Indifferentiability of 6-round Luby-Rackoff

Mridul Nandi (ISI)

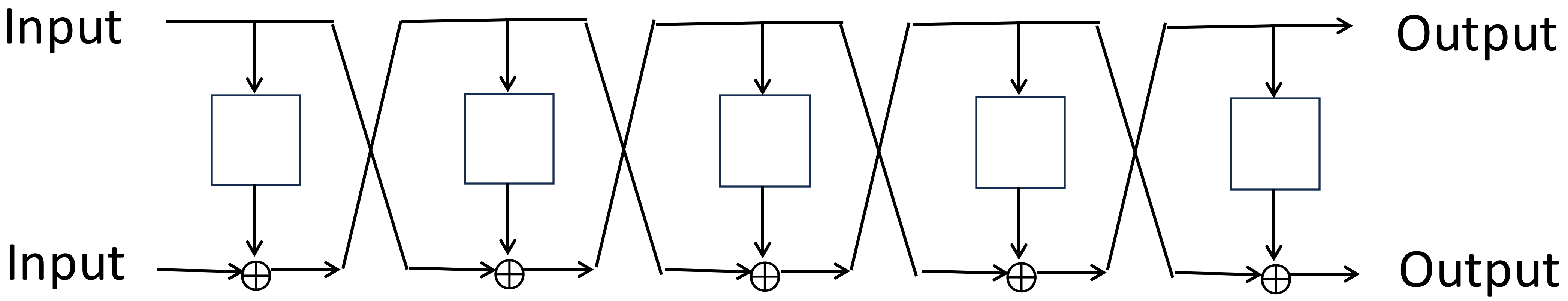
Joint work with

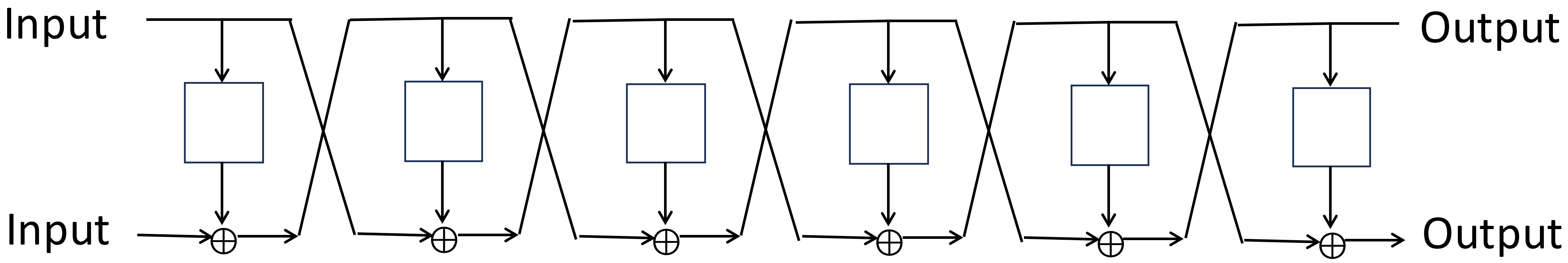
Ritam Bhaumik (TII), Ashwin Jha (RUB), Sayantan Pal (ISI) and Abishanka Saha (TU/e)

GAPS, NTU, Singapore

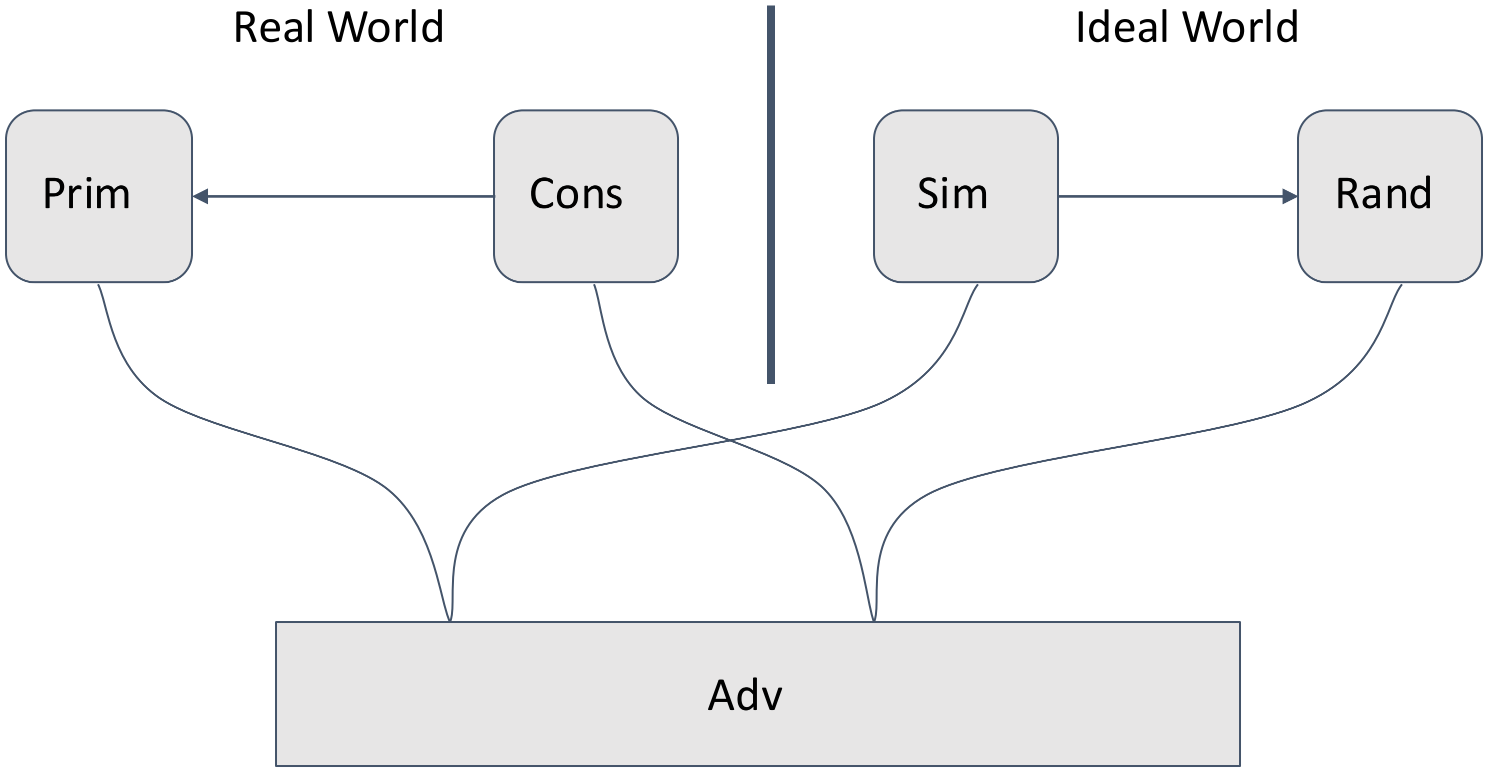
3rd September, 2025

5LR / 6LR

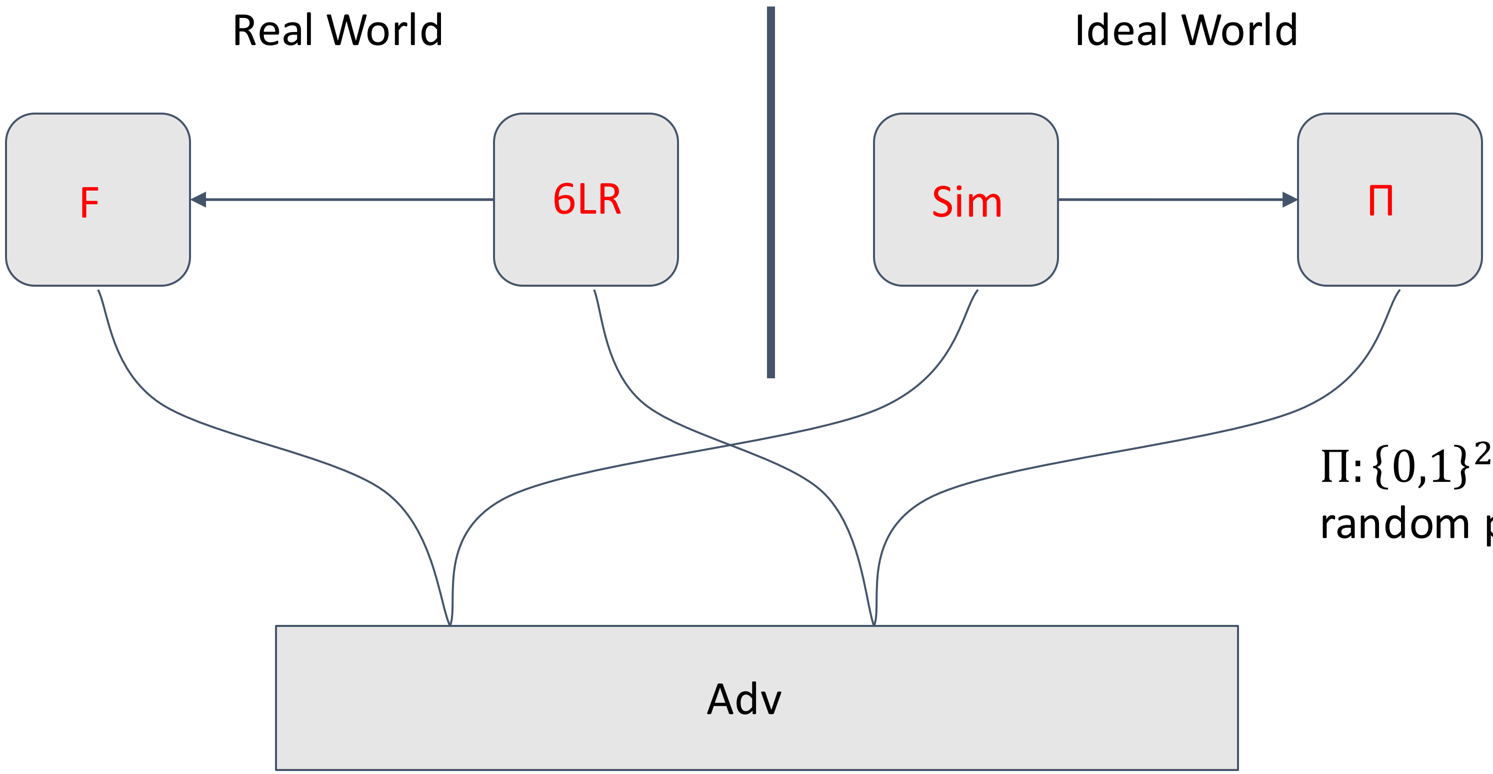




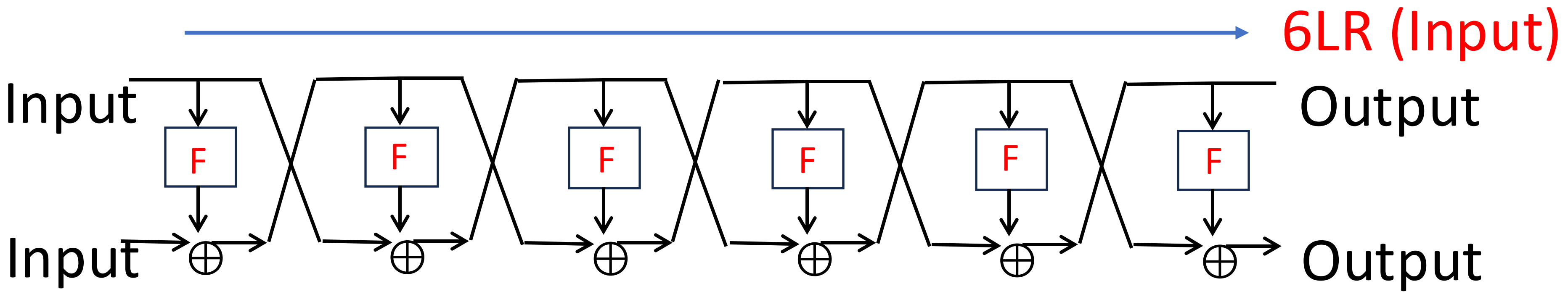
Indifferentiability Security Notion



Indifferentiability Security Notion



$\Pi: \{0,1\}^{2n} \rightarrow \{0,1\}^{2n}$
random permutation



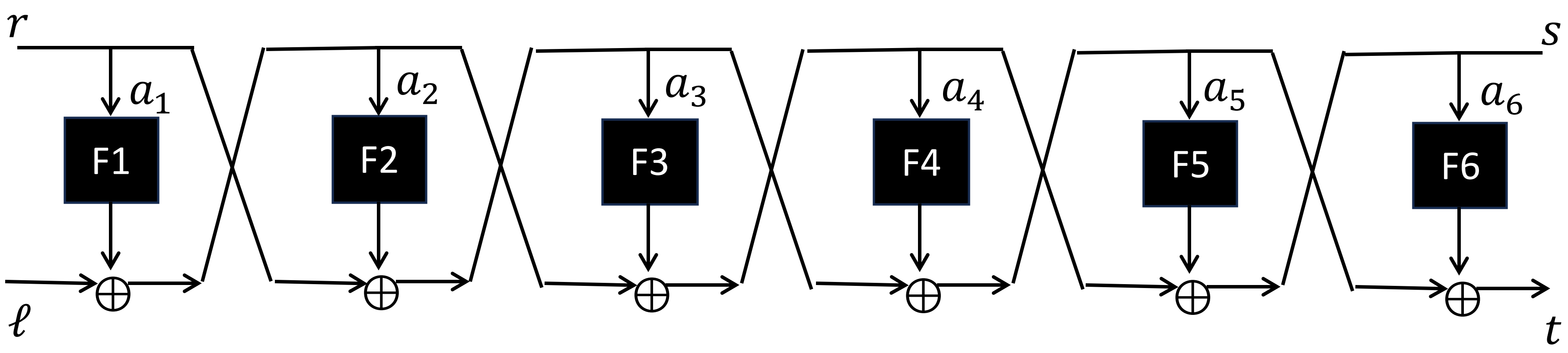
History

- **5 Rounds (Insecure)**: Coron, Patarin, and Seurin in Crypto 2008
- **6 Rounds**: Coron, Patarin, and Seurin in Crypto 2008
- **Issue and 14 Rounds**: Holenstein, Kunzler, Tessaro in STOC 2011.
- **10 Rounds**: Dachman-Soled, Katz, Thiruvengadam in Eurocrypt 2016
- **8 Rounds**: Dai, Stenberger CRYPTO 2016

Basic Proof Steps

- Simulator must behave like random (so are outputs of F1,… F6).

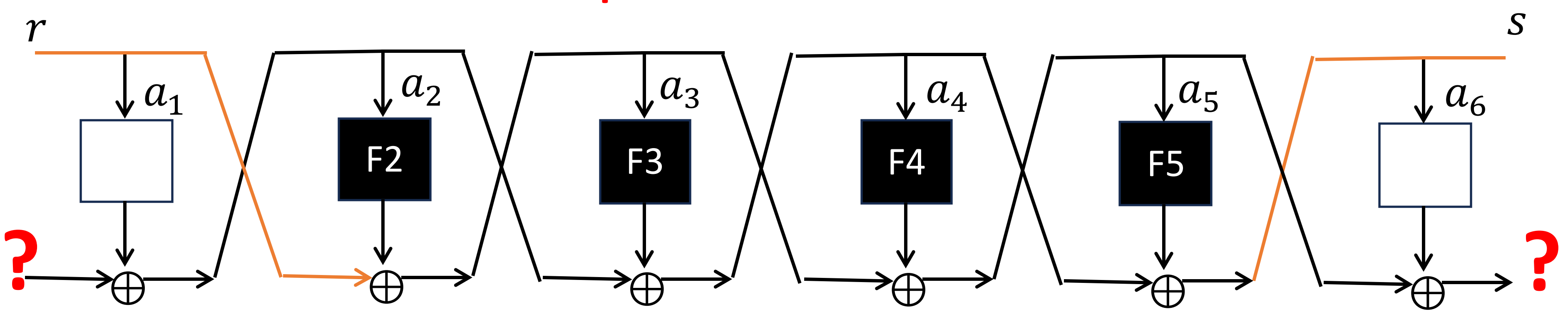
- Assume, after all queries, additional primitive queries made so that construction queries $6LR(\ell, r) = (s, t)$ can be computed completely.



- Ideal World: we must have same relation

$$6LR^{Sim}(\ell, r) = \Pi(\ell, r)$$

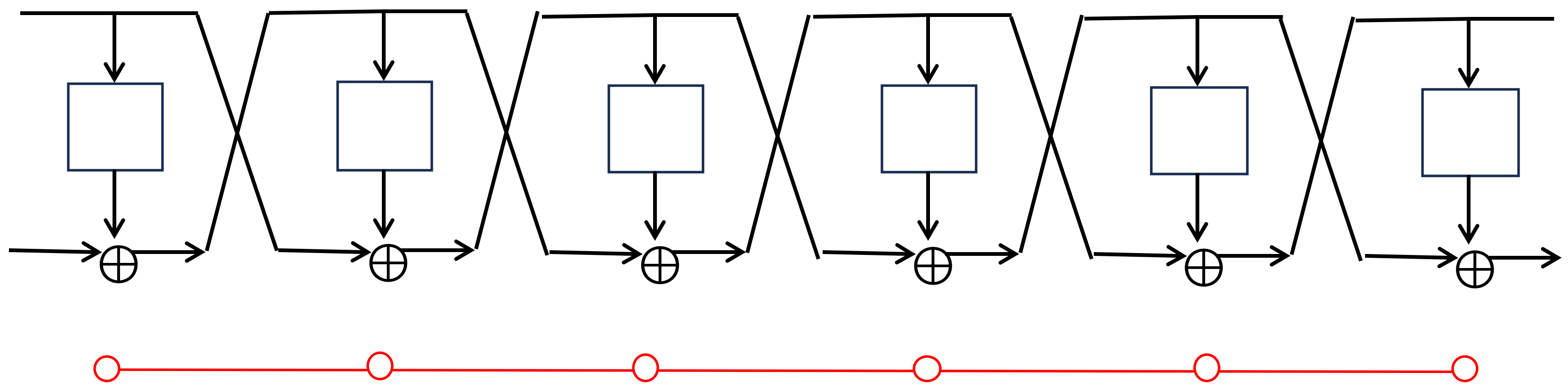
Basic Proof Steps



- If Sim returns four consecutive compatible F queries without taking help of Construction Oracle then it needs to find ℓ, t such that

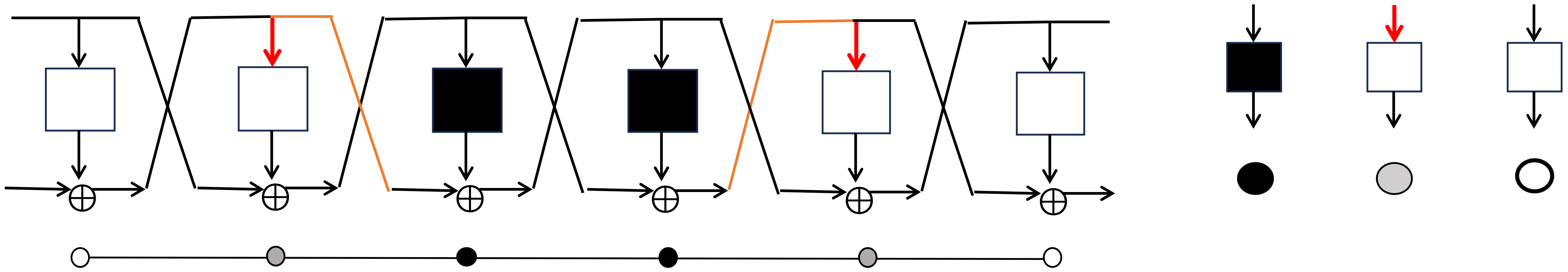
$$\Pi(\ell, r) = (s, t)$$

- Hard. So, Sim must
  - check all such consecutive F queries before and
  - pre-emptively it should make construction oracle queries to make it consistent.



Line System of a Transcript (P, F)

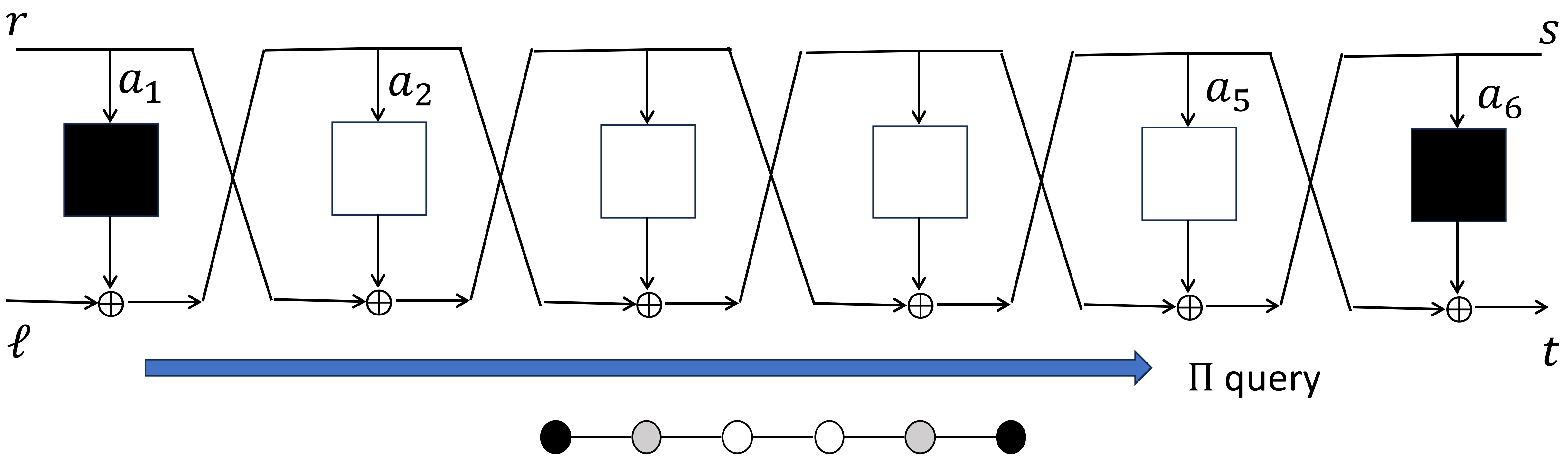
Line Representation of LR Computation



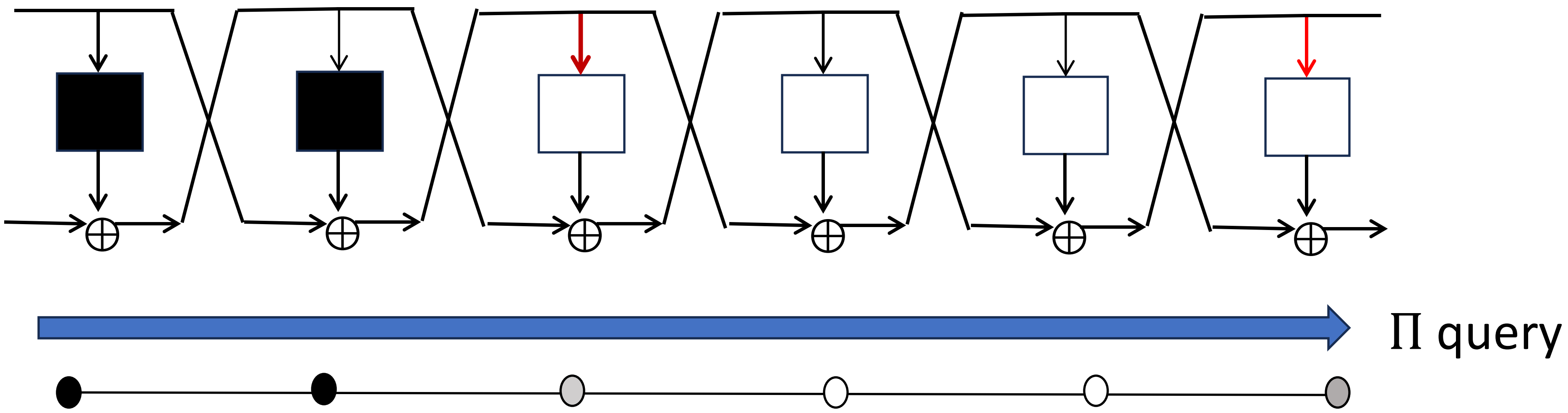
❑ Line Representation of LR Computation

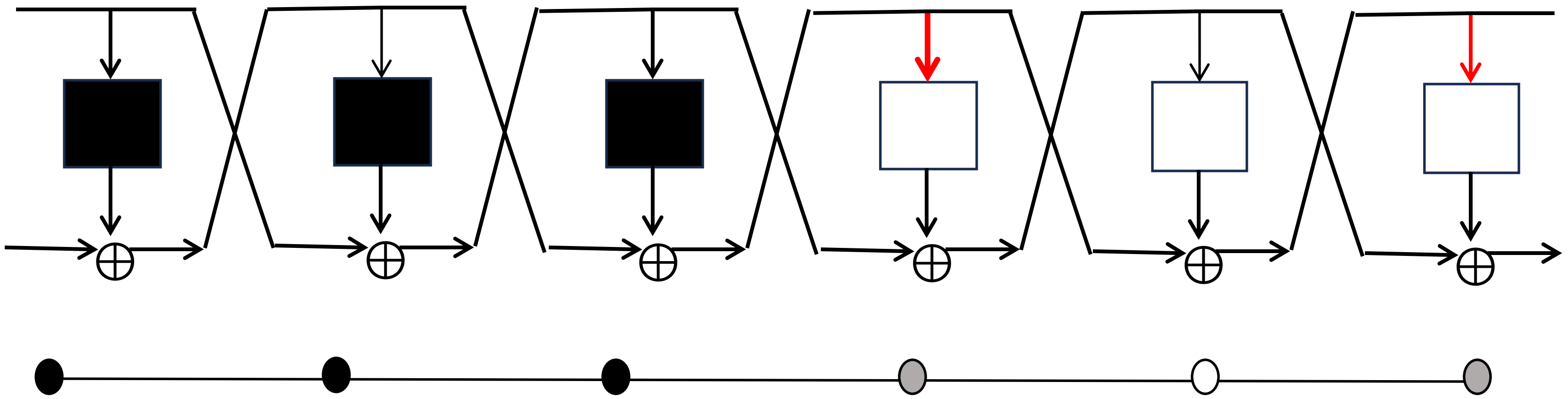
Construction Query: $6LR(\ell, r) = (s, t)$

