Generic Hardness of the Multiple Discrete Logarithm Problem

Aaram Yun

Ulsan National Institute of Science and Technology, South Korea

EUROCRYPT 2015

Background

Discrete logarithm problem

- p: prime
- G: cyclic group of order p
- $g \in G$: a generator of G
- Given $(p, G, g, h = g^{\alpha})$, find α

Group encoding

- $\xi:\mathbb{Z}_p \to \{0, 1\}^t$: an *encoding* of $\mathbb{Z}_p$
    - Injective function into some bitstrings
    - Concrete representation of group elements in $\mathbb{Z}_p$

DL is easy sometimes

- 'Somewhat' easy: subexponential algorithms like index calculus, number field sieve, ...
- Even easier: $G = (\mathbb{Z}_p, +)$, $g = 1$
- For a group, there can be good DL solvers on the group, exploiting the specific structure of the encoding

DL could be hard sometimes

- Some believe that DL on some carefully chosen elliptic curves is hard
- Proof?

DL is hard for dumb solvers

- It is known that DL is hard for *generic algorithms*
  - An algorithm on a group is *generic*, if it works for any encoding
  - Example: Baby-Step-Giant-Step
    - $O(p^{1/2})$ group operations to achieve some constant success probability

DL is hard for dumb solvers

- It is known that DL is hard for *generic algorithms*
  - An algorithm on a prime-order group is generic, if it works for any encoding
  - Example: Baby-Step-Giant-Step
    - This is optimal: $\Omega(p^{1/2})$ operations required to achieve constant success probability

Generic group model

- Proposed by Nechaev (1994, for DL) and Shoup (EUROCRYPT 1997, in general)
- Shoup, "*Lower bounds for discrete logarithms and related problems*", EUROCRYPT 1997

Generic group model

- In GGM, a prime-order group G is given via a *random encoding* $\xi : \mathbb{Z}_p \to \{0,1\}^t$
  - Group operations are done via oracle
  - Generic algorithms can be implemented in GGM

Generic group model

- Many cryptographically important problems have been studied in GGM
- Very often, tight lower bounds were proven
  - Essentially using only one standard technique, also proposed by Shoup

Multiple discrete logarithm problem

- p: prime
- G: cyclic group of order p
- $g \in G$: a generator of G
- Given $(p, G, g, g^{\alpha_1}, \ldots, g^{\alpha_n})$, find $\boldsymbol{\alpha}=(\alpha_1, \ldots, \alpha_n)$

