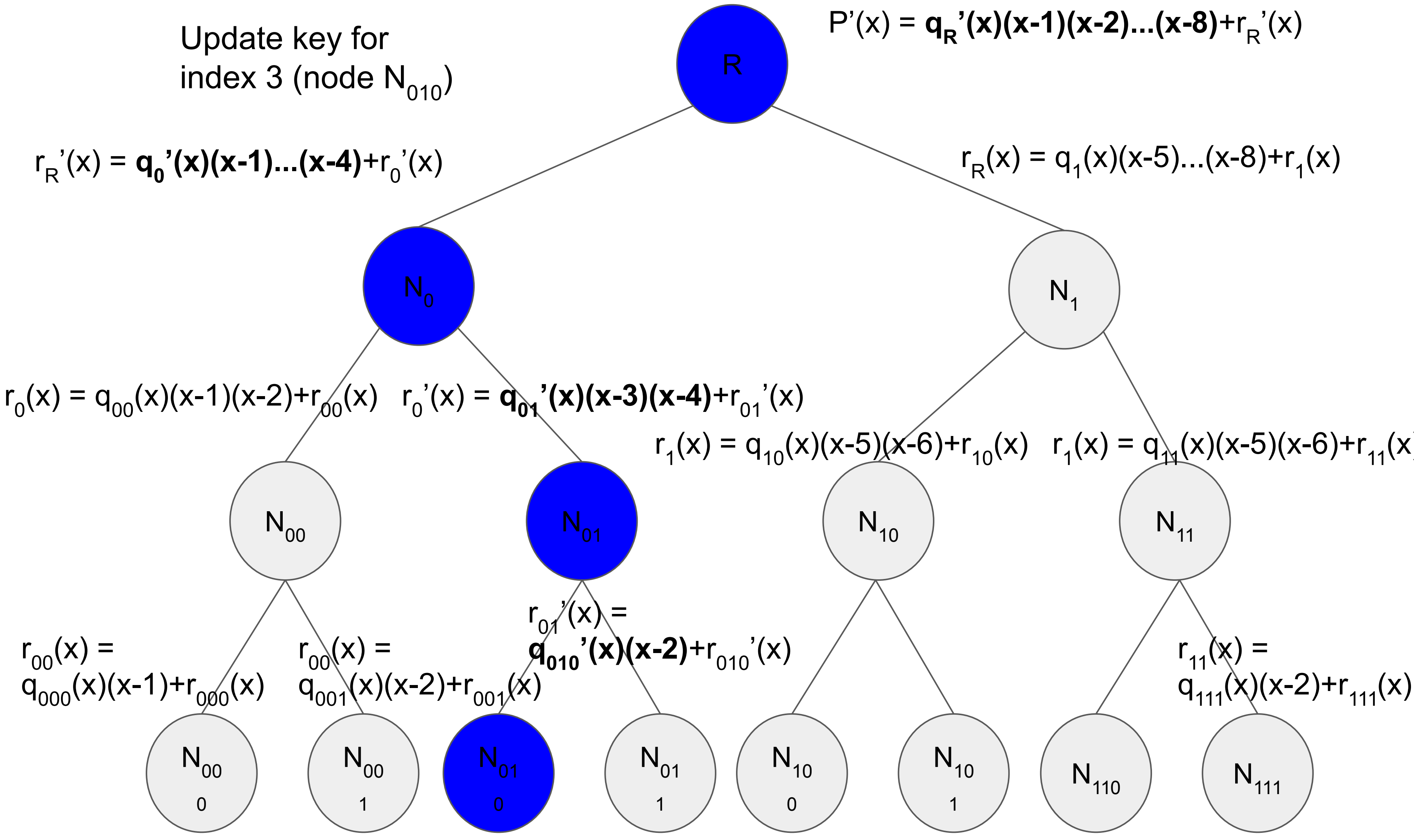
$$C' = C \cdot g^{L_i(s)\cdot(a_i'-a_i)}$$

- To update the proof $\pi_{C,j}$ when the vector changes at index i, we need the verification key consisting of commitments to all quotients in the path from the root to leaf i, which has size O(log n)