Primitive Query : $F(r \ := \ a_1) = b_1, \ F(s = \ a_6) = b_6$

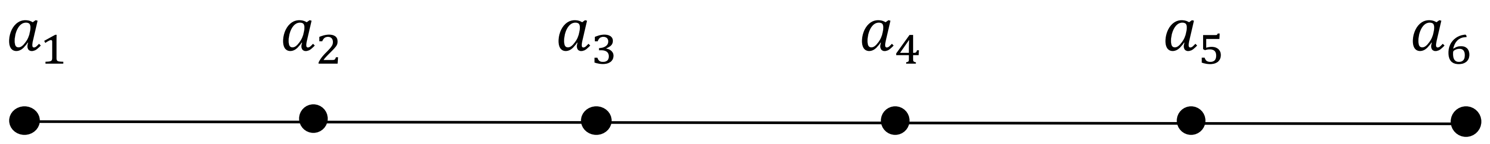


❑ Line Representation of LR Computation

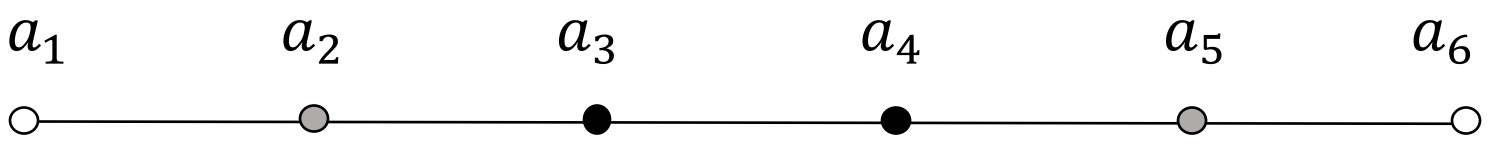




Relationship among input and outputs of F in LR

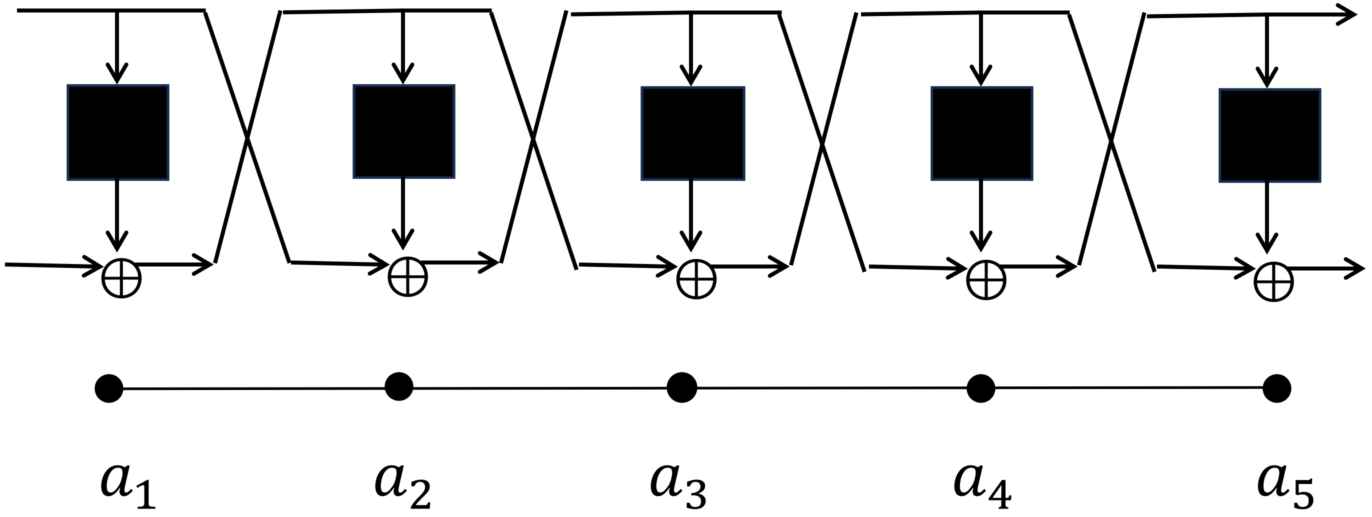


$$F(a_2) = a_1 + a_3, F(a_3) = a_2 + a_4, F(a_4) = a_3 + a_5, F(a_5) = a_4 + a_6$$
$$\Pi(F(a_1) + a_2, a_1) = (a_6, F(a_6) + a_5)$$



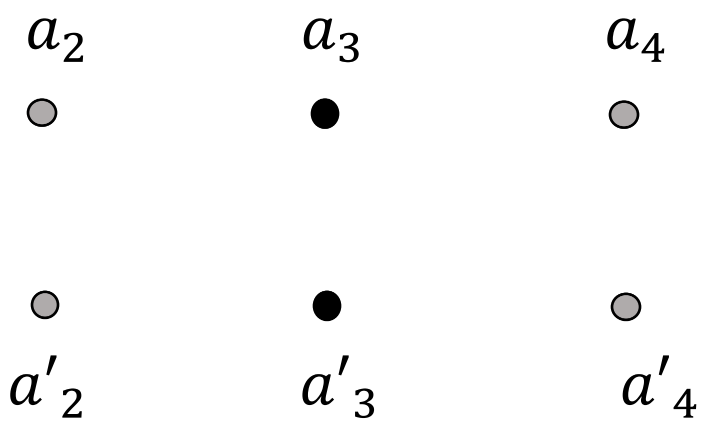
$F(a_3) = a_2 + a_4, \quad F(a_4) = a_3 + a_5, \quad a_1, a_6$ unknown

Insecurity of 5LR (Holenstein et al.)



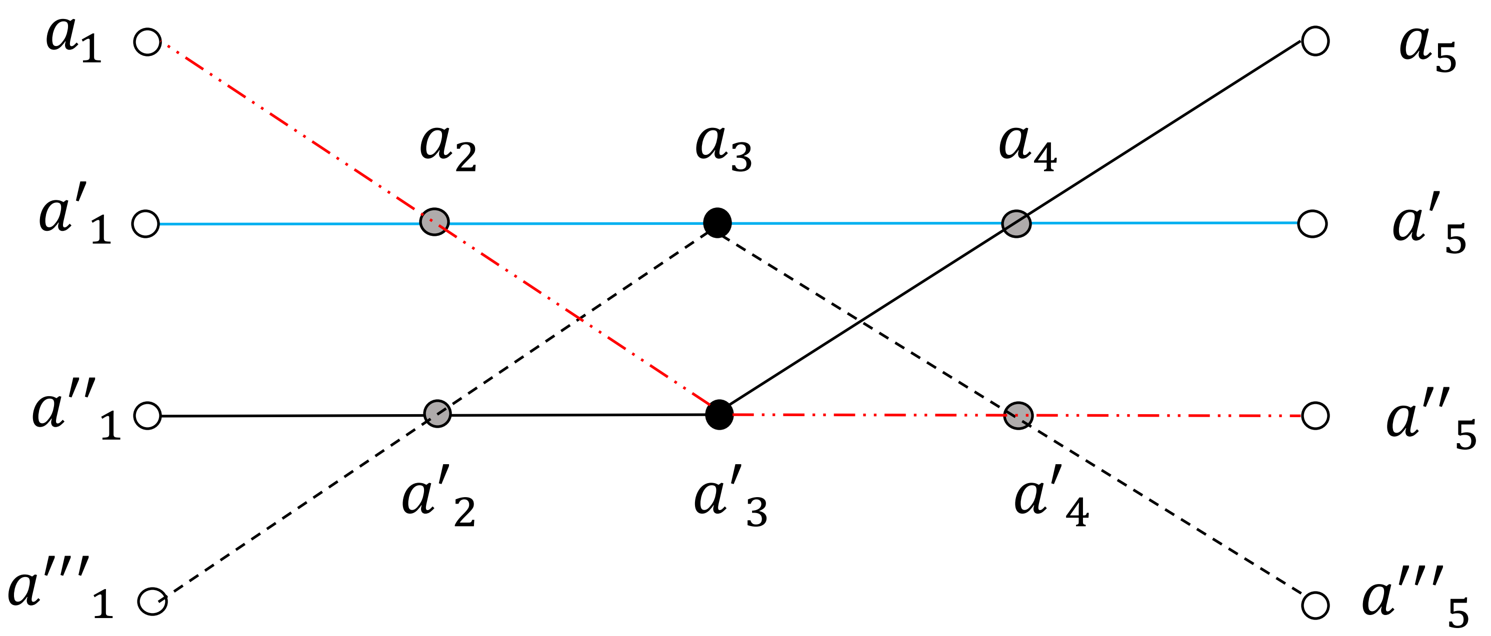
$$F(a_2) = a_1 + a_3, \qquad F(a_3) = a_2 + a_4, \qquad F(a_4) = a_3 + a_5$$

Insecurity of 5LR (Holenstein et al.)



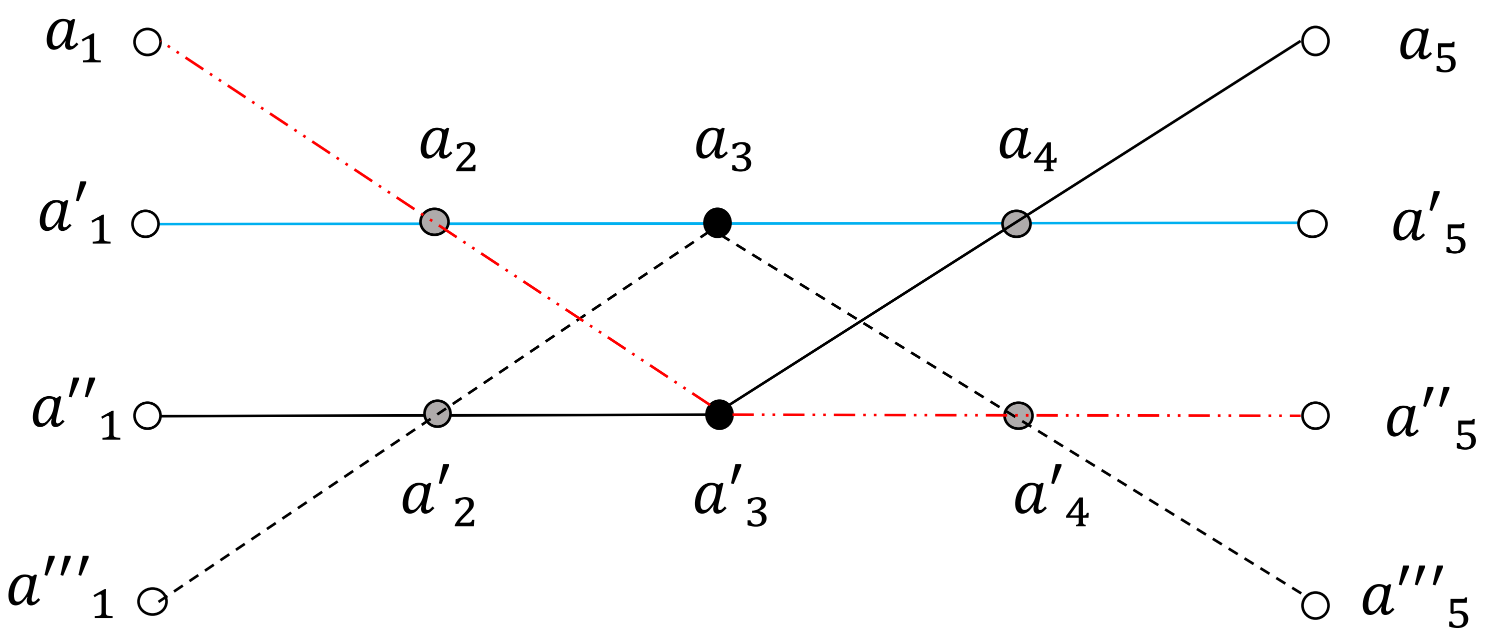
- Make Primitive queries $a_3,\ a_3'$ and responses $b_3,\ b_3'$
- Choose a_2 random.
- Define $a_4 = a_2 \oplus b_3,\ a'_4 = a_2 \oplus b'_3,\ a'_2 = a_4 \oplus b'_3 \Rightarrow a'_2 \oplus a'_4 = b_3$

Insecurity of 5LR (Holenstein et al.)



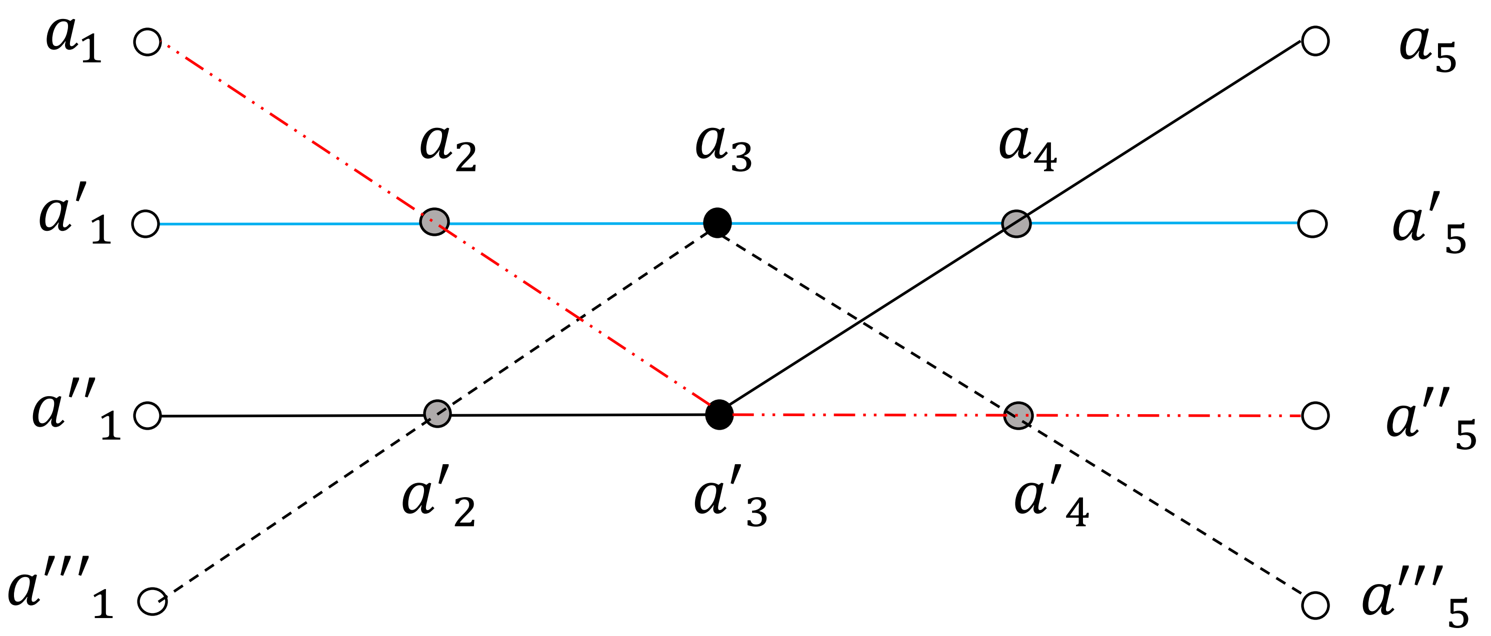
- Make Primitive queries a_3, a_3' and responses b_3, b_3'
- Choose a_2 random.
- Define $\underline{a_4 = a_2 \oplus b_3}$, $\underline{a'_4 = a_2 \oplus b'_3}$, $\underline{a'_2 = a_4 \oplus b'_3} \Rightarrow \underline{a'_2 \oplus a'_4 = b_3}$

Insecurity of 5LR (Holenstein et al.)



$$\left.\begin{array}{l} 5LR(*, a_1) = (a_5, *) \\ 5LR(*, a'_1) = (a'_5, *) \\ 5LR(*, a''_1) = (a''_5, *) \\ 5LR(*, a'''_1) = (a'''_5, *) \end{array}\right\} \Rightarrow \begin{array}{l} a_1 \oplus a'_1 \oplus a''_1 \oplus a'''_1 = 0 \\ a_5 \oplus a'_5 \oplus a''_5 \oplus a'''_5 = 0 \end{array}$$

Insecurity of 5LR (Holenstein et al.)



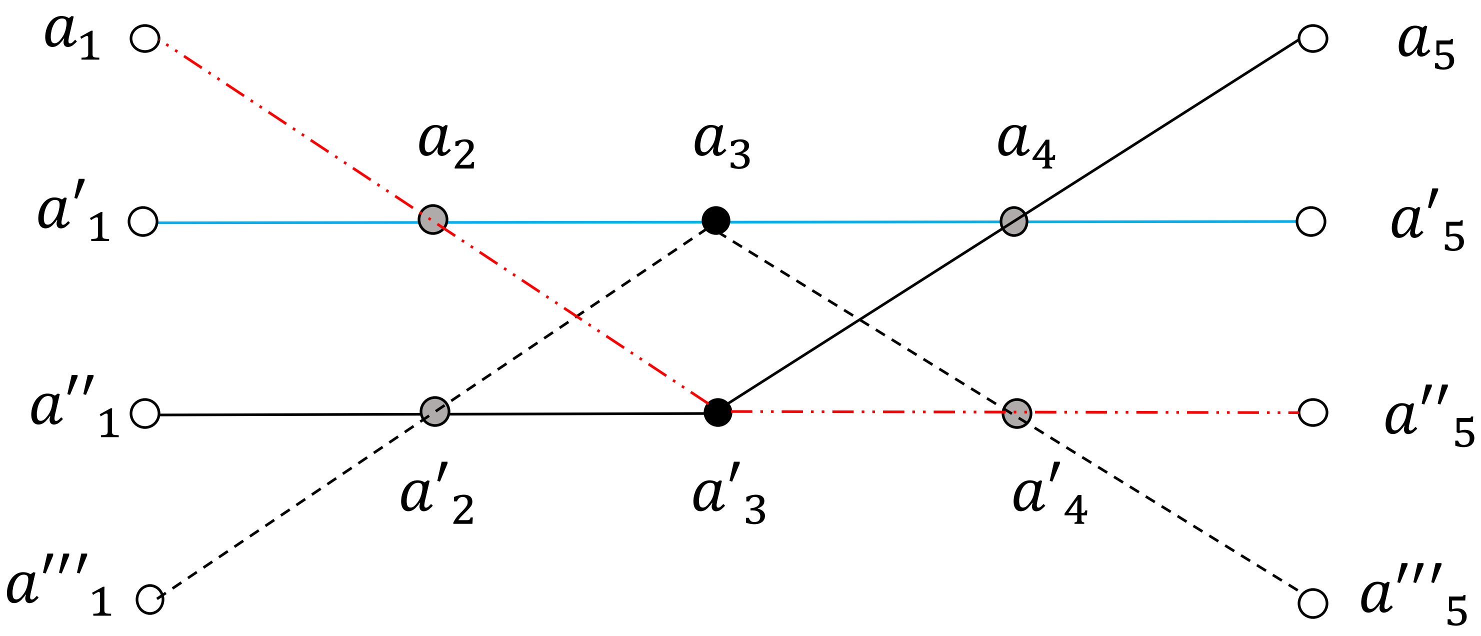
Hard !!

$$5LR(*, a_1) = (a_5, *)$$
$$5LR(*, a'_1) = (a'_5, *)$$
$$5LR(*, a''_1) = (a''_5, *)$$
$$5LR(*, a'''_1) = (a'''_5, *)$$

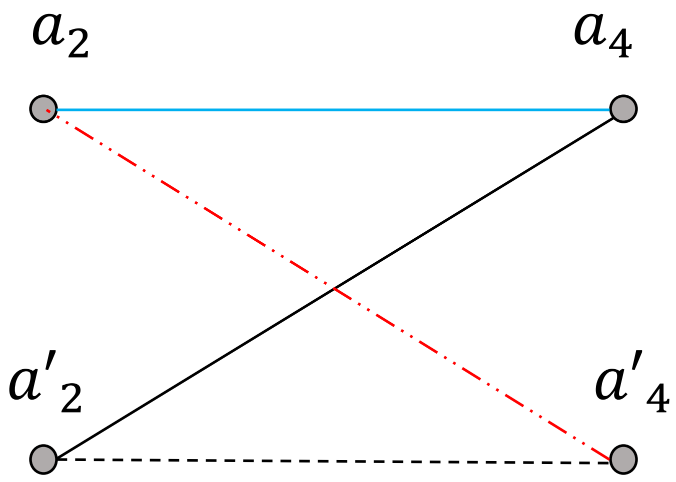
$$a_1 \oplus a'_1 \oplus a''_1 \oplus a'''_1 = 0$$
$$a_5 \oplus a'_5 \oplus a''_5 \oplus a'''_5 = 0$$

$$\Pi(*, a_1) = (a_5, *)$$
$$\Pi(*, a'_1) = (a'_5, *)$$
$$\Pi(*, a''_1) = (a''_5, *)$$
$$\Pi(*, a'''_1) = (a'''_5, *)$$

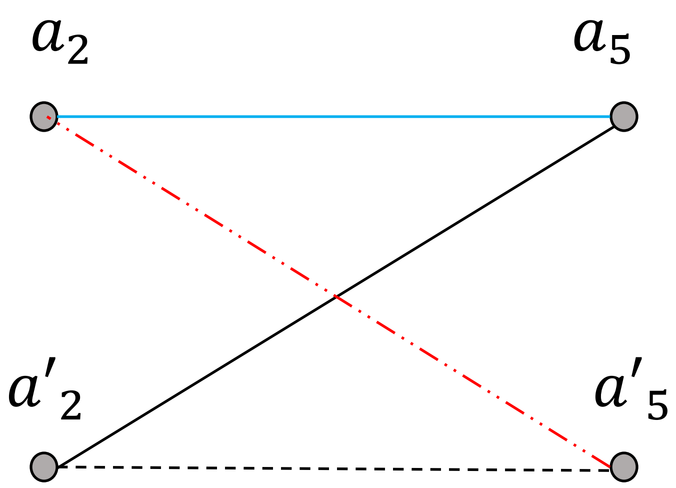
Insecurity of 5LR



For each line, edges between two vertices



Cycles



$\Rightarrow \quad b_3 \oplus b_4 \oplus b'_3 \oplus b'_4 = 0$

Constructing Cycle is NOT EASY in 6LR....