MDL in GGM

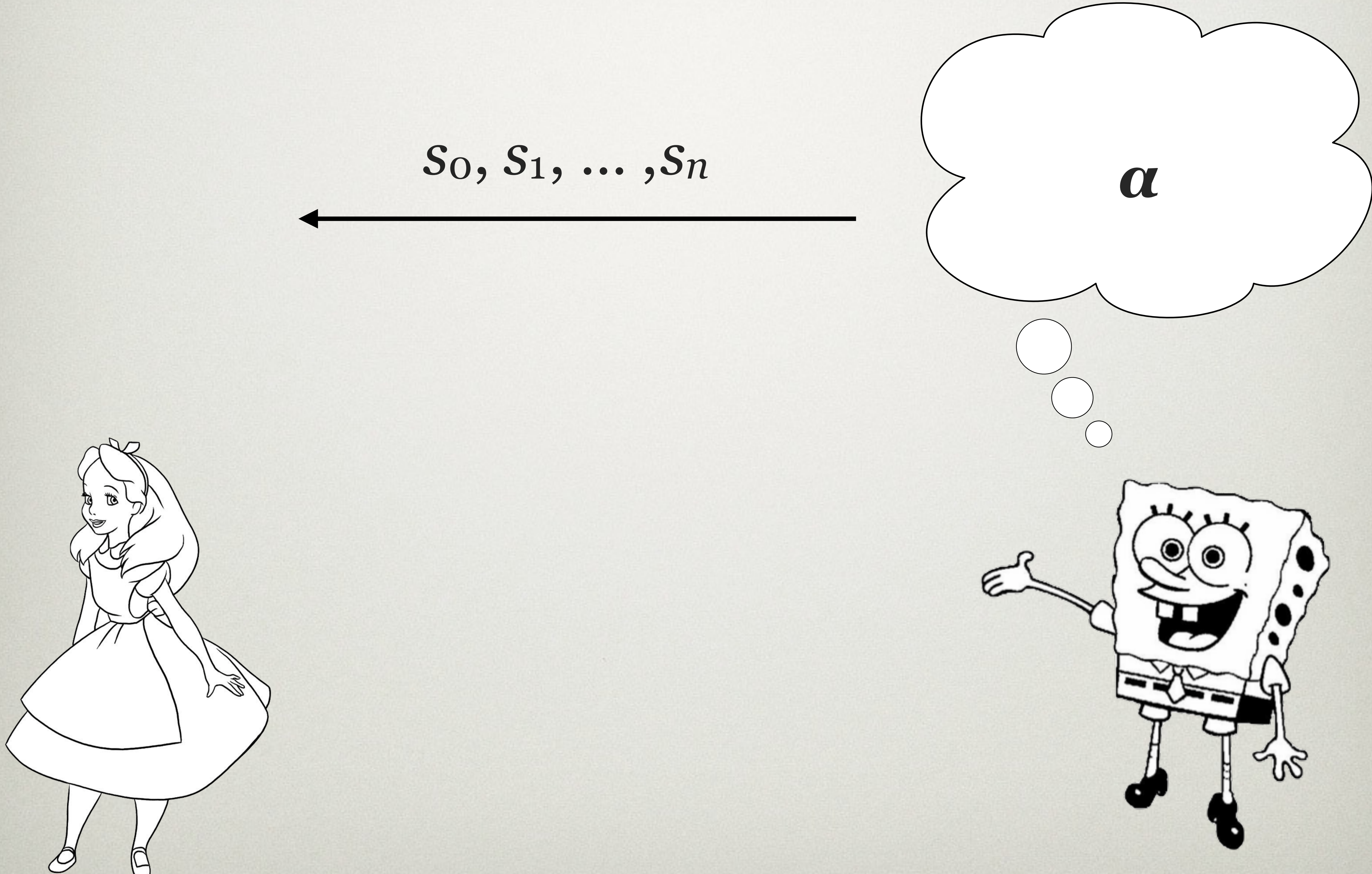
- ∃ a generic algorithm which solves MDL in $O((np)^{1/2})$ group operations
  - Kuhn and Struik, SAC 2001
- Shoup's technique gives only a trivial lower bound of $\Omega(p^{1/2})$
  - Rare exception where the standard technique fails to give a tight bound