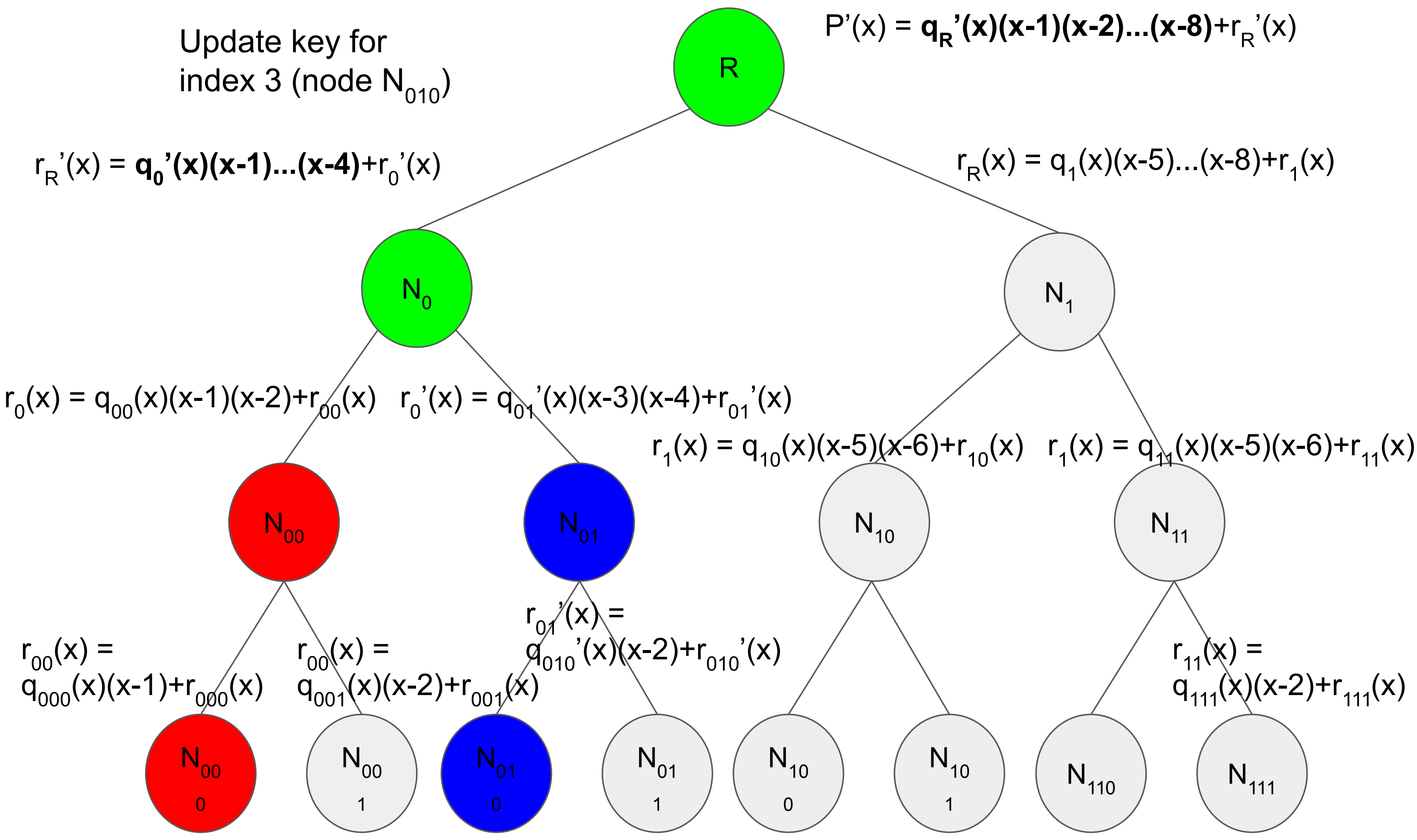
Updating Proofs



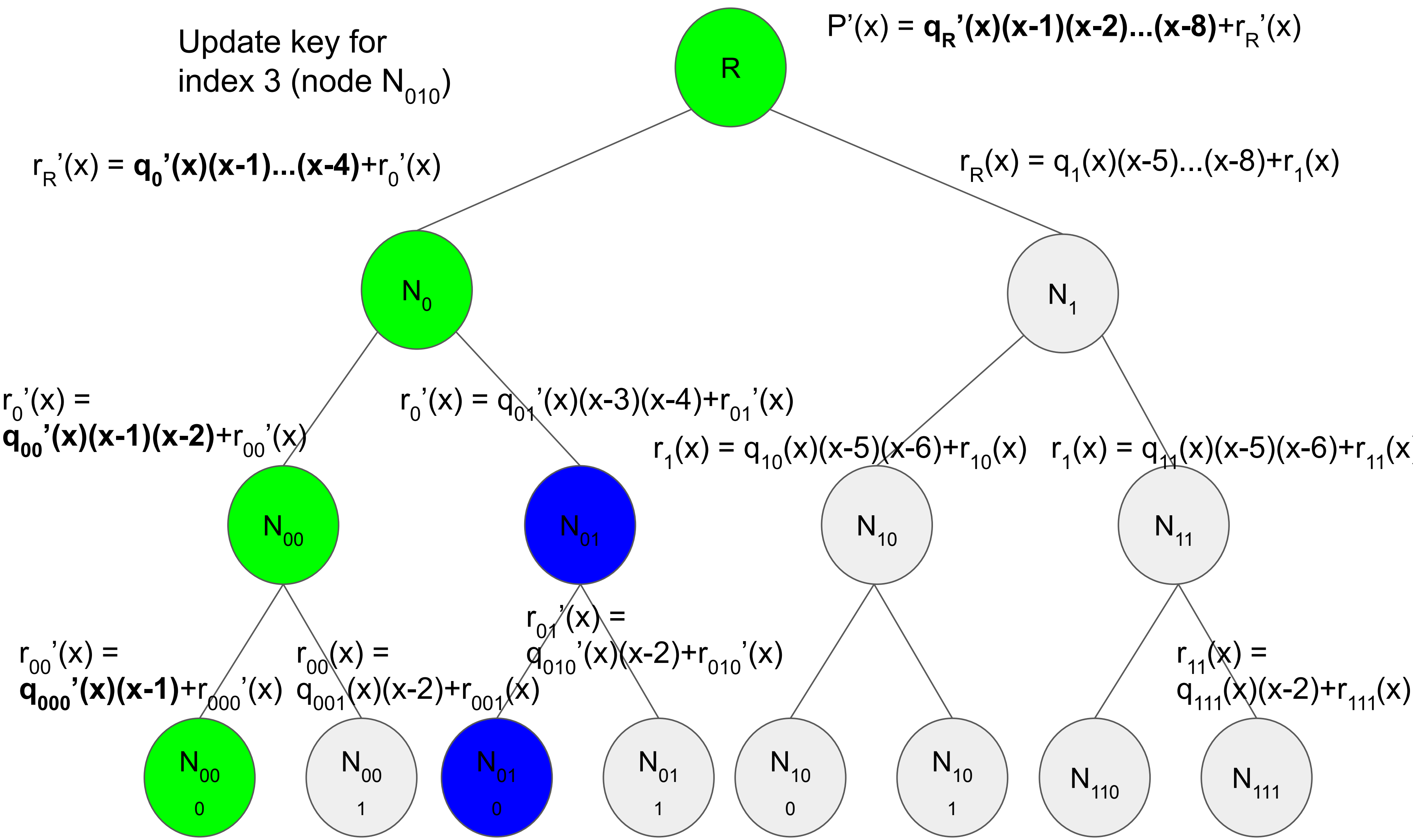
Updating Proofs



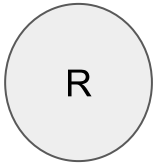
Updating Proofs



Updating Proofs

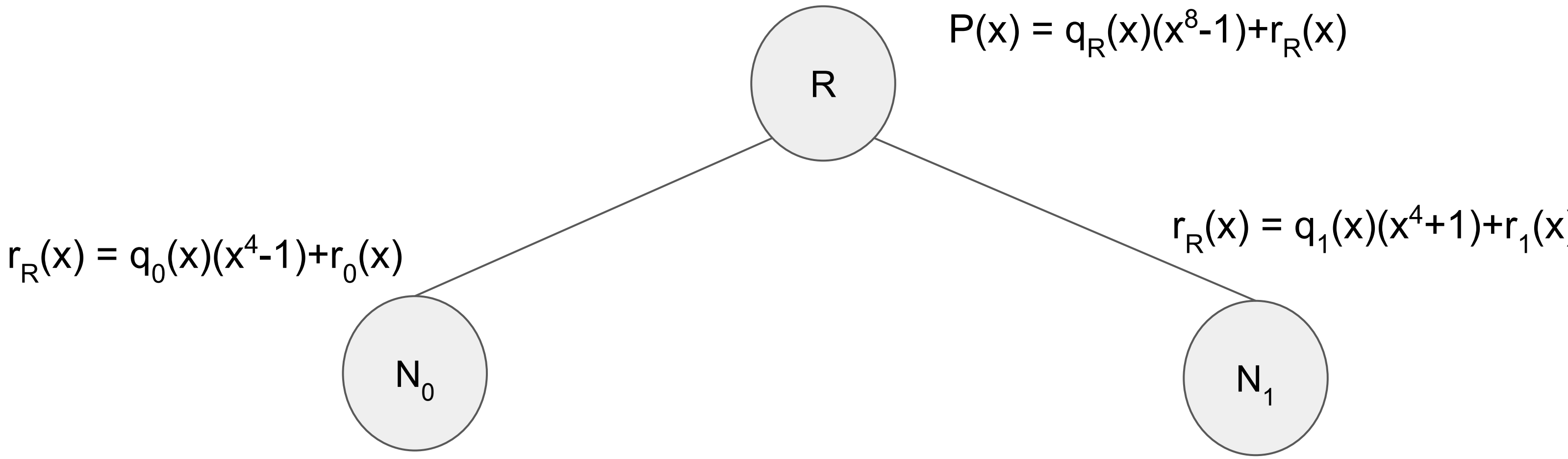


Multipoint Evaluation Trees with Roots of Unity

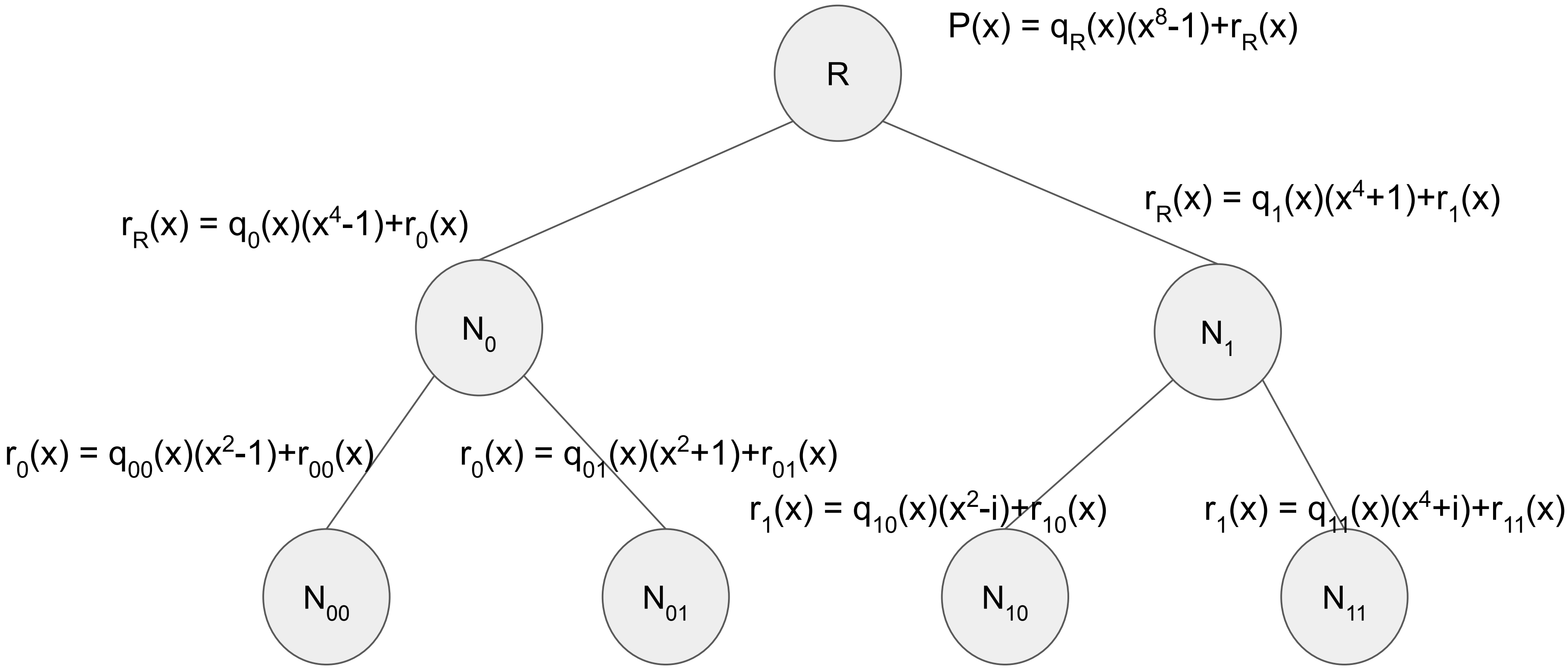