- Disjoint Shores of **points**: $\mathbb{S}_1, \mathbb{S}_2, \ldots, \mathbb{S}_6$

- **Line Geometry**: **set of lines, no two lines intersect at more than one points.**
- Each point is coloured: Black, White and Grey.

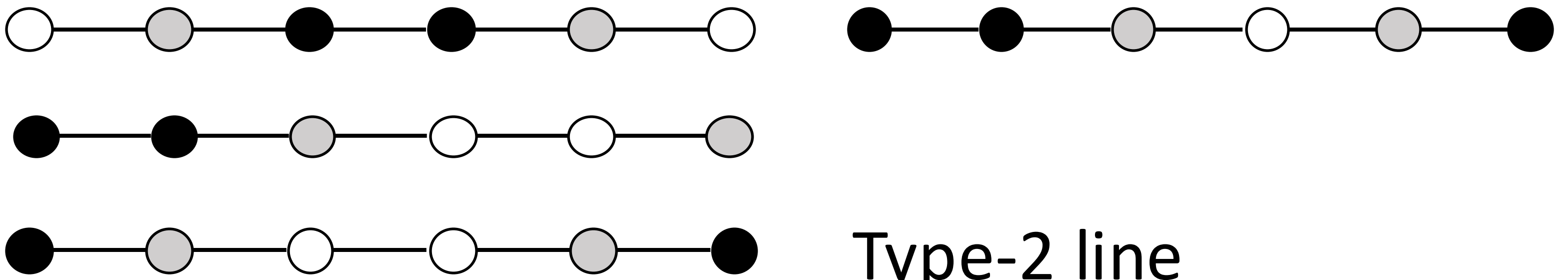
Type-1 line

Gen-3 line

Sure Revealed Lines:

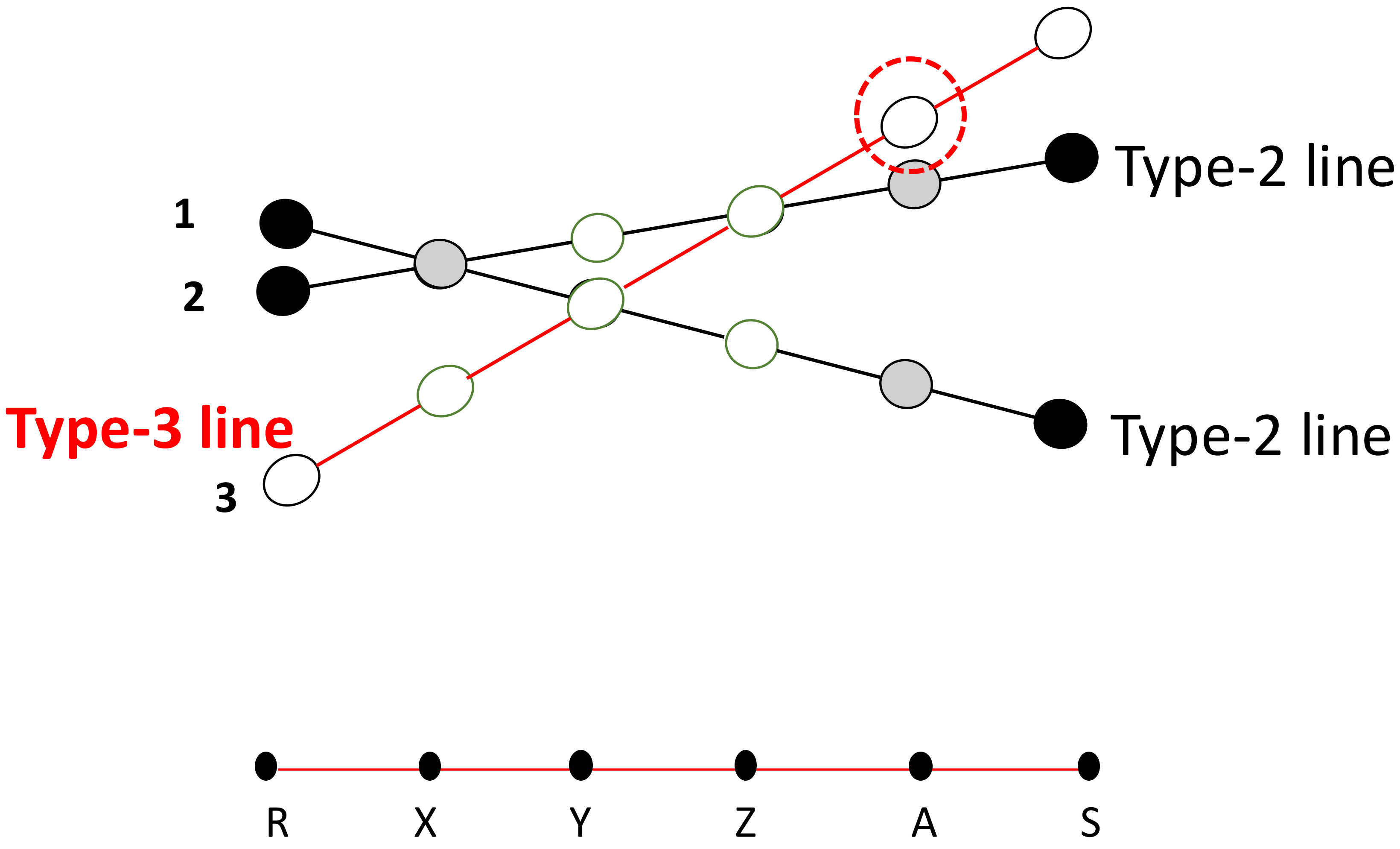
Doubtful Revealed Lines

Hidden Lines:

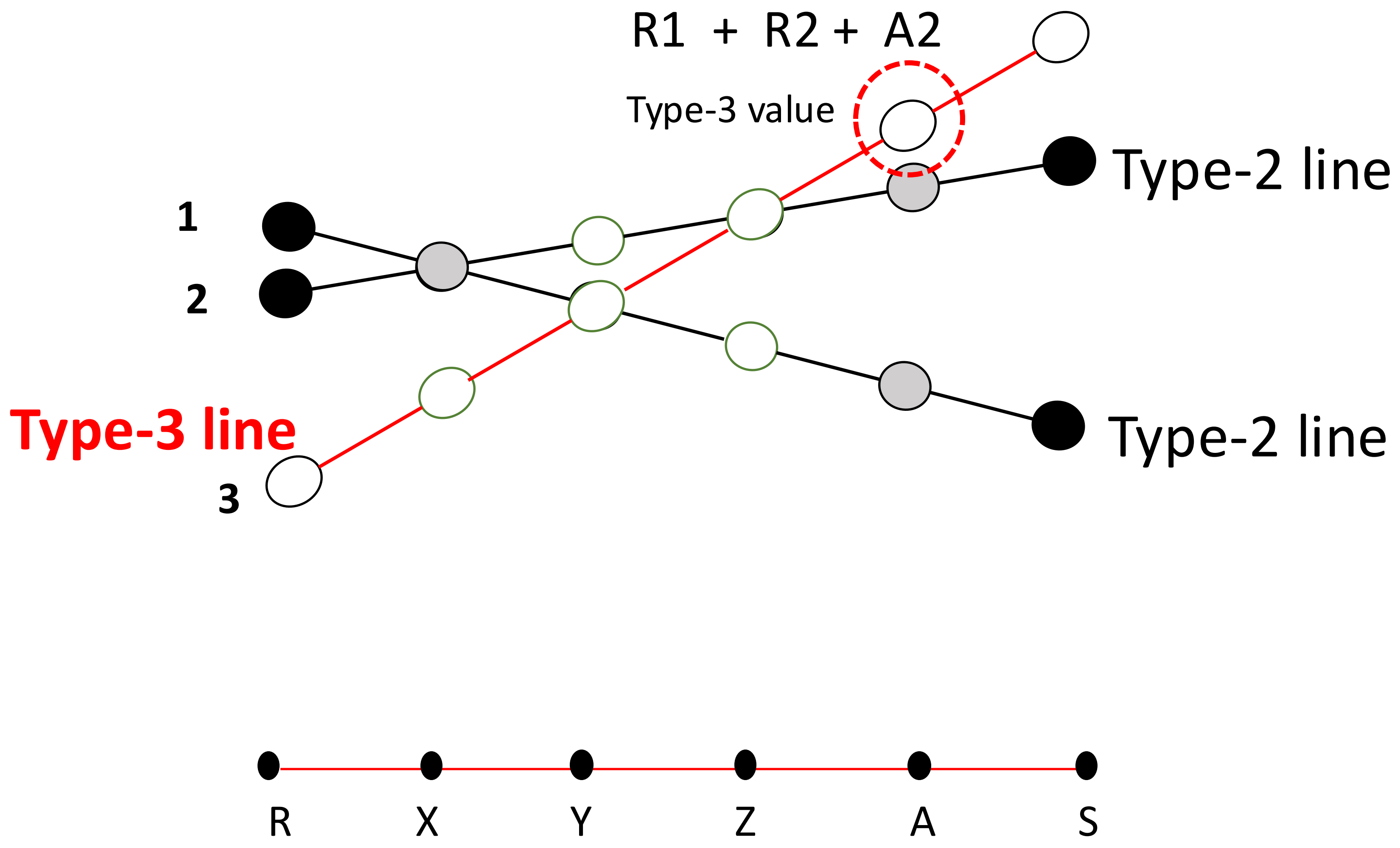


Type-2 line

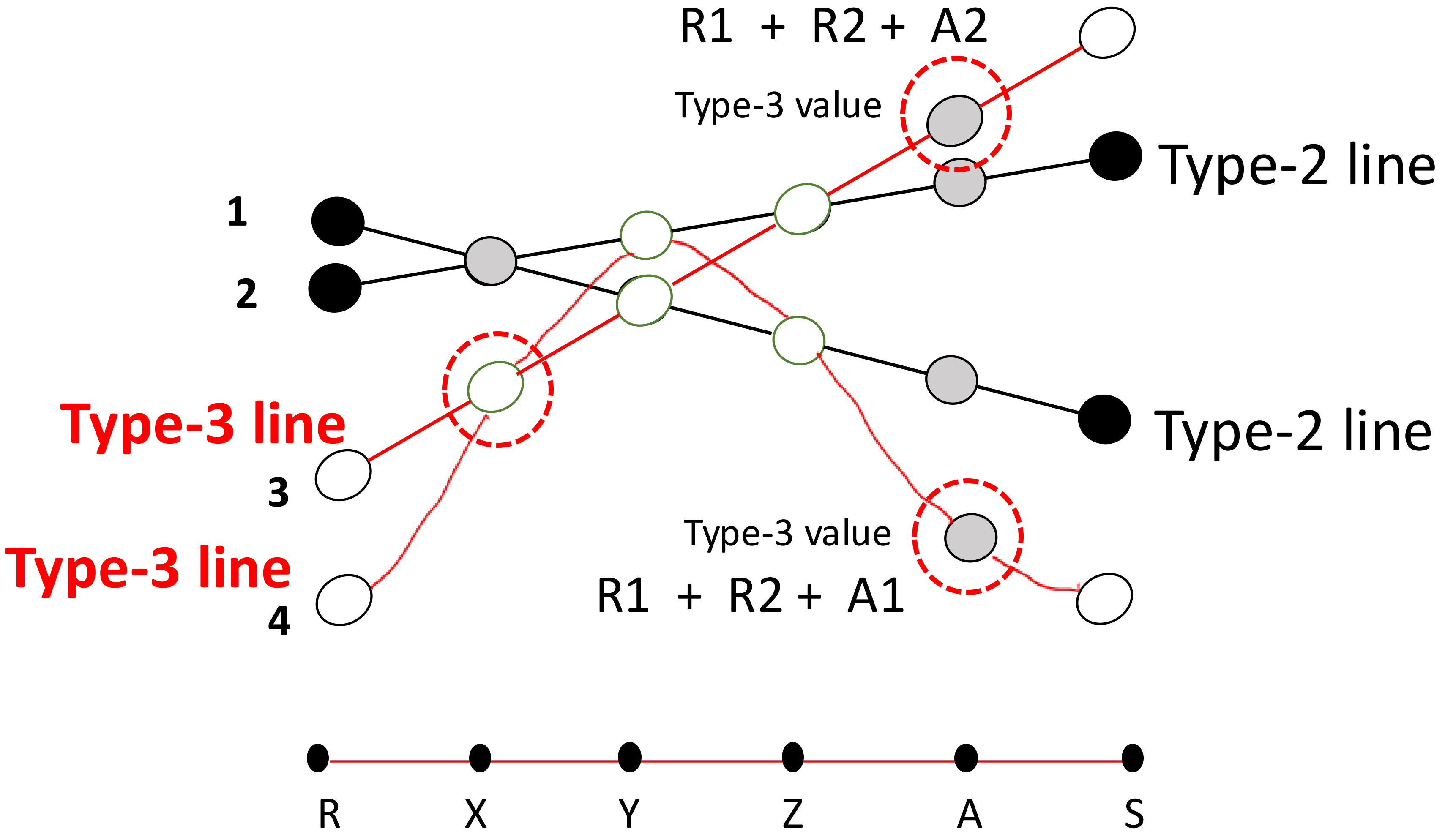
Type-3 Lines



Type-3 Lines


R1 + R2 + A2
Type-3 value
Type-2 line
1
2
Type-3 line
3
Type-2 line
R
X
Y
Z
A
S

Type-3 Lines


R1 + R2 + A2
Type-3 value
Type-2 line
1
2
Type-3 line
3
Type-3 line
4
Type-2 line
Type-3 value
R1 + R2 + A1
R
X
Y
Z
A
S