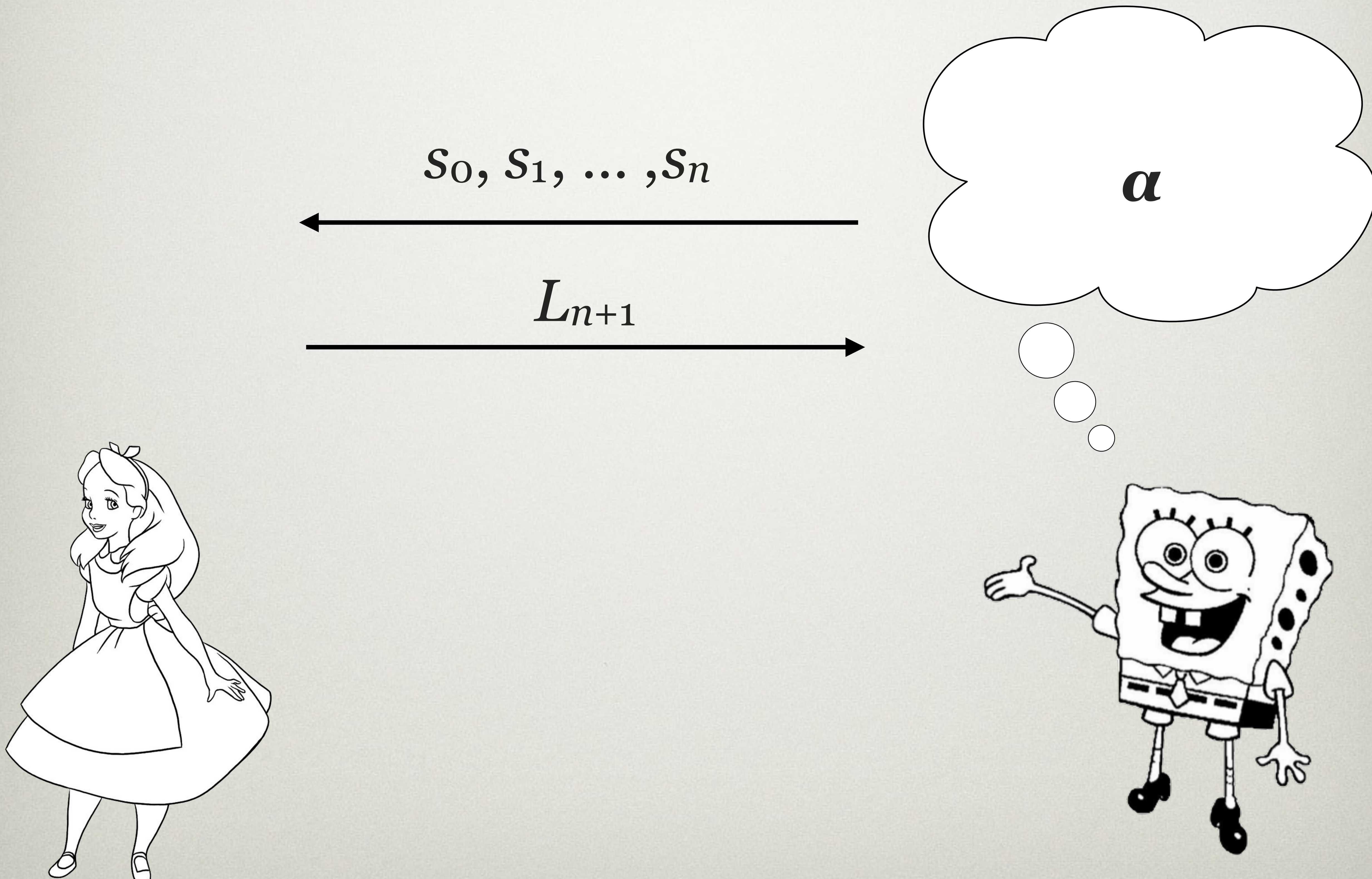
MDL in GGM

$$\boldsymbol{\alpha}=(\alpha_1, \ldots, \alpha_n)$$

$$L_0=1$$
$$L_1=X_1$$
$$\vdots$$
$$L_n=X_n$$

$$s_0=\xi(L_0(\boldsymbol{\alpha}))=\xi(1)$$
$$s_1=\xi(L_1(\boldsymbol{\alpha}))=\xi(\alpha_1)$$
$$\vdots$$
$$s_n=\xi(L_n(\boldsymbol{\alpha}))=\xi(\alpha_n)$$