$P(x) = q_R(x)(x^8-1)+r_R(x)$

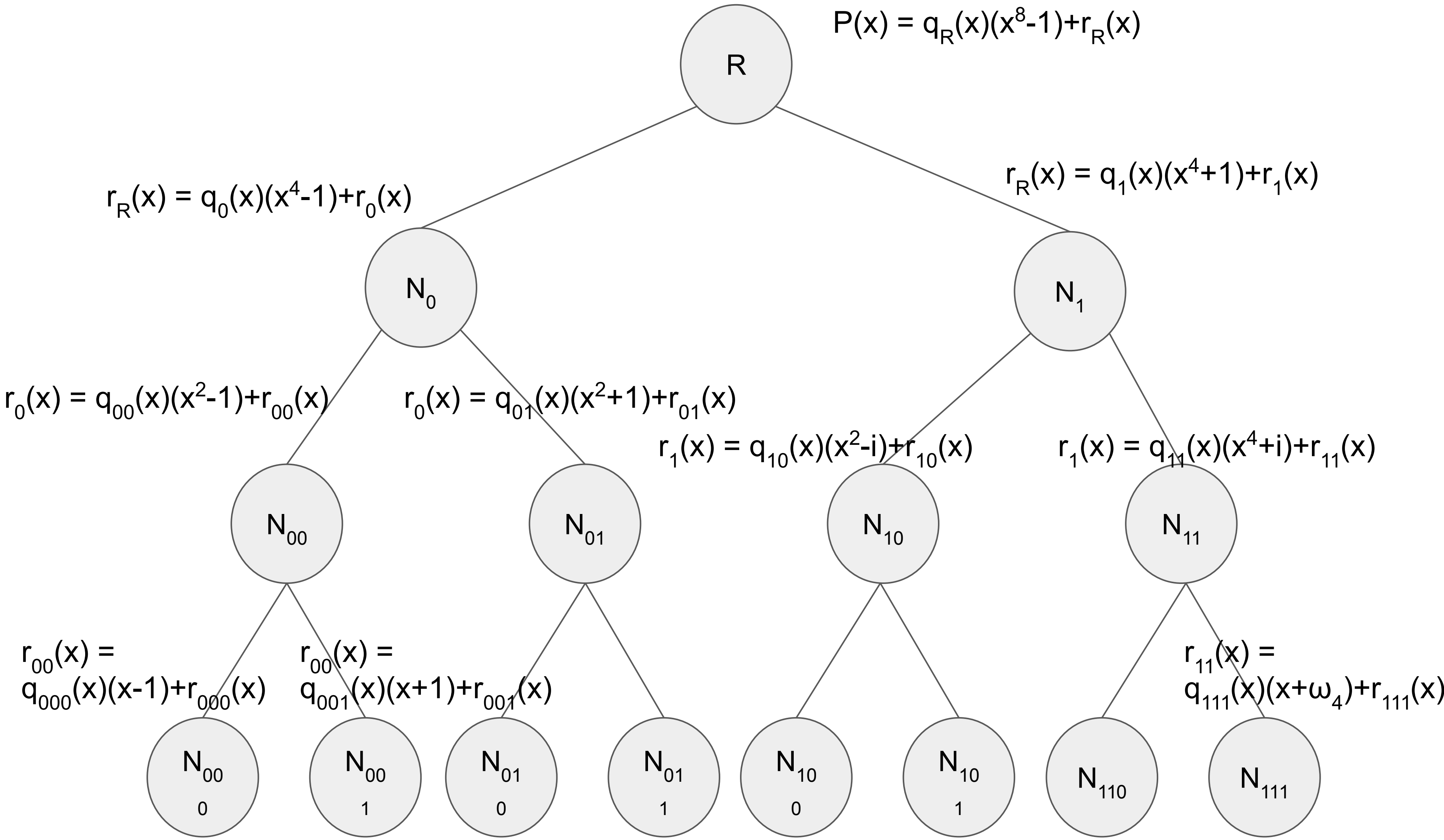
Multipoint Evaluation Trees with Roots of Unity



Multipoint Evaluation Trees with Roots of Unity



Multipoint Evaluation Trees with Roots of Unity



Summary

- Proof size: O(log n)
- Proof update time: O(log n)
- "Update key" size: O(log n)
- Public parameter size: O(n)

Summary

Scheme	*Proof size*	*Proof update time*	*Proof "update key" size*	*Precompute all proofs*	*Public parameters size*
Catalano & Fiore	1	1	**n**	**n^2**	**n^2**
Papamathou et al	log n	log n	log n	**n^2**	**n**
Our scheme	log n	log n	log n	**n log n**	**n**

Conclusion and Future Work

- A new VC scheme from univariate polynomials
- Lots of applications: VSS, DKG, stateless cryptocurrencies, etc.
- Future work: build an AAD with this VC scheme using "append-only proofs": given old VC and new VC, an append-only proof shows the new VC does not change any positions in the old VC