Basic Simulation and Issue

Simulation Fails

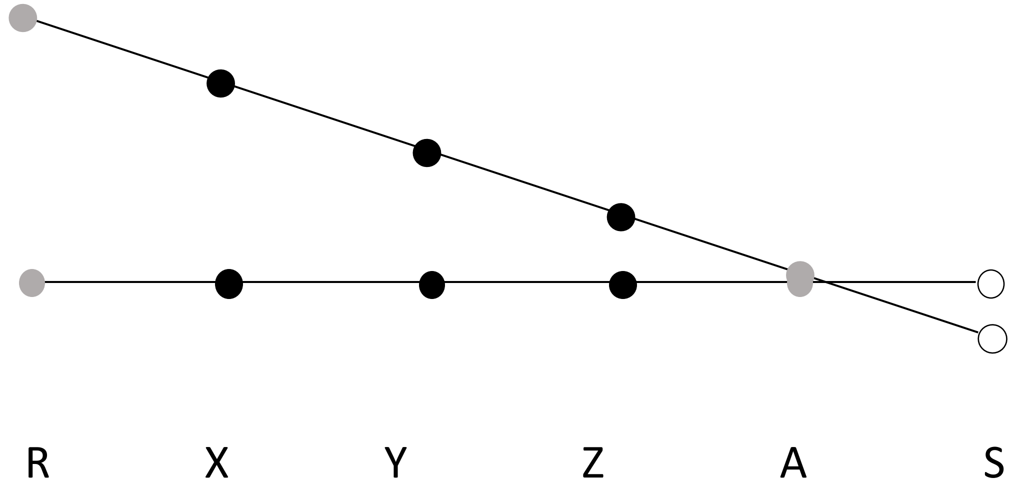
Case 1

Sample 1 and then forward construction query does not guarantee 6

Sample 6 and then backward construction query does not guarantee 1

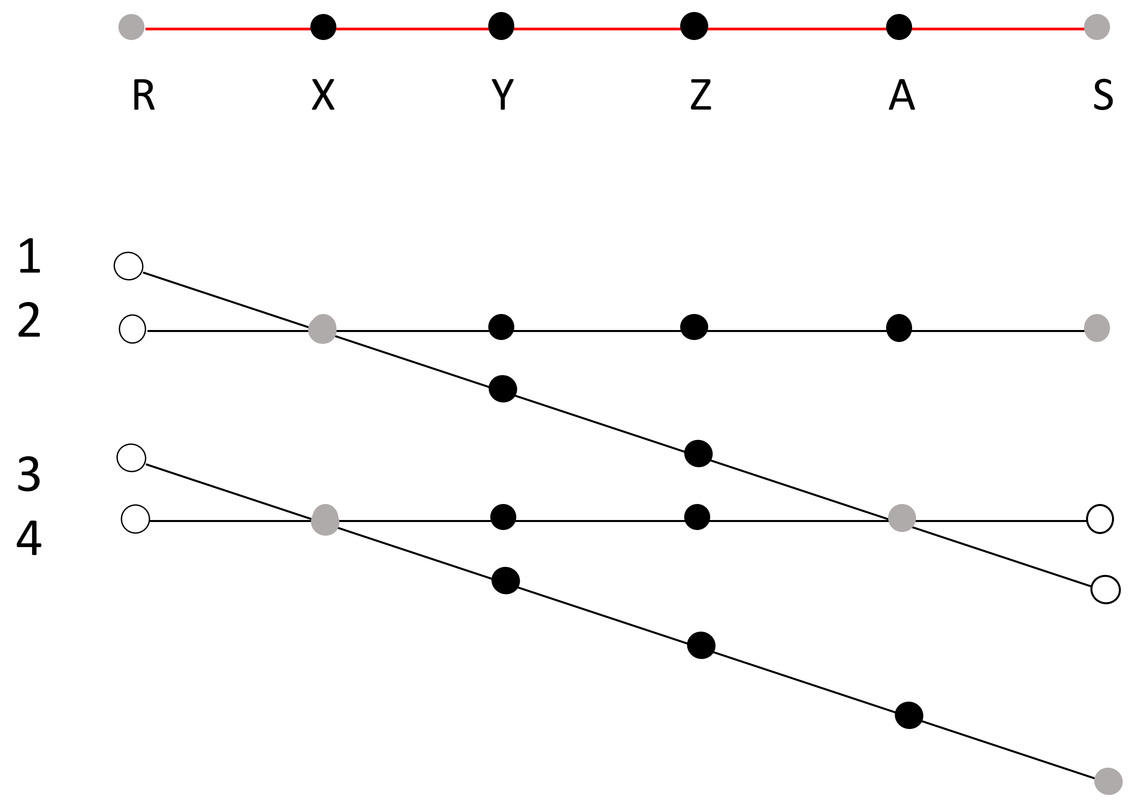
$$\Pi(\ell, r) = (s, t)$$

Case 2



R1 and R2 fixed and
S1 + S2 = Z1 + Z2
Π(R1,*) + Π(R2, *) = constant

Simulation Fails



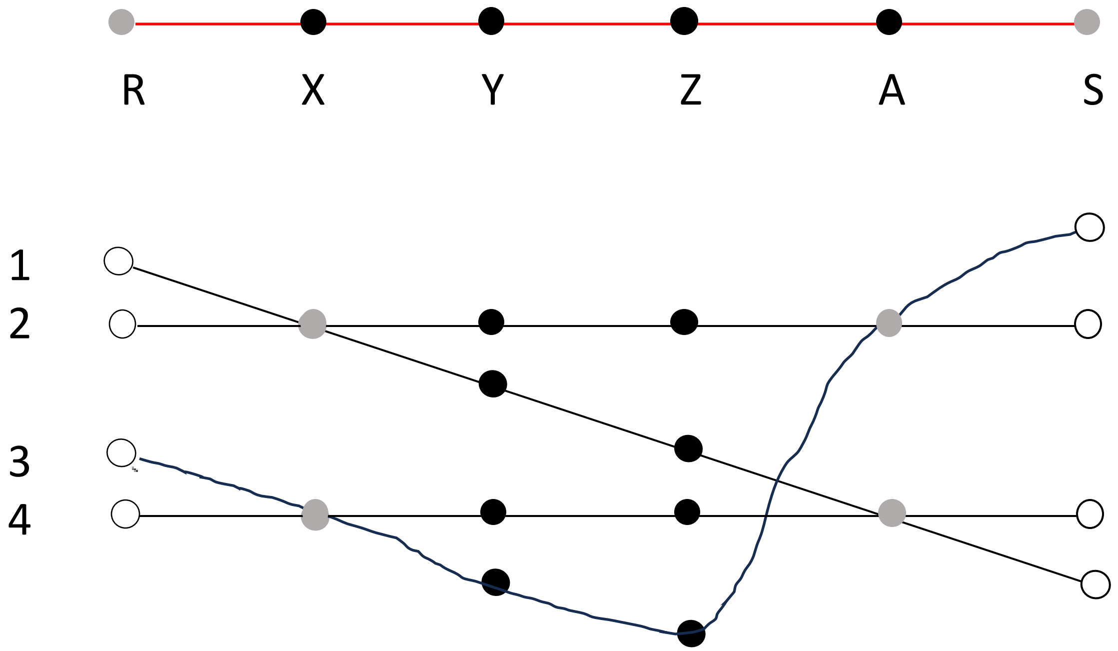
S2 and S3 fixed

R2 + R1 = Y1 + Y2

S1 + S4 = Z1 + Z4

S4 + S3 = Y3 + Y4

Simulation Fails



Cycle in 2-5 Line Graph

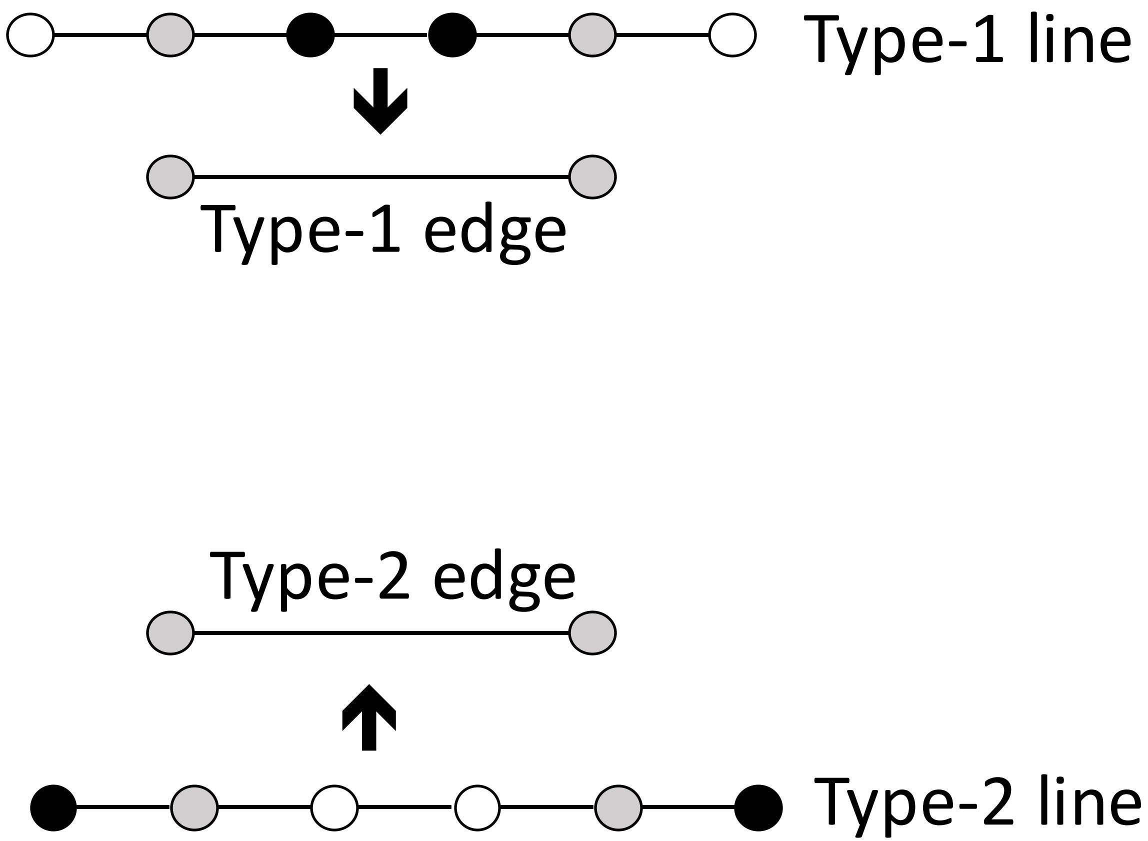
R1 + R2 + R3 + R4 fixed

S1 + S2 + S3 + S4 fixed

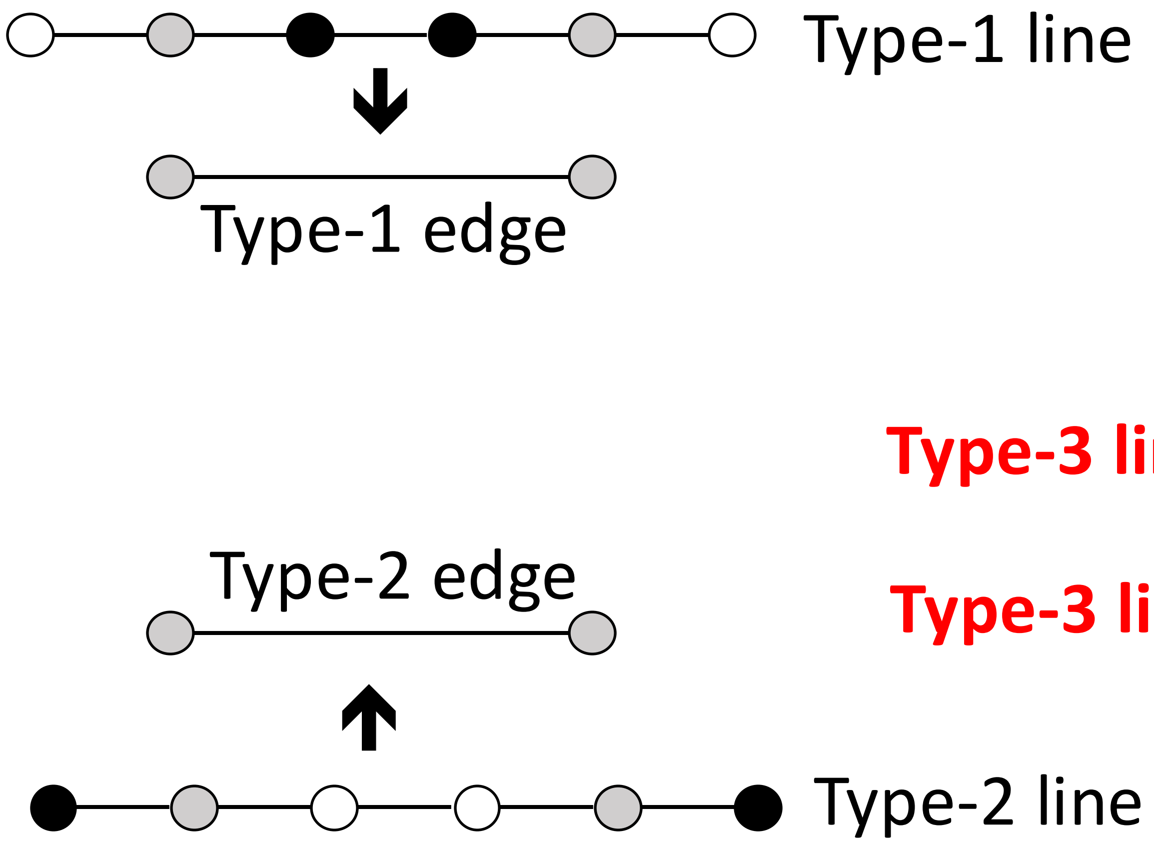
Y1 + [Y1] + … + Y4 + [Y4] +
Z1 + [Z1] + … + Z4 + [Z4] = 0

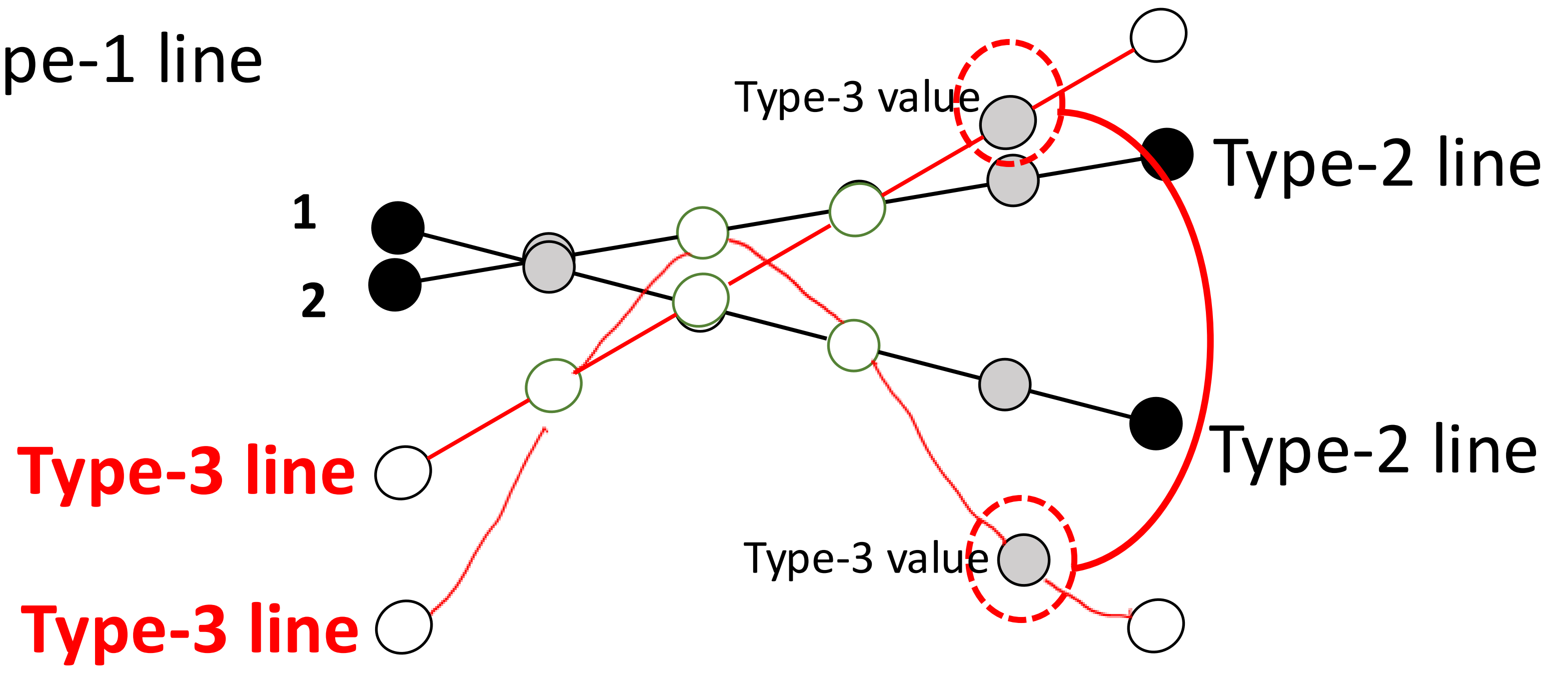
This holds with low prob in real world.

How do we capture all these?



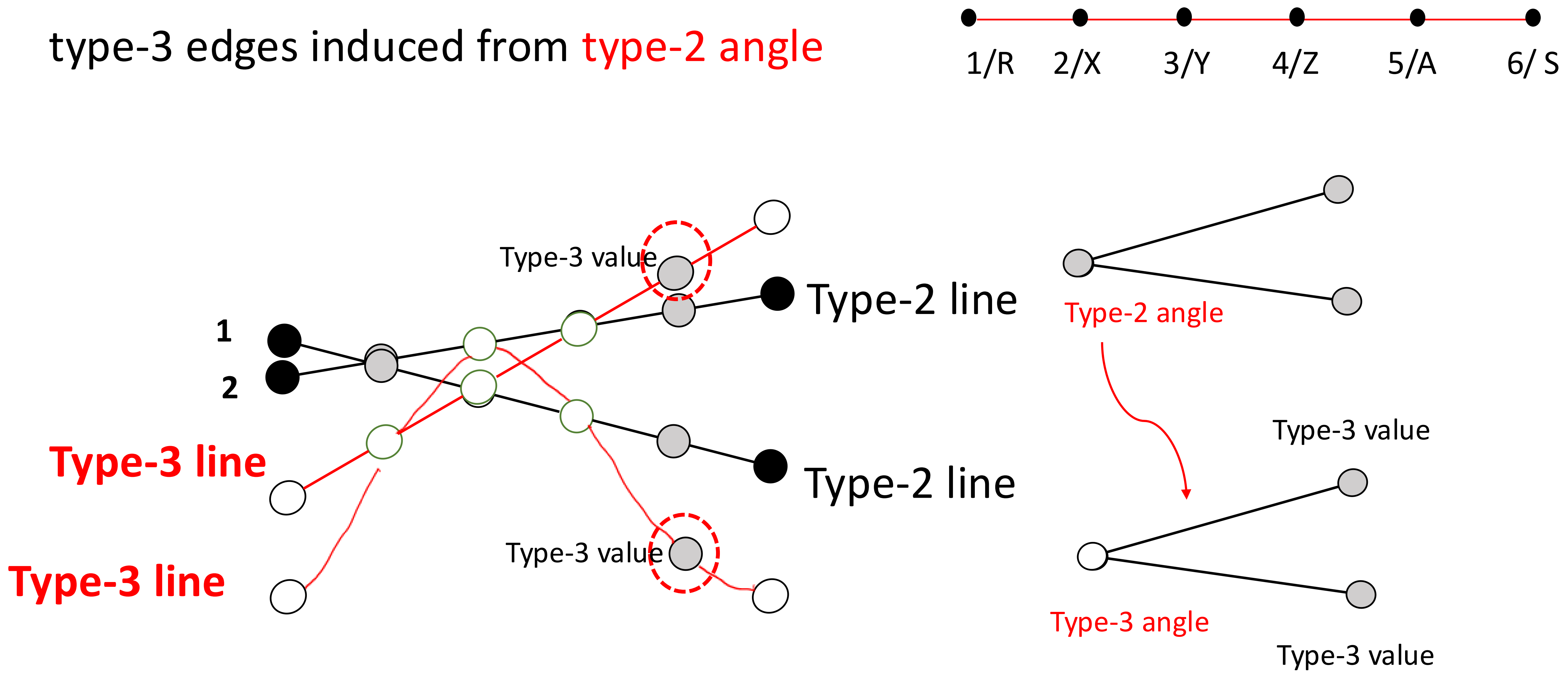
Line Graph


Type-1 line
Type-1 edge
Type-2 edge
Type-2 line

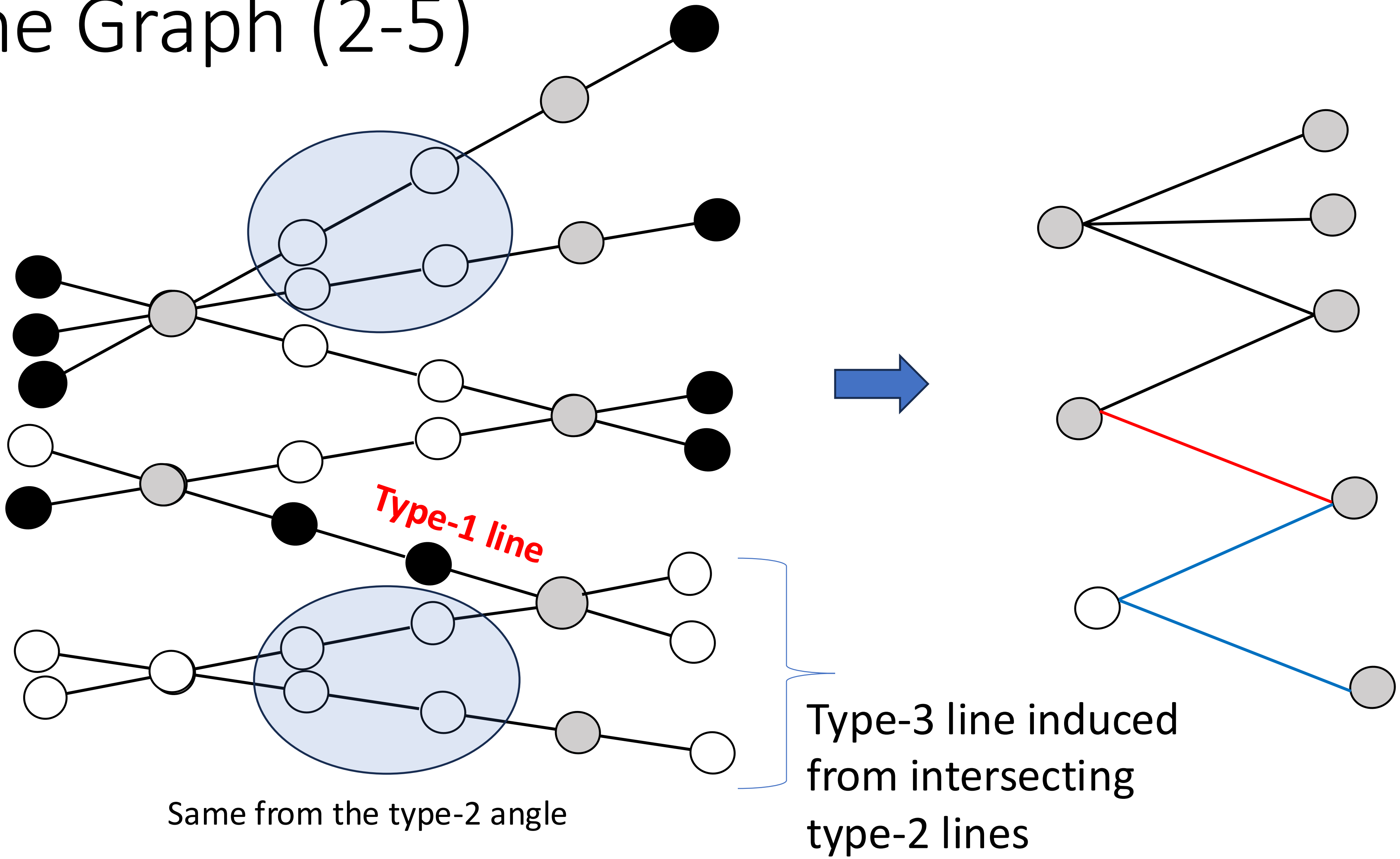

Type-3 value
Type-2 line
1
2
Type-2 line
Type-3 line
Type-3 value
Type-3 line

Line Graph

type-3 edges induced from type-2 angle



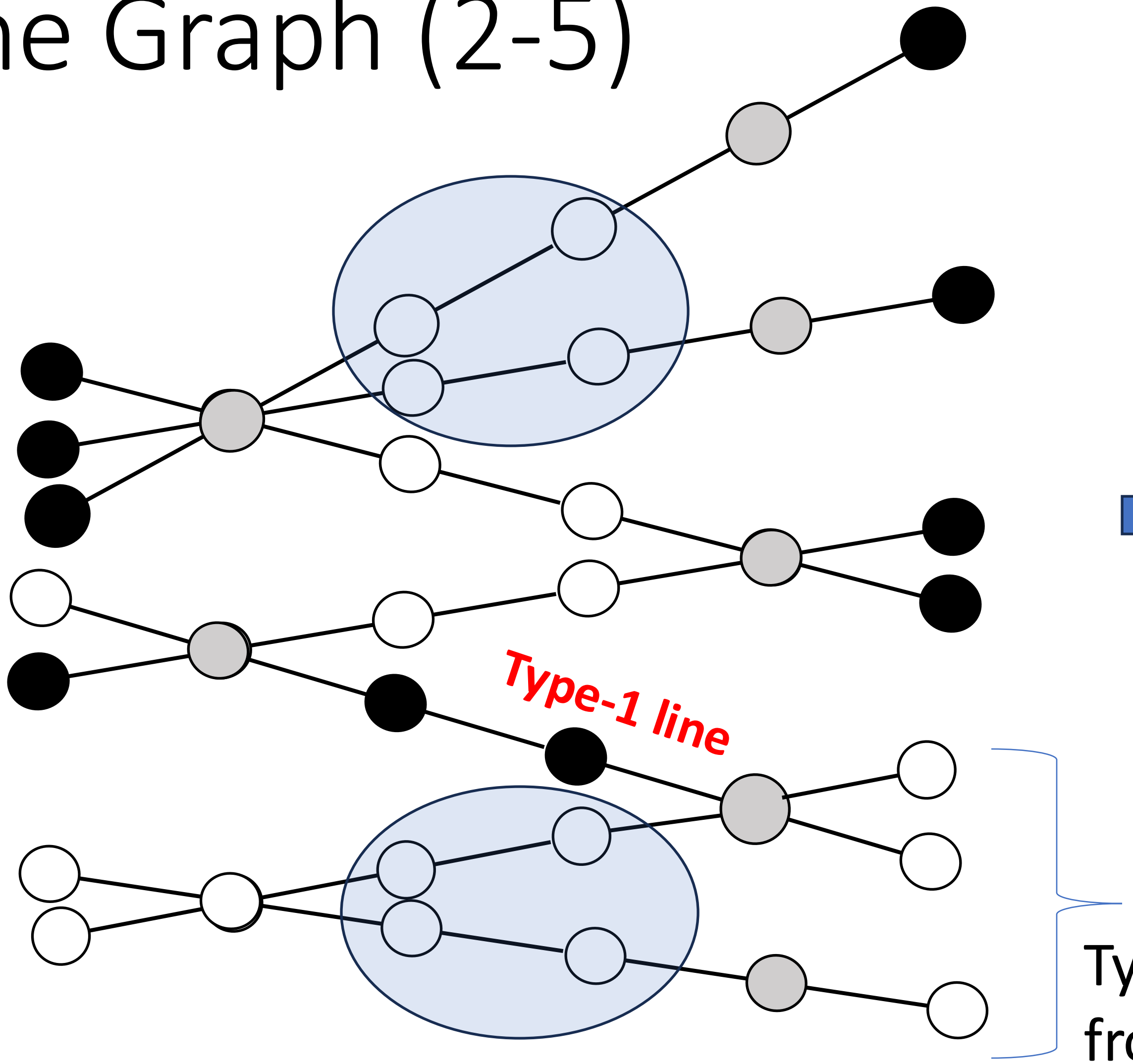
Line Graph (2-5)



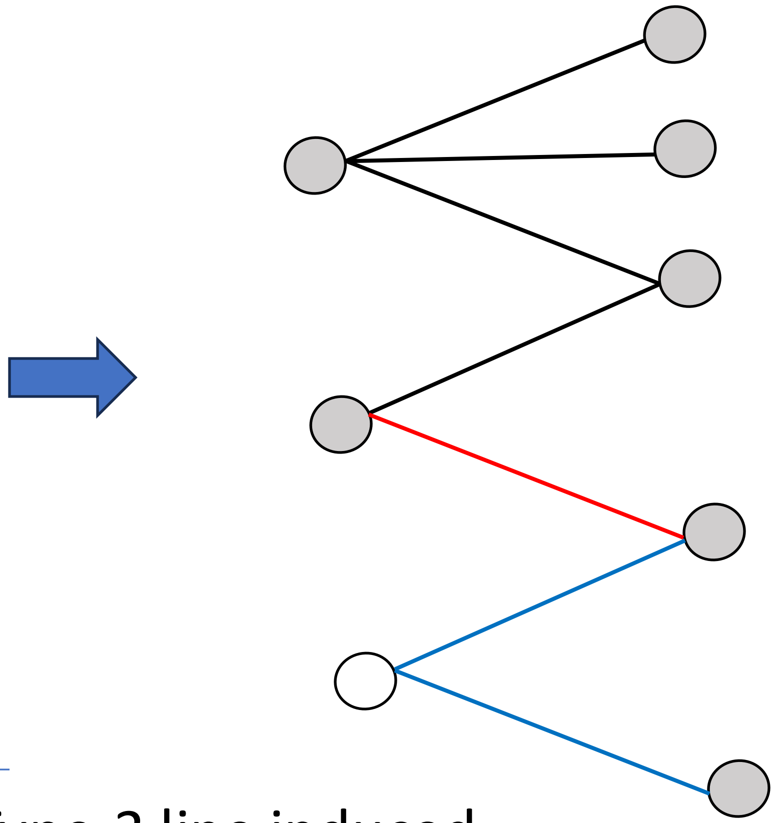
Same from the type-2 angle

Line Graph (2-5)

We can similarly define Line-graph for 1-4 and 3-6.