$s_0, s_1, \ldots, s_n$
$\boldsymbol{\alpha}$

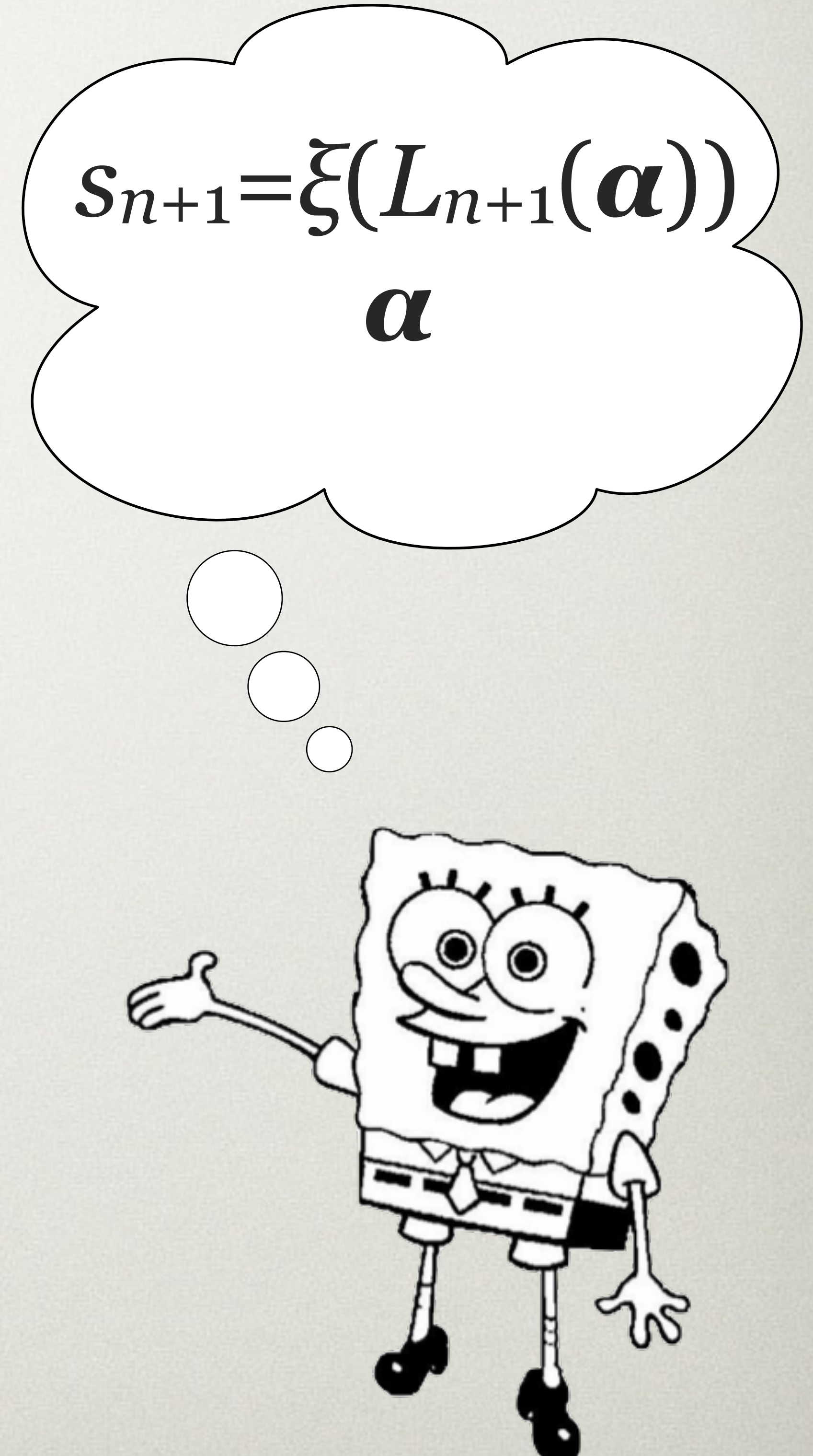

$s_0, s_1, \ldots, s_n$
L_{n+1}
α

$s_0, s_1, \ldots, s_n$