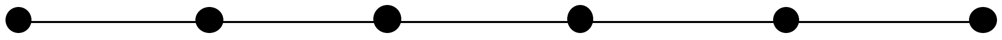


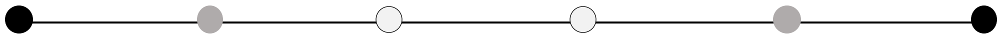
Same from the type-2 angle



Type-3 line induced from intersecting type-2 lines

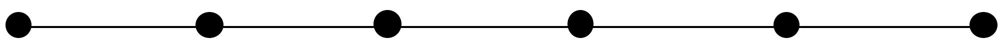
Simulator (Coron et al. Crypto 2008)

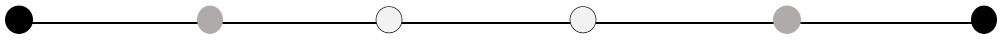
Simulation Steps to make 



Type-2 line

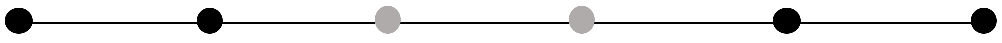
Simulation Steps to make





Type-2 line

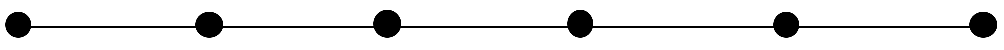
Sample (if required) 2 and 5

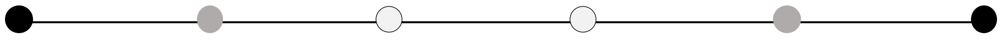


$$a_3 = F(a_2) + a_1$$
$$F(a_3) = a_2 + a_4,$$

$$a_4 = F(a_5) + a_6,$$
$$F(a_4) = a_3 + a_5$$

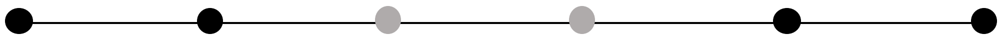
Simulation Steps to make





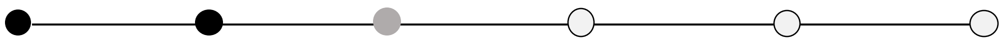
Type-2 line

Sample (if required) 2 and 5

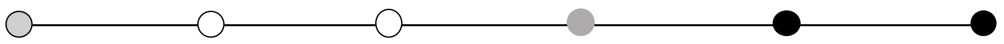


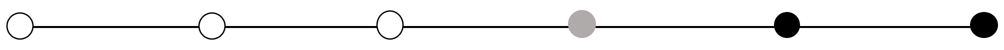
$$a_3 = F(a_2) + a_1$$
$$F(a_3) = a_2 + a_4,$$

$$a_4 = F(a_5) + a_6,$$
$$F(a_4) = a_3 + a_5$$

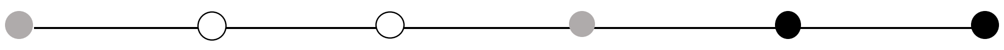


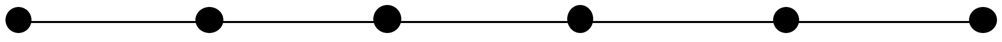
Forward Construction Query and then as before ➔

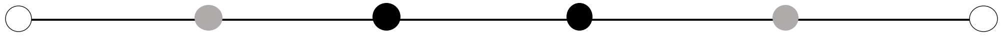




Similarly we first apply Backward Construction Query ➔

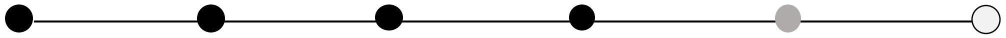


Simulation Steps to make 

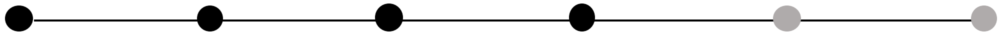

Type-1 line

Option 1 : Forward Direction

Sample (if required) 1 and 2 and Π query

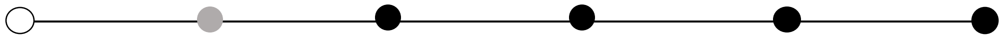


Forward Construction Query

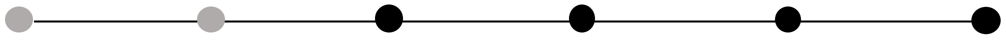


Option 2 : Backward Direction

Sample (if required) 5 and 6 and Π^{-1} query

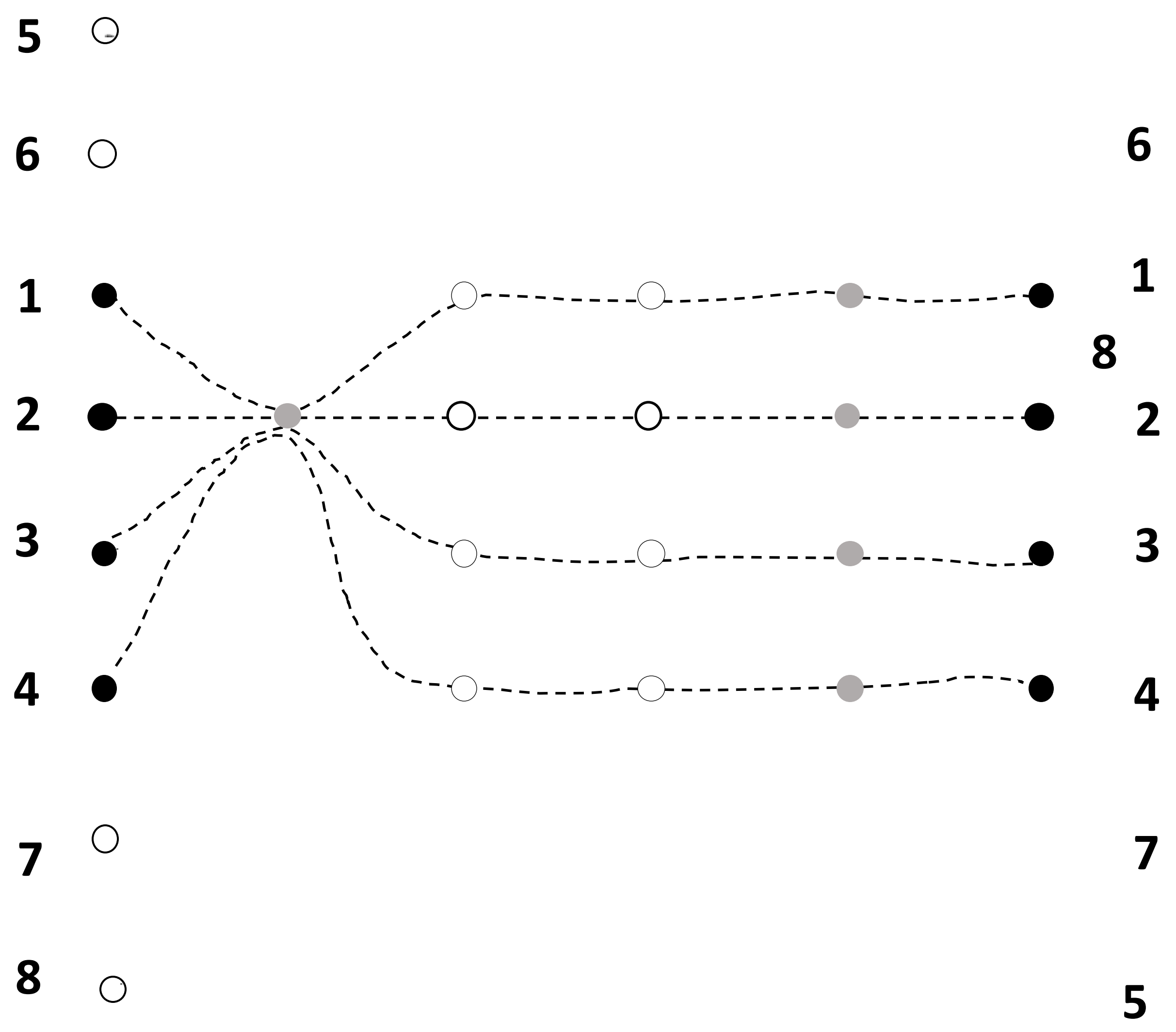


Backward Construction Query

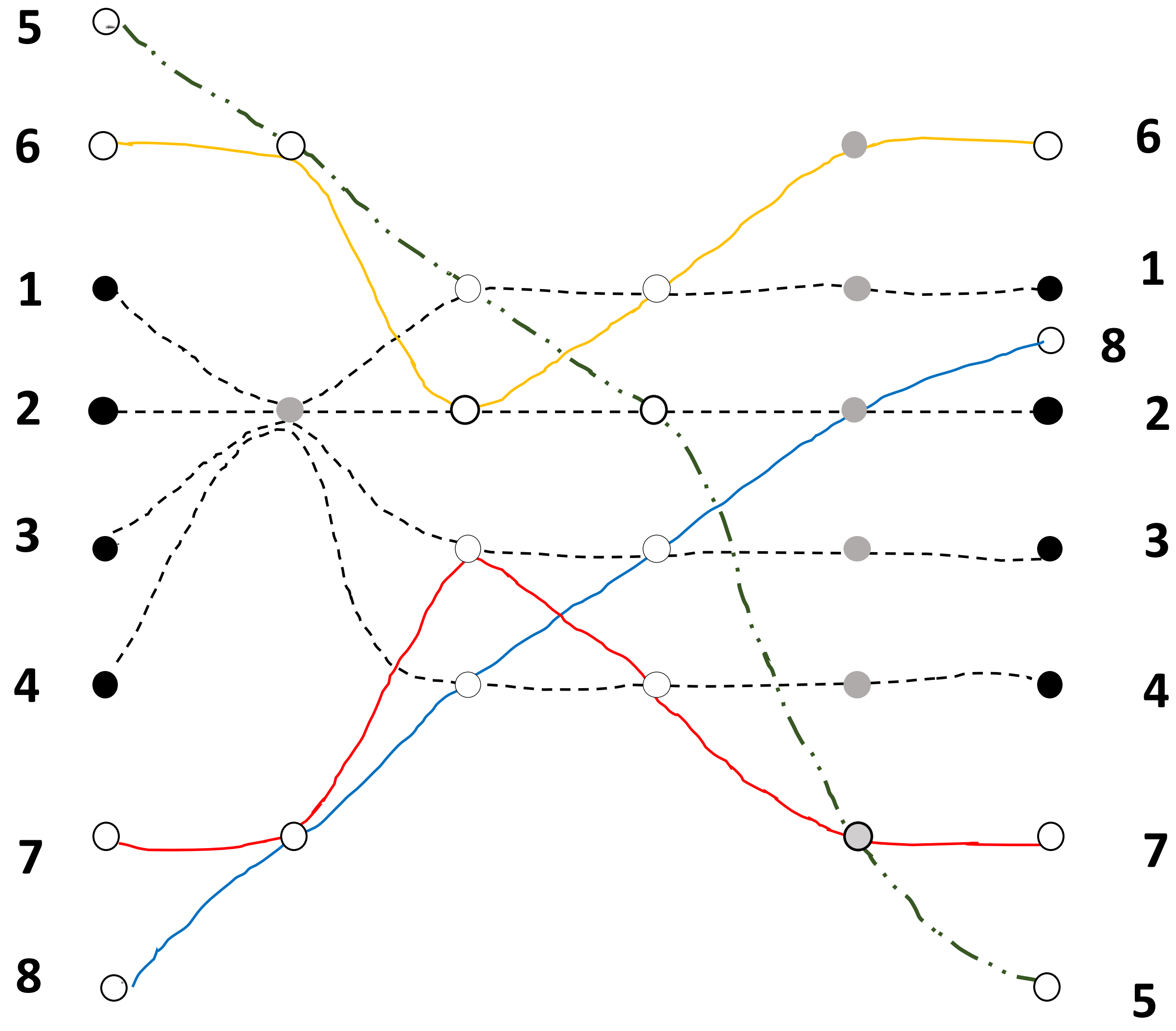


Set $F(a_5) = a_4 + a_6,\ F(a_6) = a_5 + t$

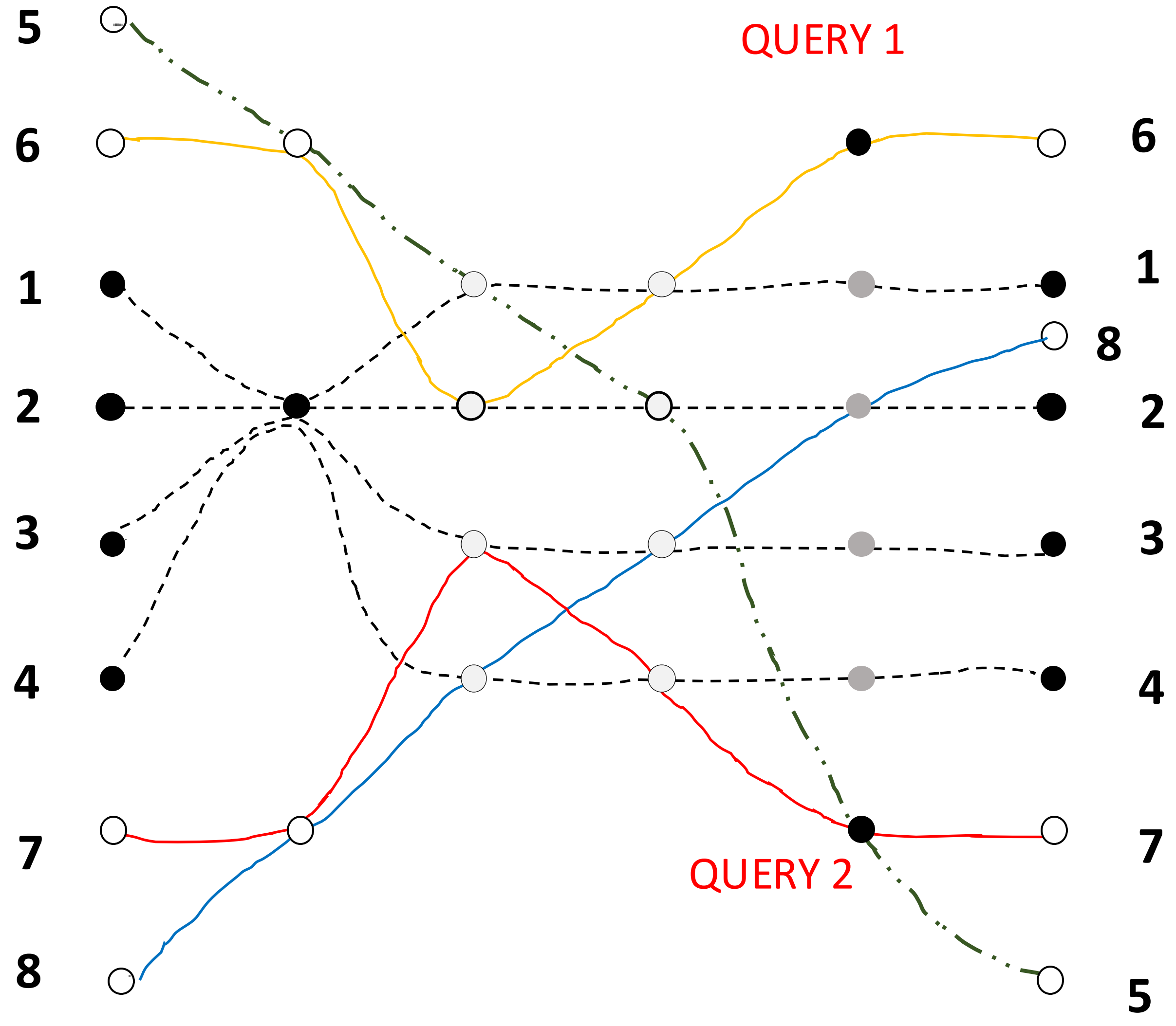
Holenstein, Kunzler, Tessaro Attack STOC 11



Holenstein, Kunzler, Tessaro Attack STOC 11

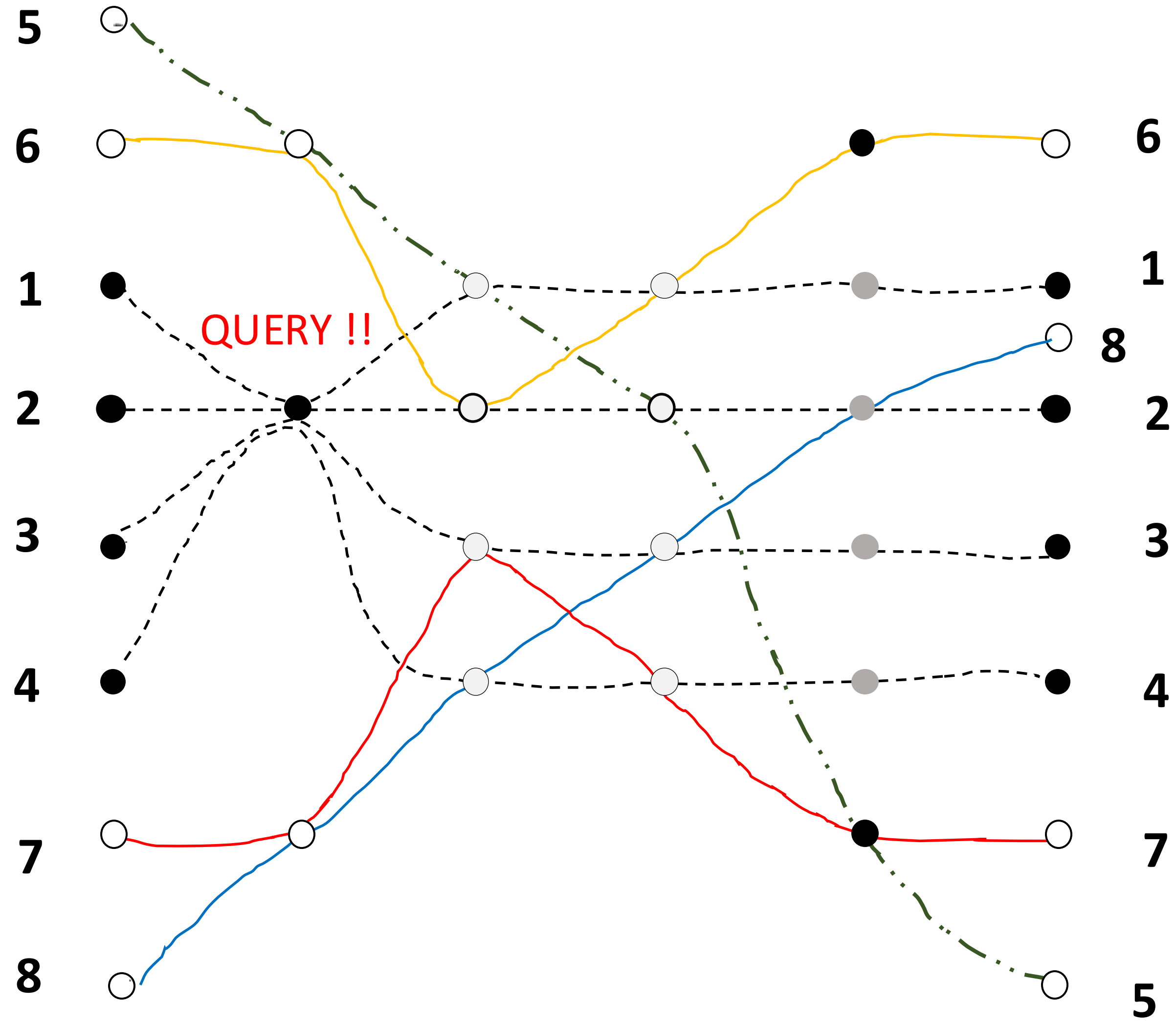


Holenstein, Kunzler, Tessaro Attack STOC 11



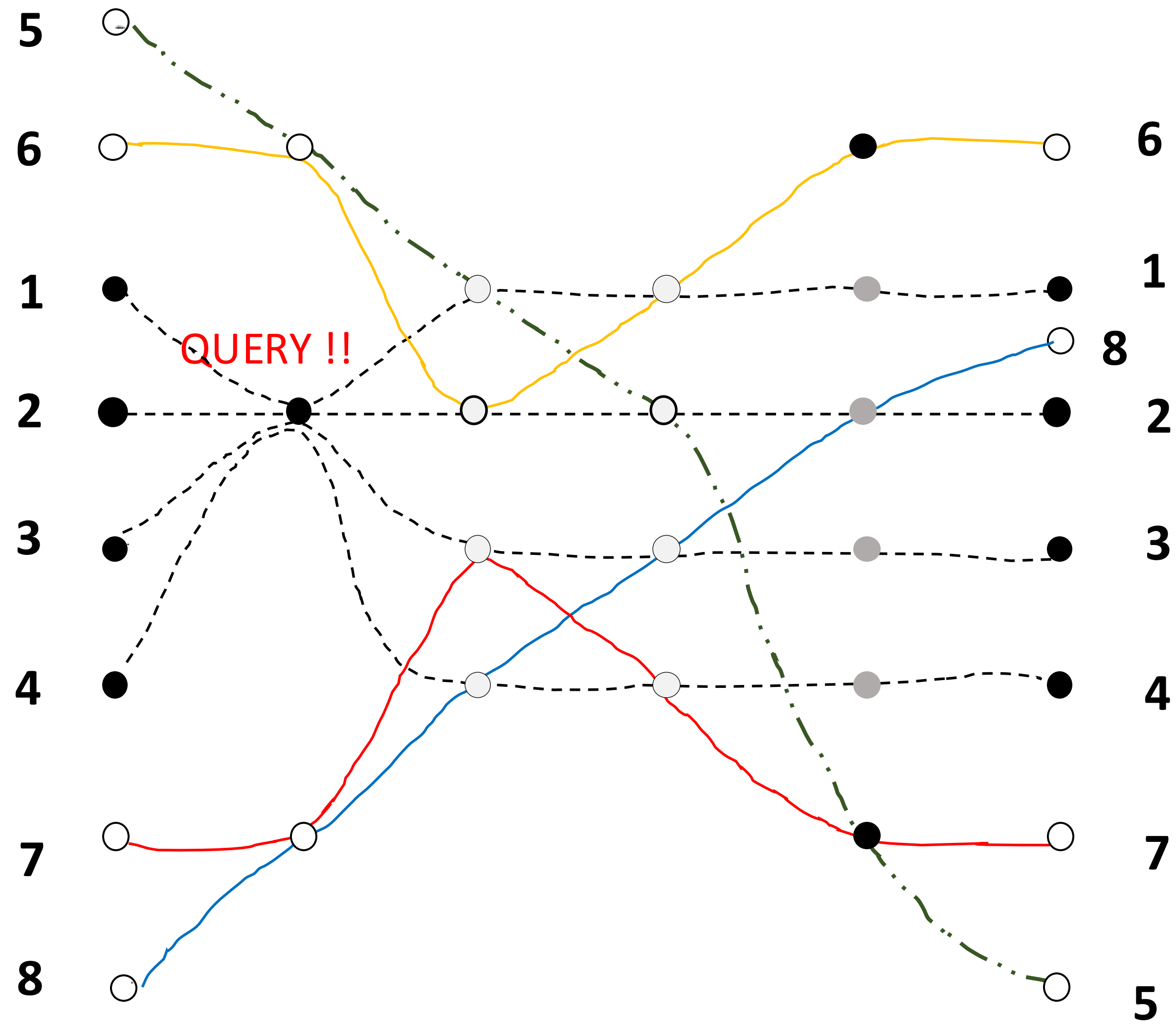
Holenstein, Kunzler, Tessaro Attack STOC 11

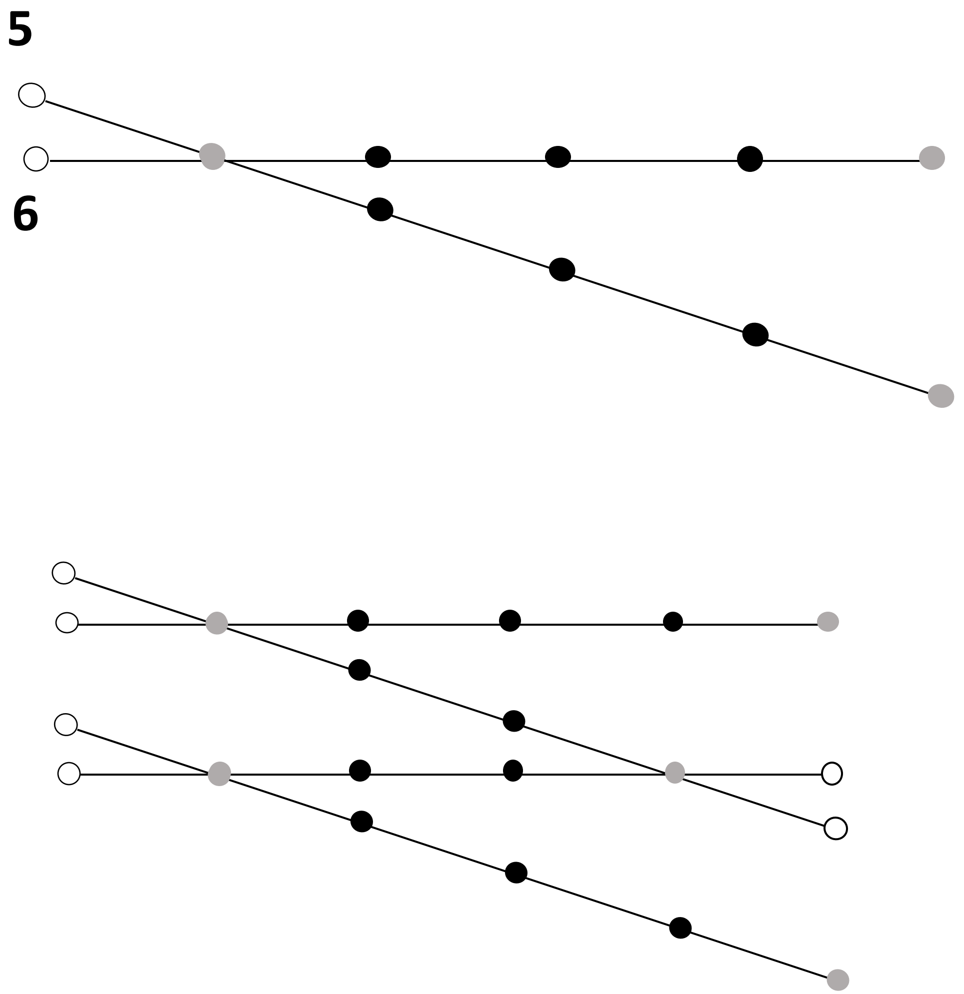
Gen 3 Lines are revealed and start simulating lines



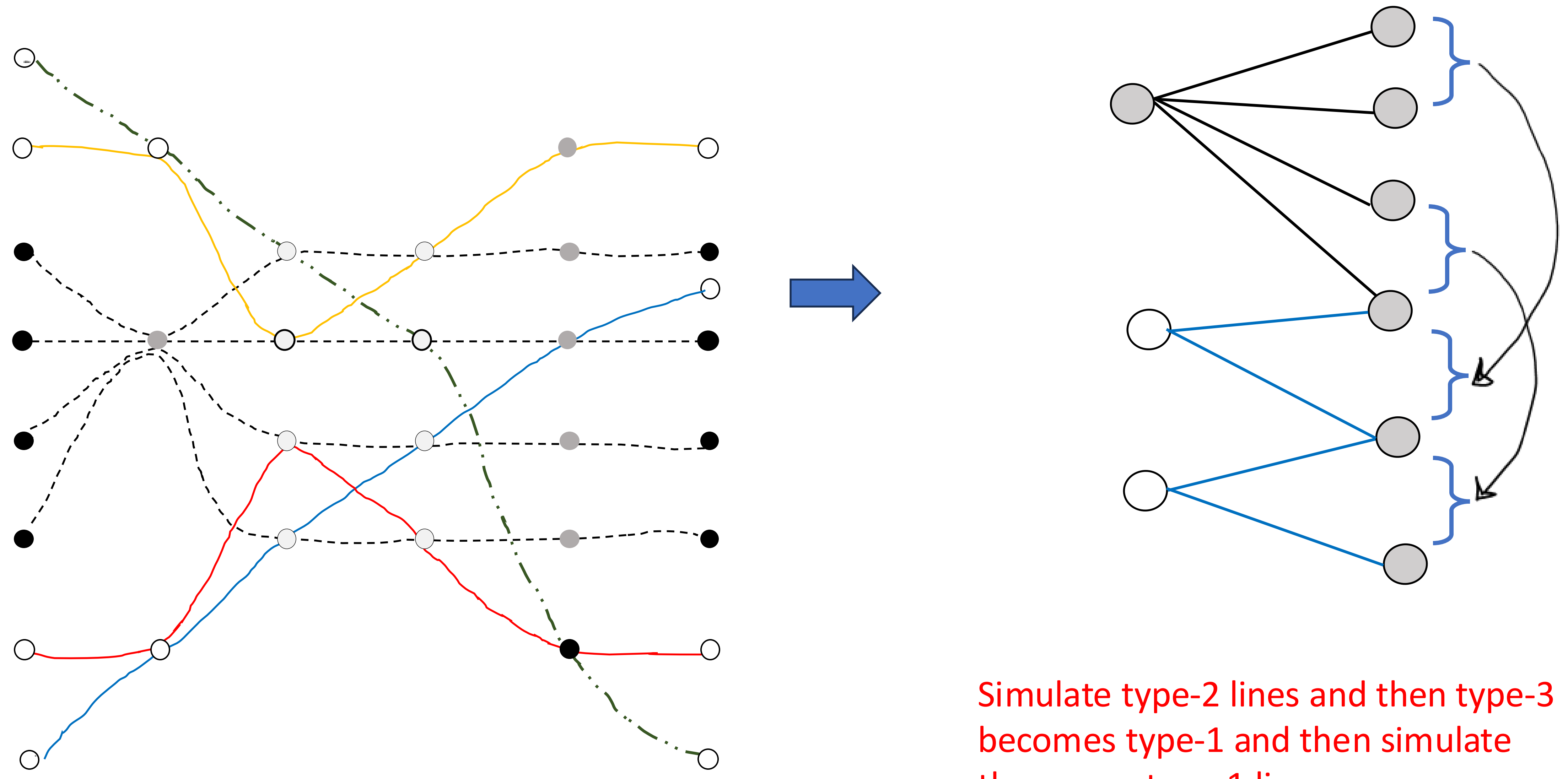
Holenstein, Kunzler, Tessaro Attack STOC 11

Gen 3 Lines are revealed and start simulating lines



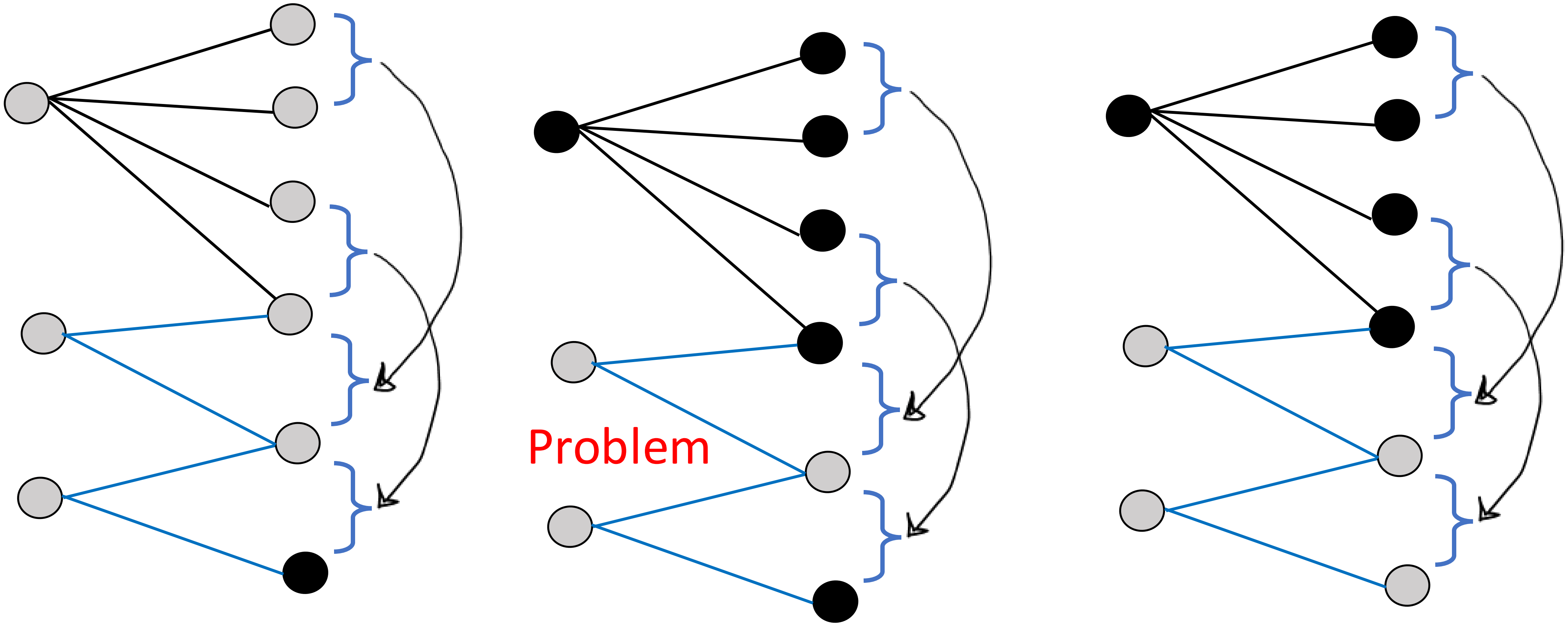


Holenstein, Kunzler, Tessaro Attack STOC 11



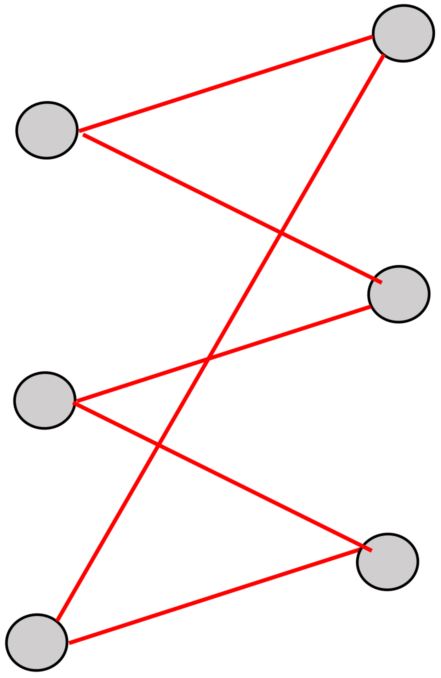
Simulate type-2 lines and then type-3 becomes type-1 and then simulate these new type-1 lines.

Holenstein, Kunzler, Tessaro Attack STOC 11

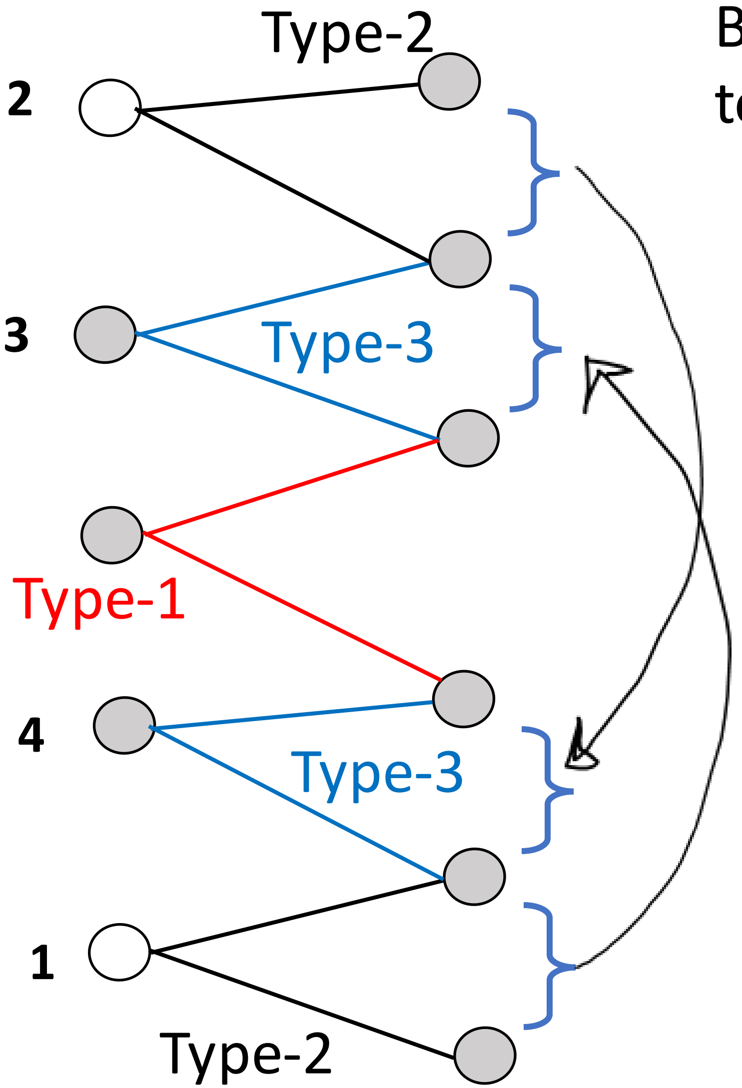


Simulate 4 type-2 lines and
then type-3 lines in order

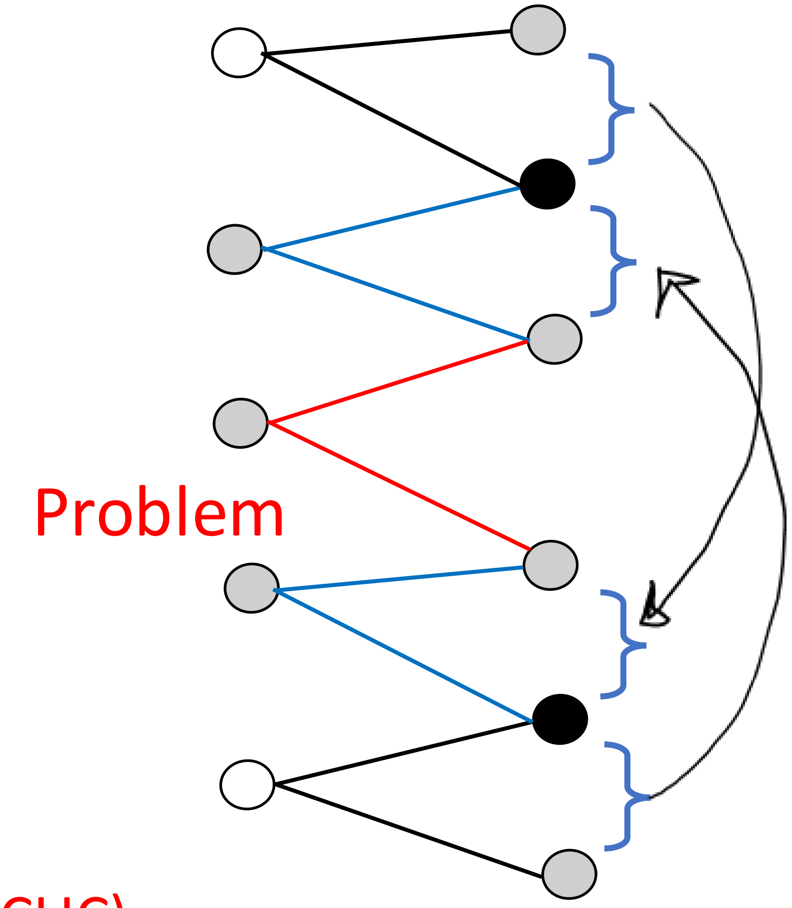
HARD TO SIMULATE



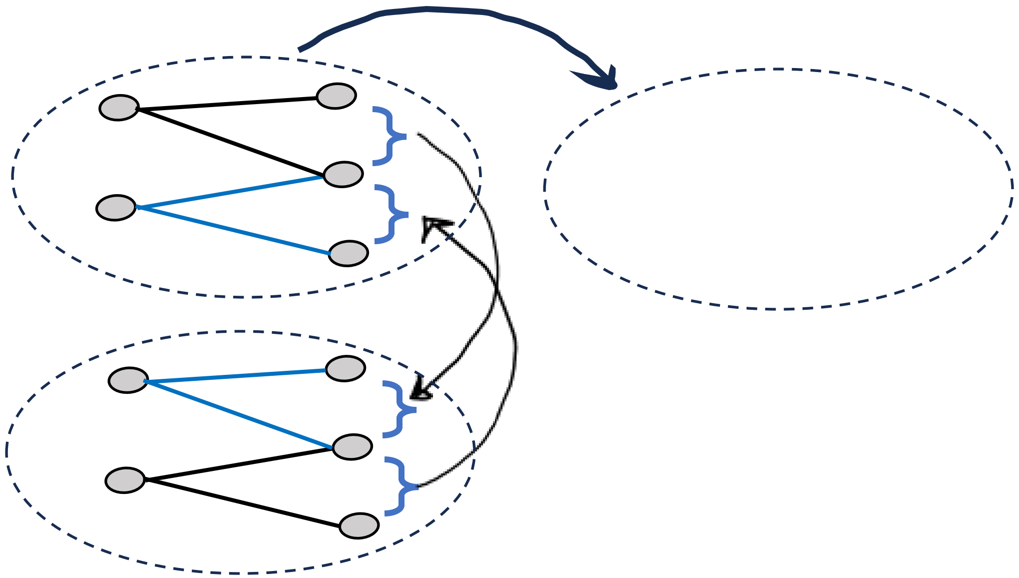
CYCLE



Before we simulate type-3, we need to simulate corresponding type-2



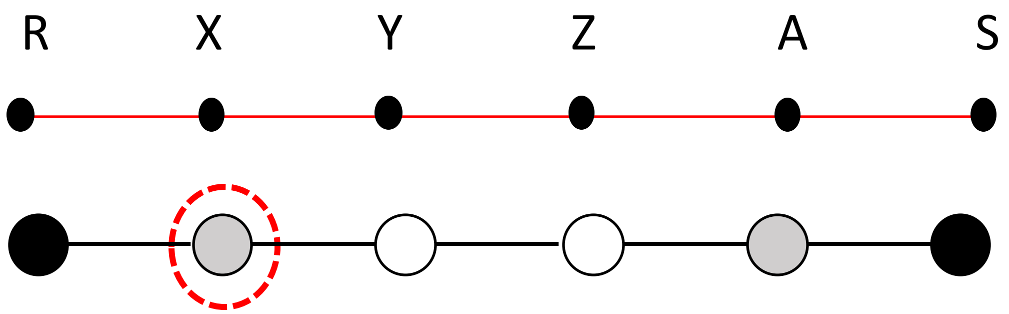
NOT Simulation Friendly (ACYCLIC)



Part 6
Simulator Description

Main Steps:

- CompTrace(u) : Identify 2-5 component (extended) C containing u.
- It was hidden but must be revealed after Sim knows u.



Query = x

For all $R_1 \in Dom(F)$ Checks

$$\Pi(x + [R_1], R_1) = (S_1, *), \qquad S_1 \in Dom(F)$$

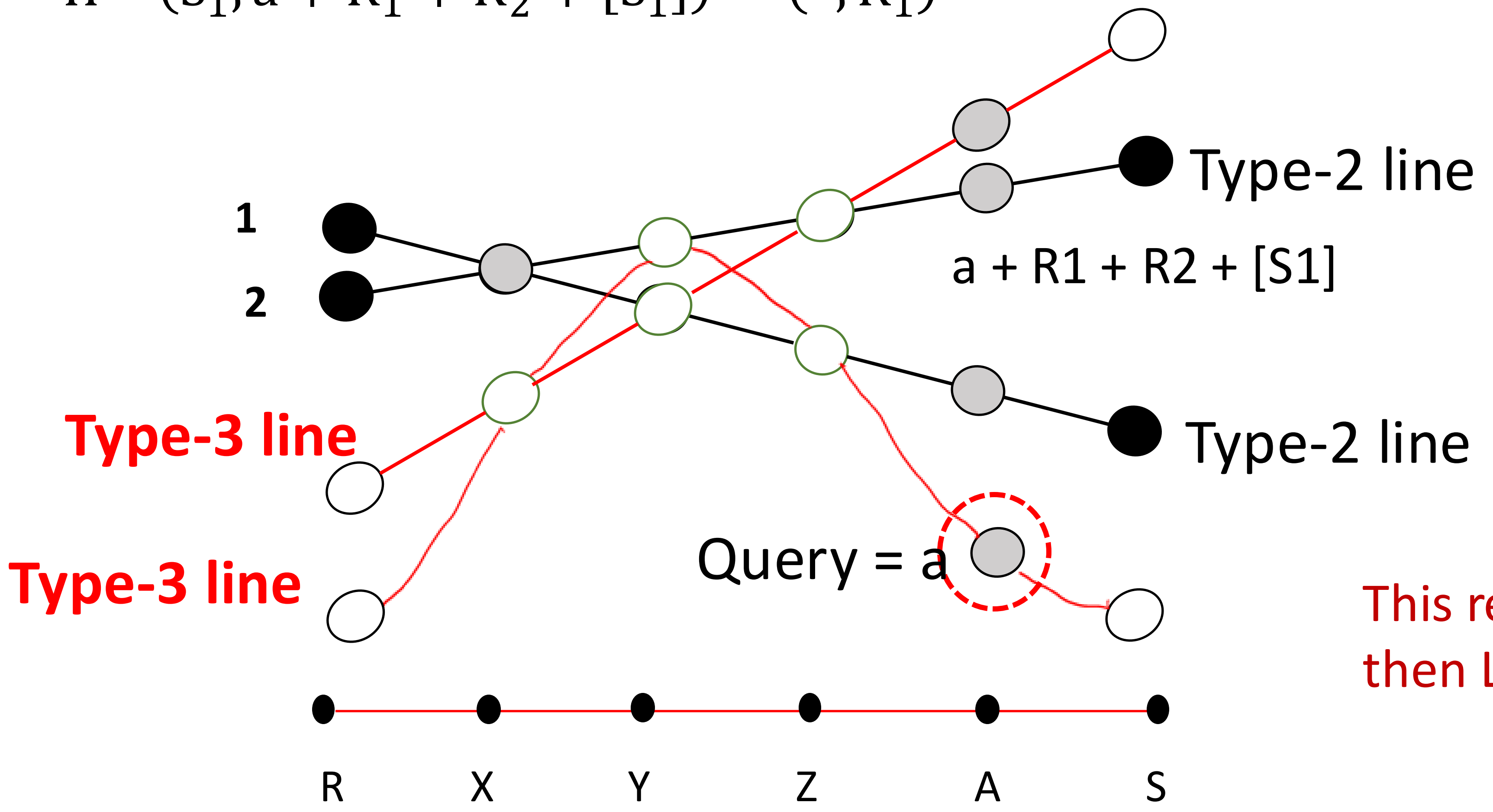
This reveals A1. We can continue…

Main Steps:

- CompTrace(u): Identify 2-5 component (extended) C containing u.

For all $\mathrm{R}_1, \mathrm{R}_2, \mathrm{S}_1 \in Dom(F)$, Checks

$\Pi^{-1}(\mathrm{S}_1, \mathrm{a} + \mathrm{R}_1 + \mathrm{R}_2 + [\mathrm{S}_1]) = (*, \mathrm{R}_1)$



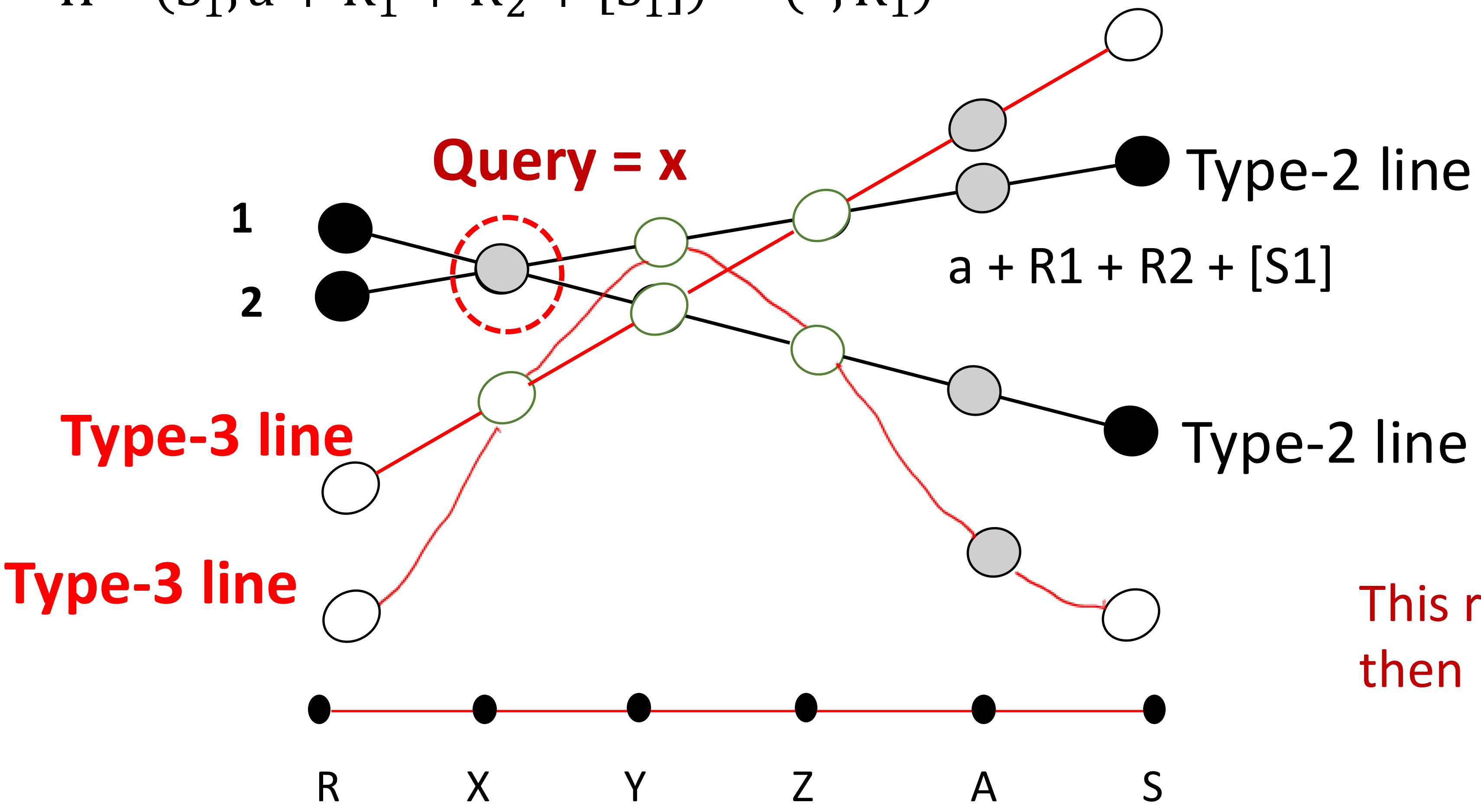
This reveals Line 1 and then Line 2 and so on.

Main Steps:

- CompTrace(u): Identify 2-5 component (extended) C containing u.

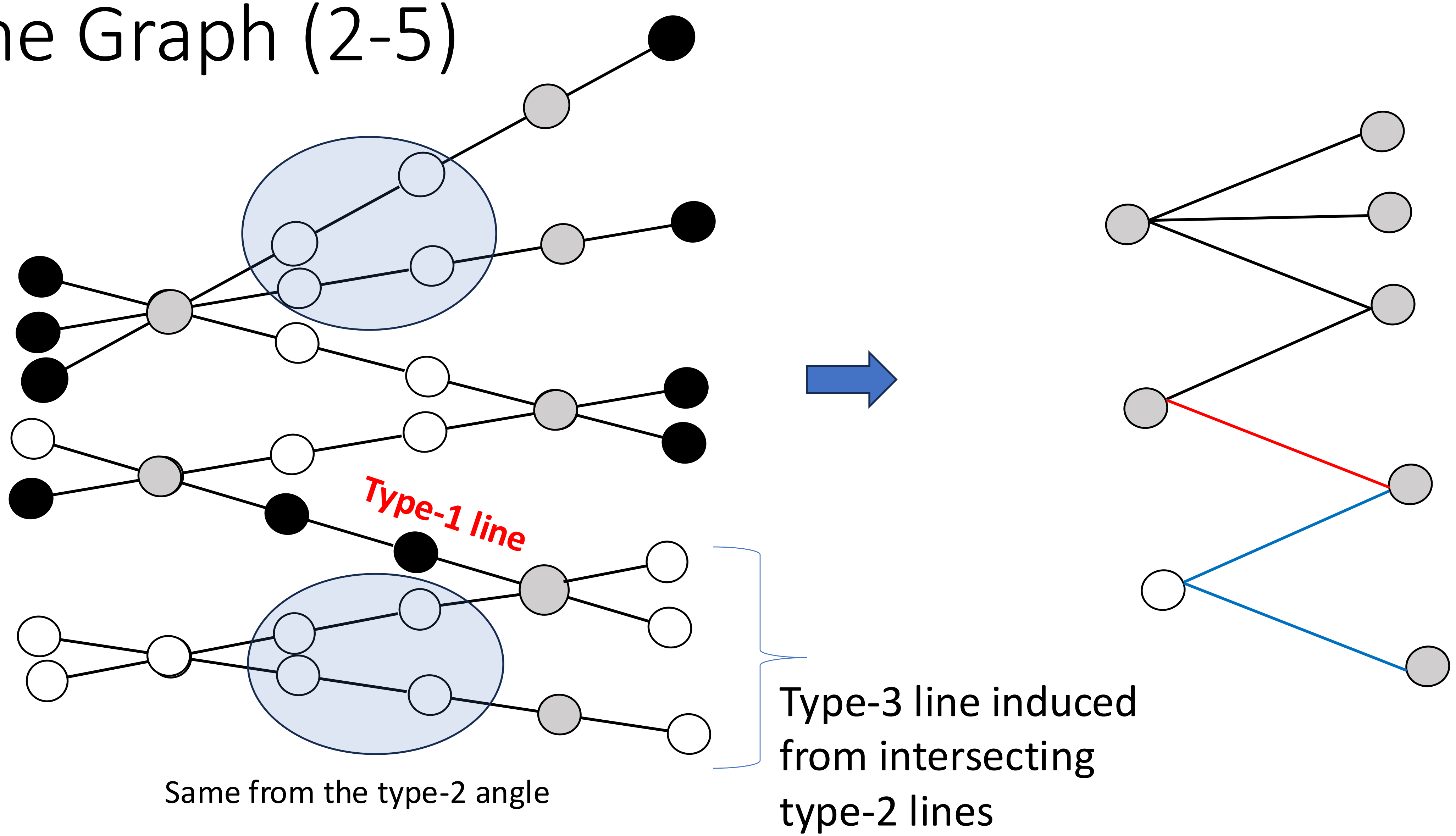
For all $\mathrm{R}_1, \mathrm{R}_2, \mathrm{S}_1 \in Dom(F)$, Checks

$\Pi^{-1}(\mathrm{S}_1, \mathrm{a} + \mathrm{R}_1 + \mathrm{R}_2 + [\mathrm{S}_1]) = (*, \mathrm{R}_1)$



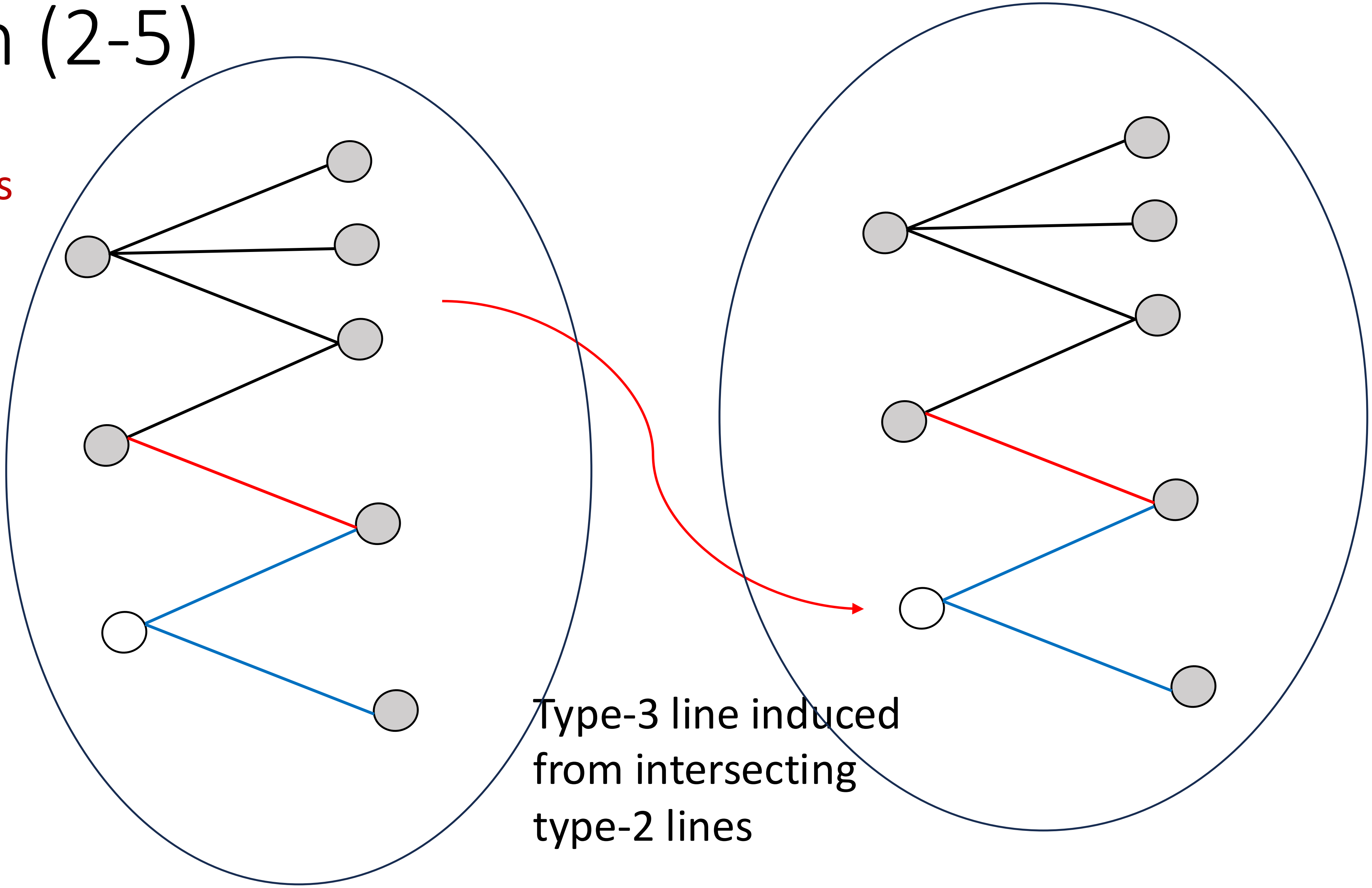
This reveals Line 1 and then Line 2 and so on.

Line Graph (2-5)



Line Graph (2-5)

All Components related to the query are revealed

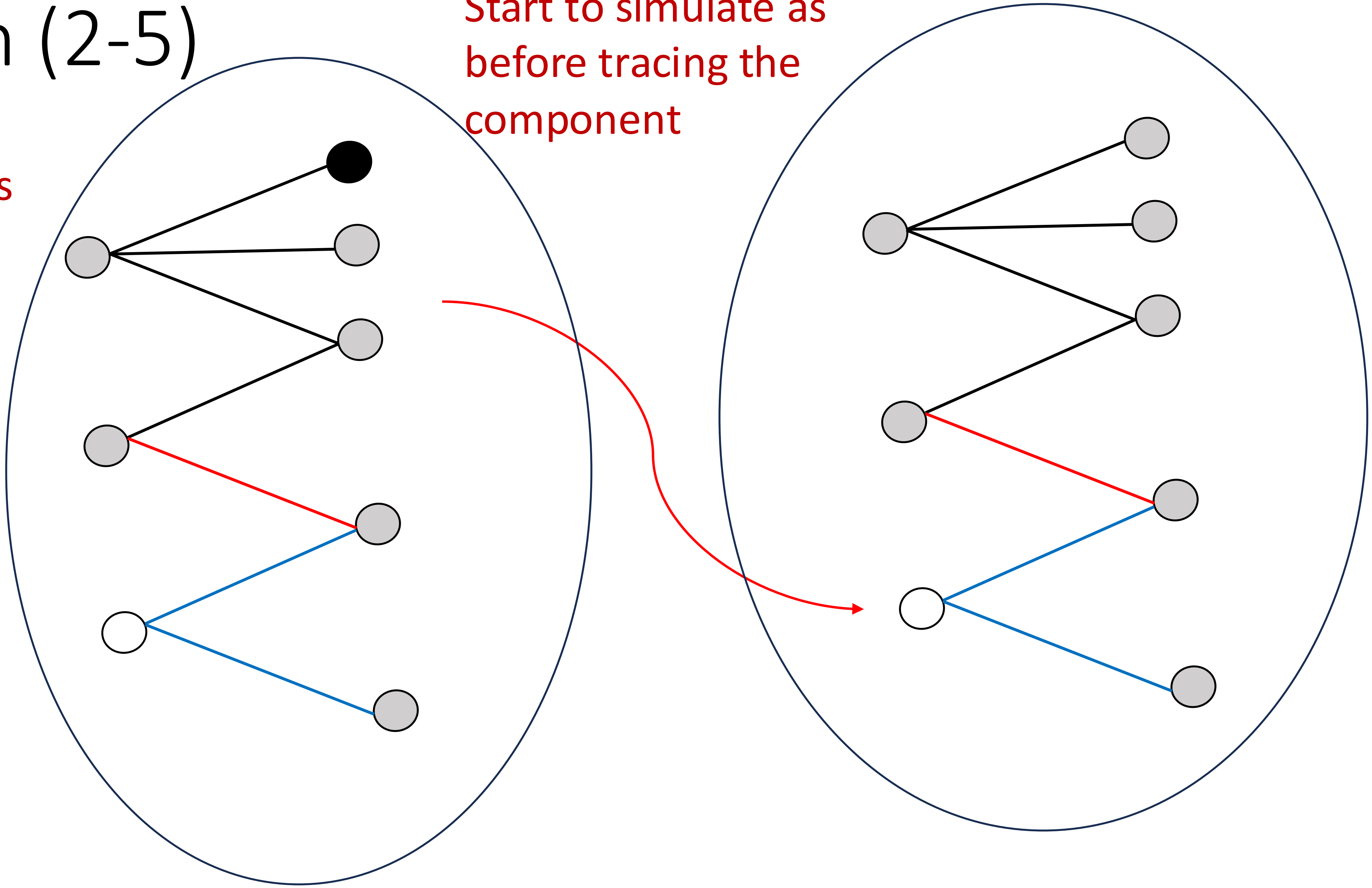


Type-3 line induced from intersecting type-2 lines

Line Graph (2-5)

All Components related to the query are revealed

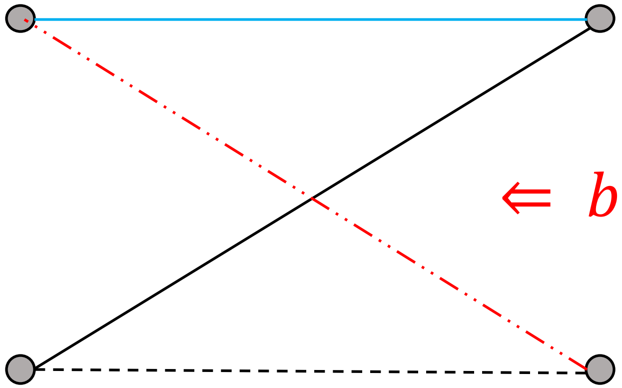
Start to simulate as before tracing the component



Open Problems

- What are Other 2n bit Ideal Cipher Construction based on n bit ideal?
- Can we make 5 calls? Or 6 is minimum?
- Progress on this: Characterized all constructions and found 6 is minimum.

- The Current bound is O($q^{13}/2^n$). We can have an attack in O($2^{n/4}$). Not Tight. Improve the Proof or Attack.

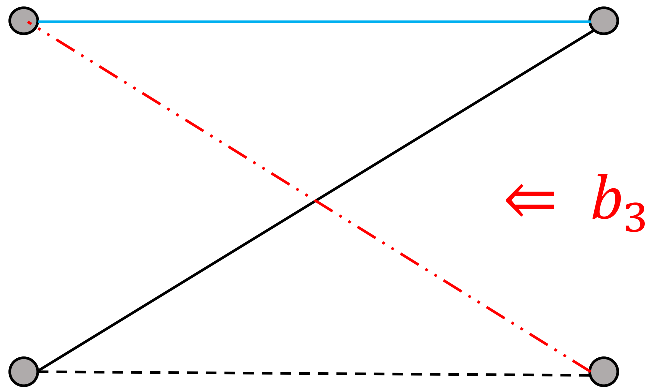


$\Leftarrow\ b_3 \oplus b_4 \oplus b_3' \oplus b_4' = 0$

Open Problems

- What are Other 2n bit Ideal Cipher Construction based on n bit ideal?
- Can we make 5 calls? Or 6 is minimum?
- Progress on this: Characterized all constructions and found 6 is minimum.

- The Current bound is O($q^{13}/2^n$). We can have an attack in O($2^{n/4}$). Not Tight. Improve the Proof or Attack.



$\Leftarrow\ b_3 \oplus b_4 \oplus b_3' \oplus b_4' = 0$

THANK YOU

Bad Events (Hitting).

1. For any fresh construction -query response (r or s) hits an existing point.
2. For any fresh primitive query ‘u’ on shore ‘i’, $F_i(u)$ is sampled such that there is an accidental collision in shore ‘i+1’.
3. At time of identifying Component, simulator reveals unqueried lines due to accident (hitting due to Simulator construction queries)

Bad Events (Guessing).

Simulator completes 2-5 components (and similarly others). Many inputs are not known to Adversary. Querying such point is guessing.

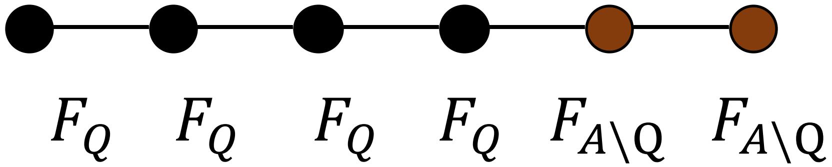
If No BAD, Simulator Friendly Line Graph is Preserved. $\Pr(BAD) \leq q^{12}/2^n$

F_Q is an Well-Approximation of (P_A, F_A)

- Suppose only revealed P-lines are complete

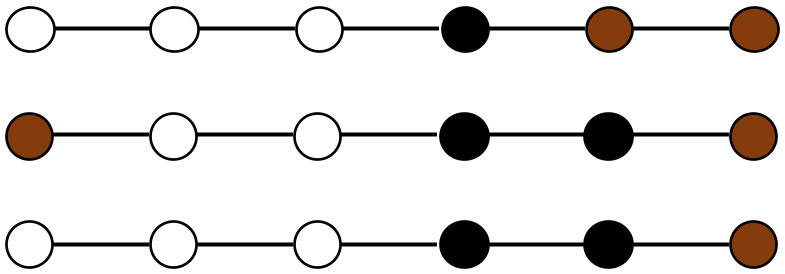
Point of $F_{A\setminus Q}$

✓ $p \in F_{A\setminus Q}$ must be in a complete line



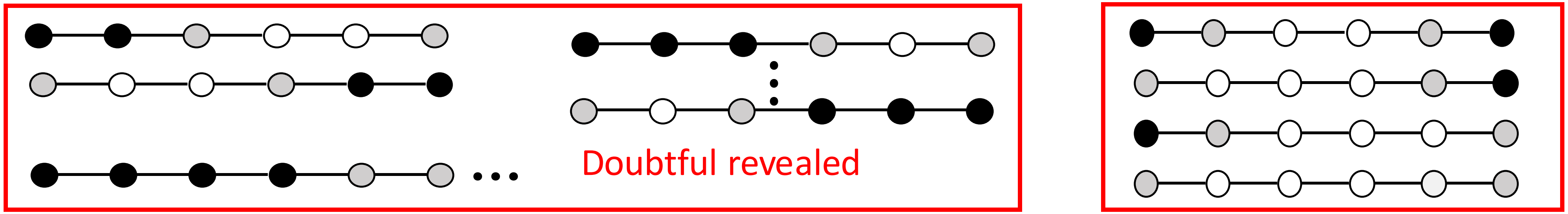
$$\boxed{|Q| = O(q) \Rightarrow |A| = O(q)}$$

X Not Allowed (Hitting Event)

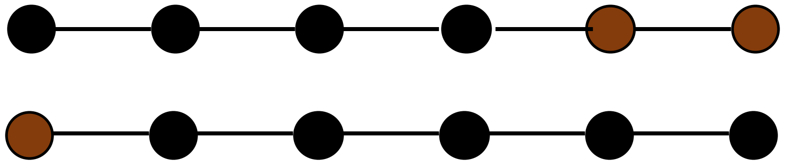


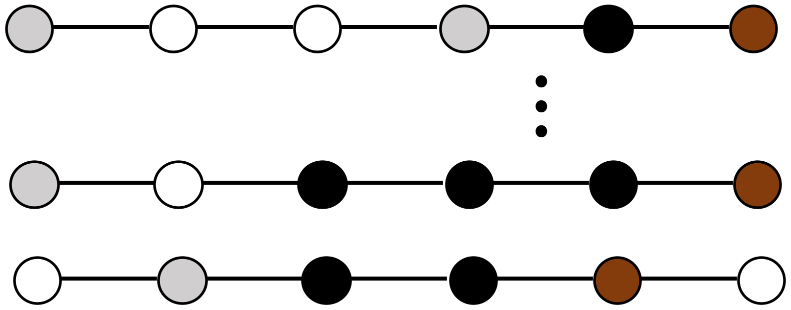
← It does not come with 5 Q points in a complete line and so input part is random

Collection of Lines



Point of $F_{A\backslash Q}$





Examples of P-lines using $F_{A\backslash Q}$

Examples of direct lines using $F_{A\backslash Q}$

Extension of Adversary Transcript: $(P_A, F_A) \rightarrow (P_E, F_E)$

- 4-Saturation: Saturate all Gen4+ making Construction queries

$$(P_A, F_A) \rightarrow (P_1, F_1)$$

- Boundary Closure: Add all Construction queries using two consecutive boundary F_1 points (at least one should be from the saturation)

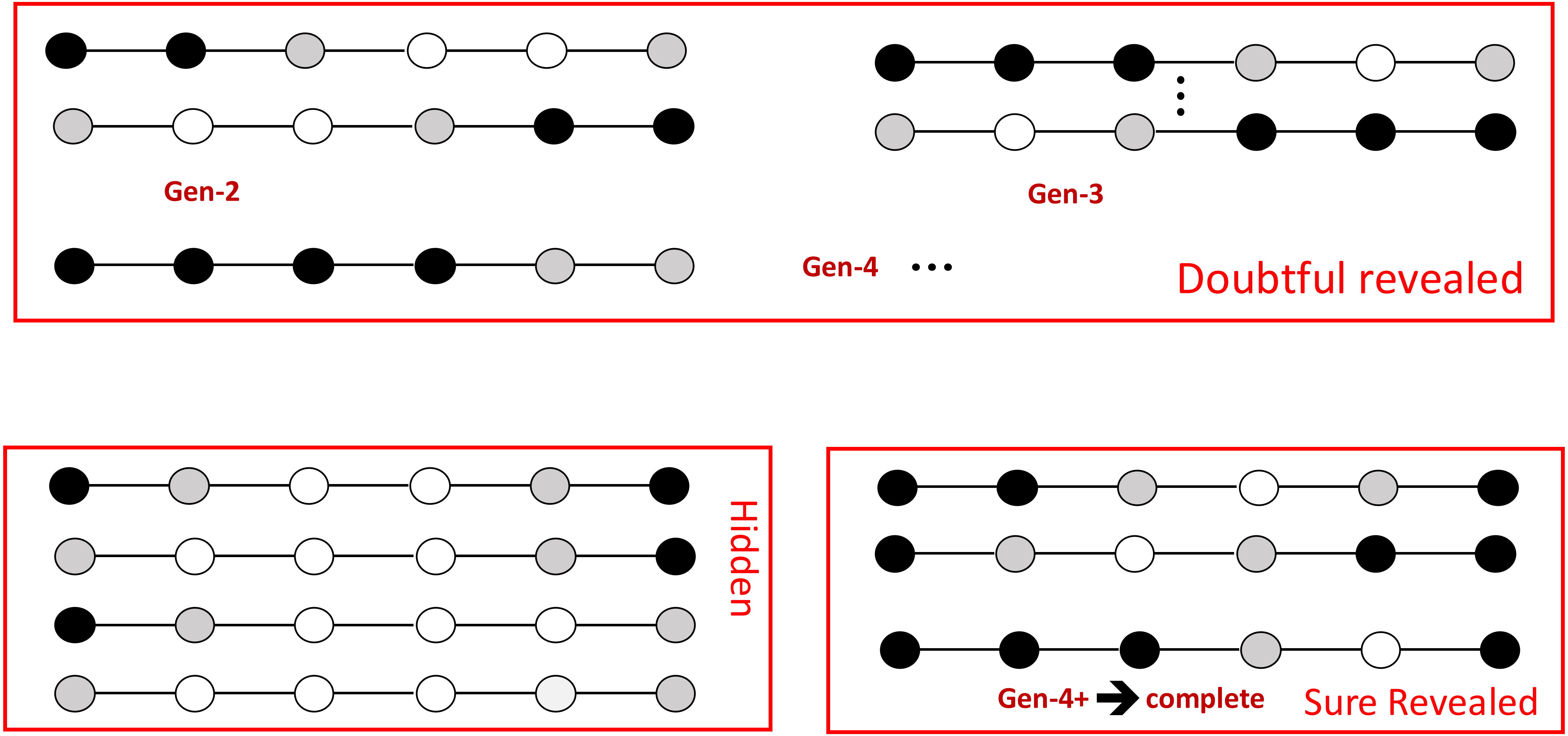
$$(P_1, F_1) \rightarrow (P_E, \boldsymbol{F_E = F_1}).$$

$$\boxed{|F_1| = O(q^2), \quad |P_E| = O(q^4)}$$

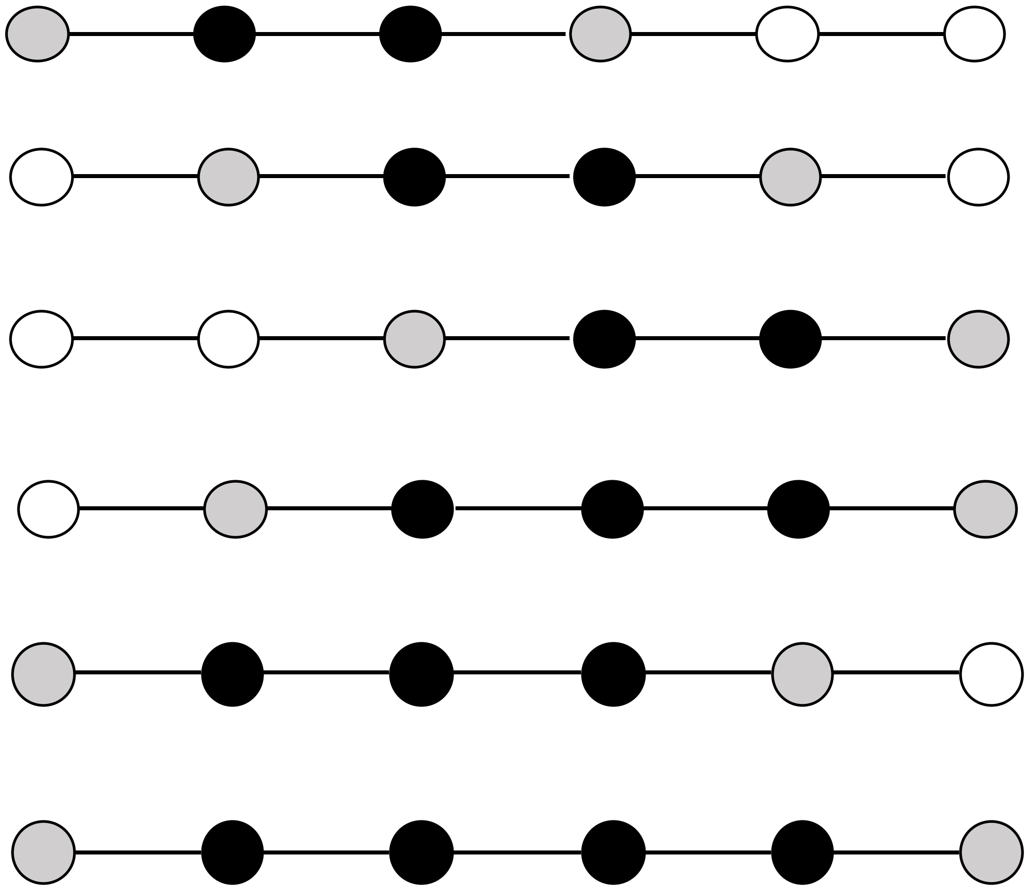
Simulator Transcript: (P_S, F_S)

- Captures F_1 completely.
- Captures $P_E \setminus P_{E,Hidden}$ completely.
- Maintains Incomplete Lines wrap-up and direct Gen-2 Lines.
  - (Gen-3+ → Complete).
- A point in $F_{S\setminus\mathrm{E}}$ in a complete line uniquely determined by at most three F_A points. Hence, $|F_S| = O(q^3)$
- Number of all Lines $= O(q^6)$.

Collection of P Lines



Collection of Internal Lines (Always Revealed)



Base Line System of Transcript (P, F)

- Every pair of consecutive F we have internal (direct) lines

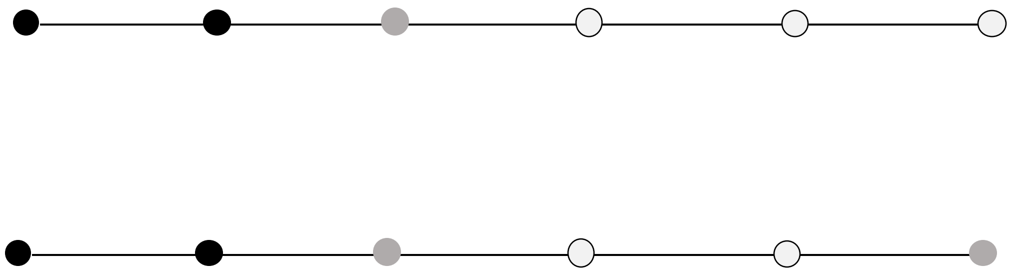
- Every P element we have a P-line (wrapped up line) either complete or at most Gen-3 (or Gen-2 and doubtful (or boundary) revealed lines)

- No two lines intersects more than one points

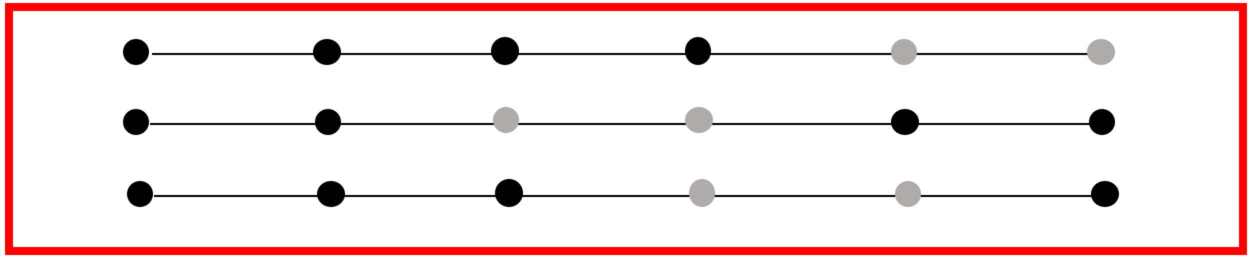
- Intersecting pair of type-2 lines induces type-3 lines. The only intersection at white points are type-3 vs type-3/2.

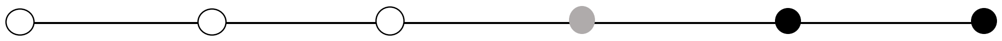
Simulation Steps

Forward Construction Query and then set as before for type-2 line

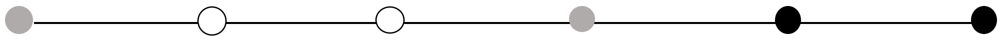


Options of Sampling

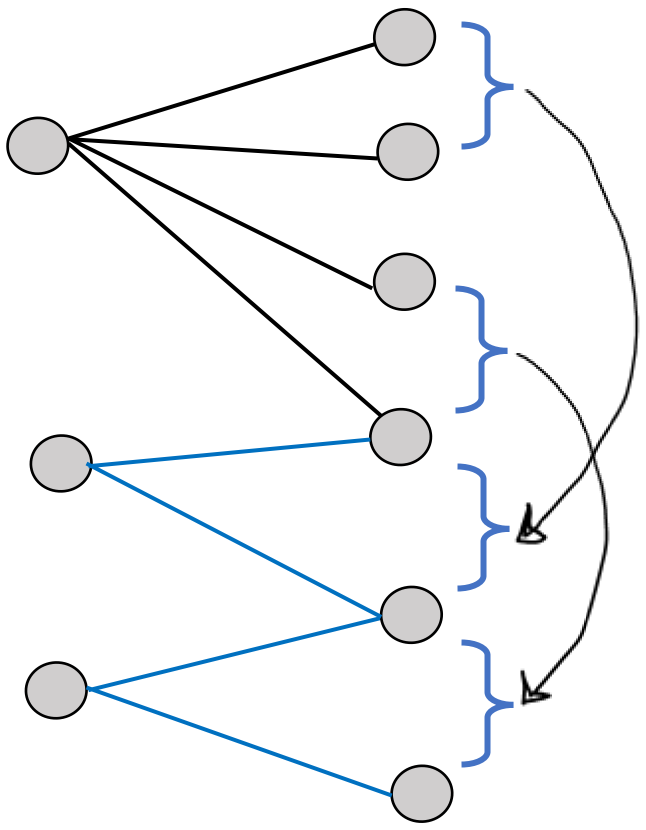




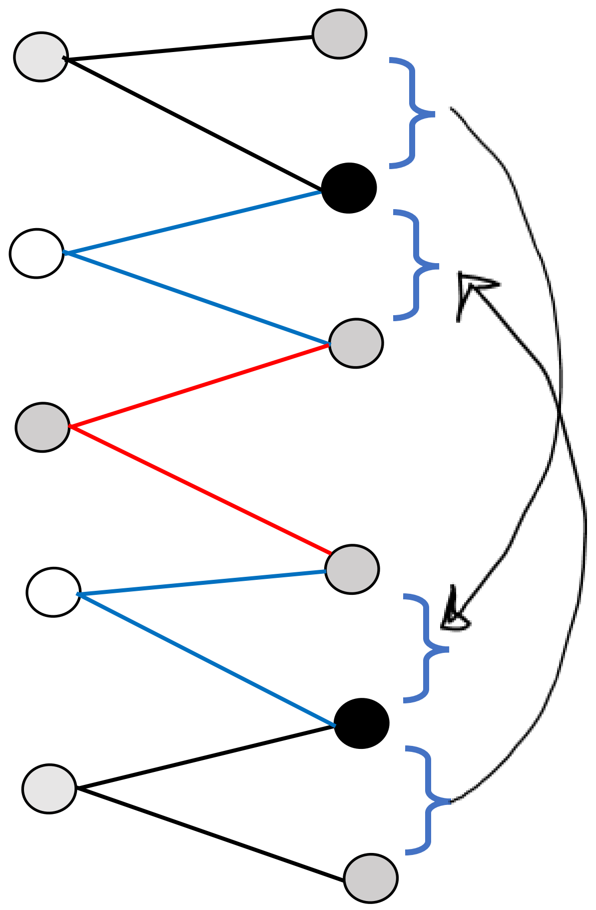
Similarly we first apply Backward Construction Query ➔



Examples



Example of Simulation Friendly



Example of Not Simulation Friendly

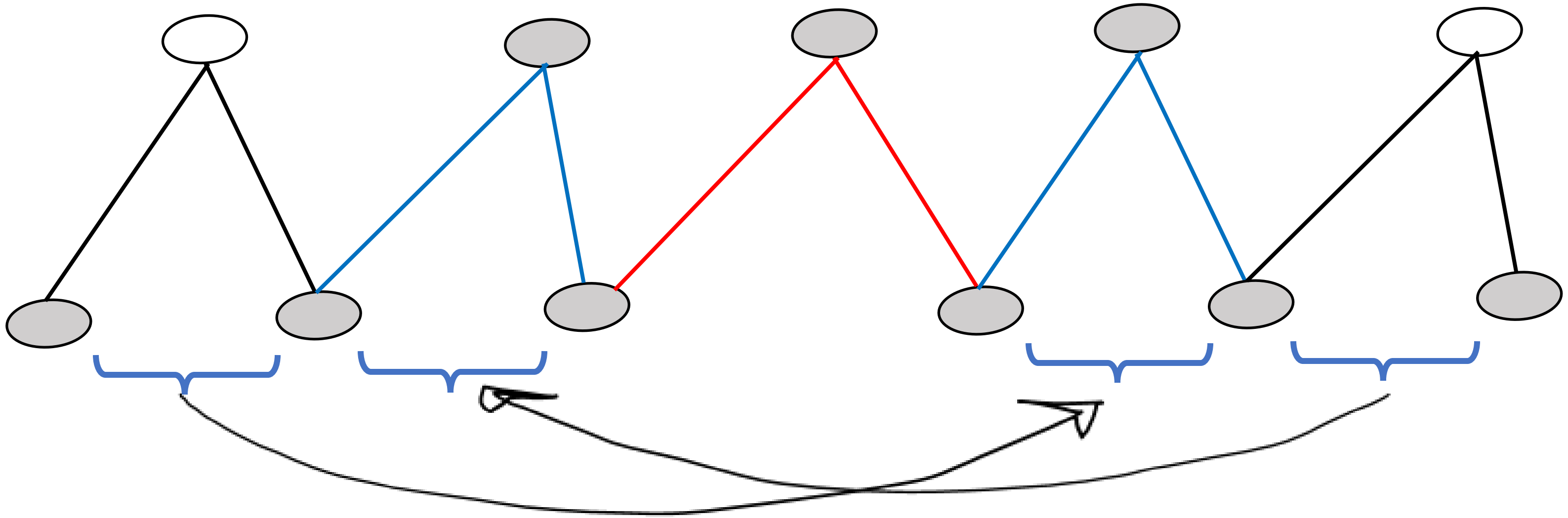
Simulation Friendly Line Graph

Ordering N on all lines in 2-5 line graph satisfying:

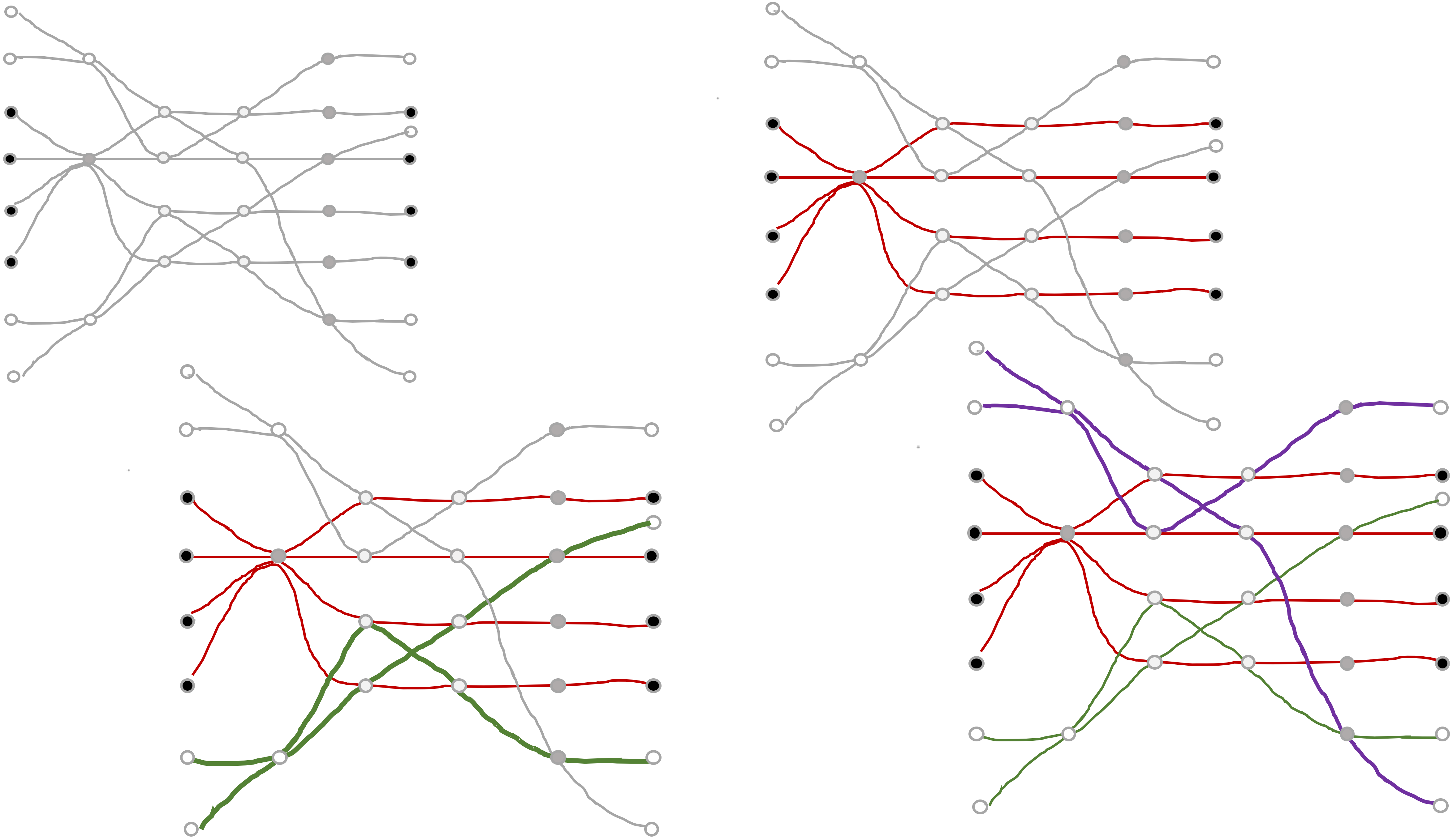
✓ **Topological ordering:** ℓ -- ℓ' -- ℓ'' $\Longrightarrow$ $N(\ell) < \max\{N(\ell'), N(\ell'')\}$

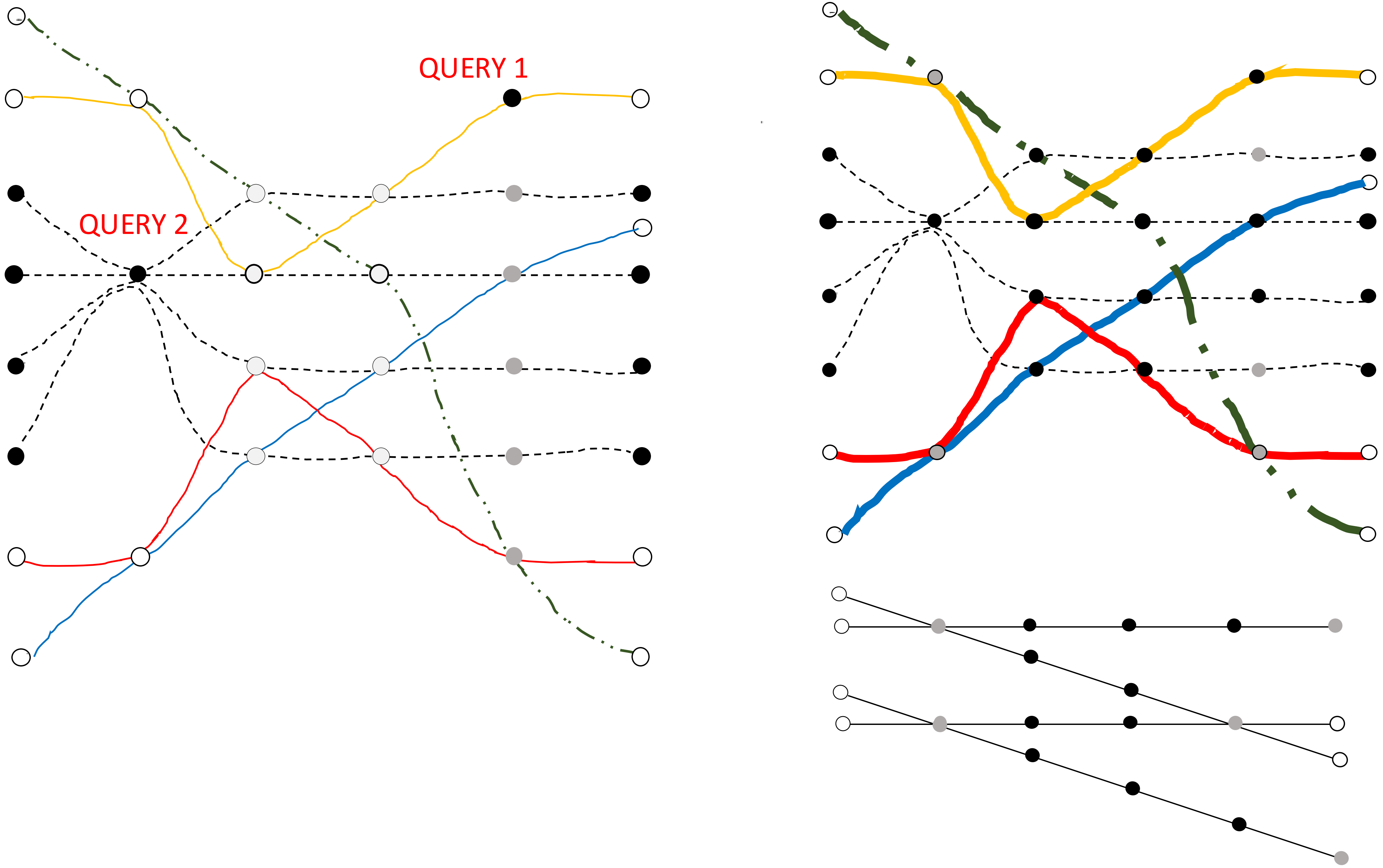
✓**Conjugate dependency**. If d is type-3 line of ℓ, ℓ' then $N(d) > N(\ell)$

We simulate all lines in the order of N



Example of Not Simulation Friendly





QUERY 1
QUERY 2