$\longleftarrow$

L_{n+1}

$\longrightarrow$



$$s_0, s_1, \ldots, s_n$$

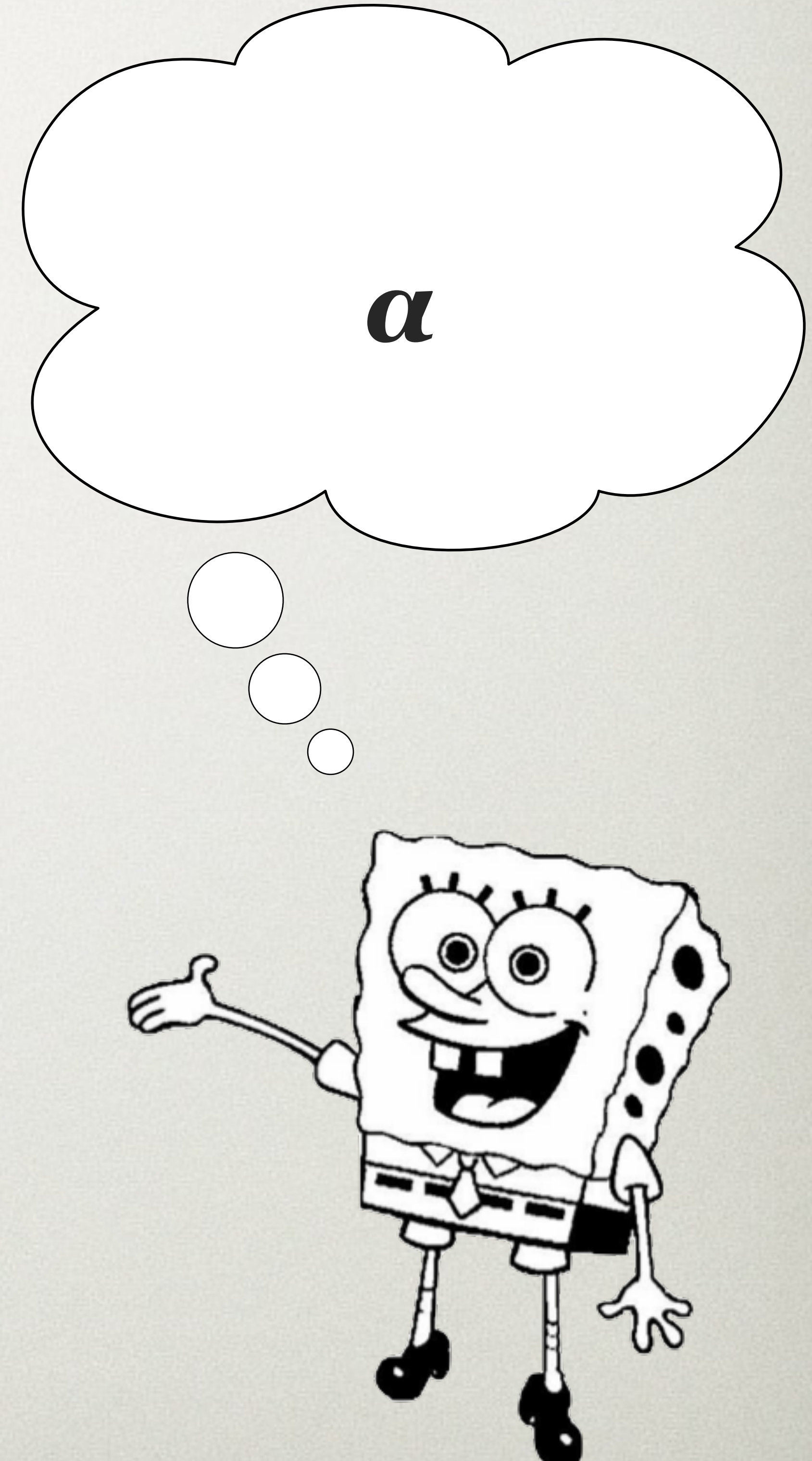
$$\longleftarrow$$

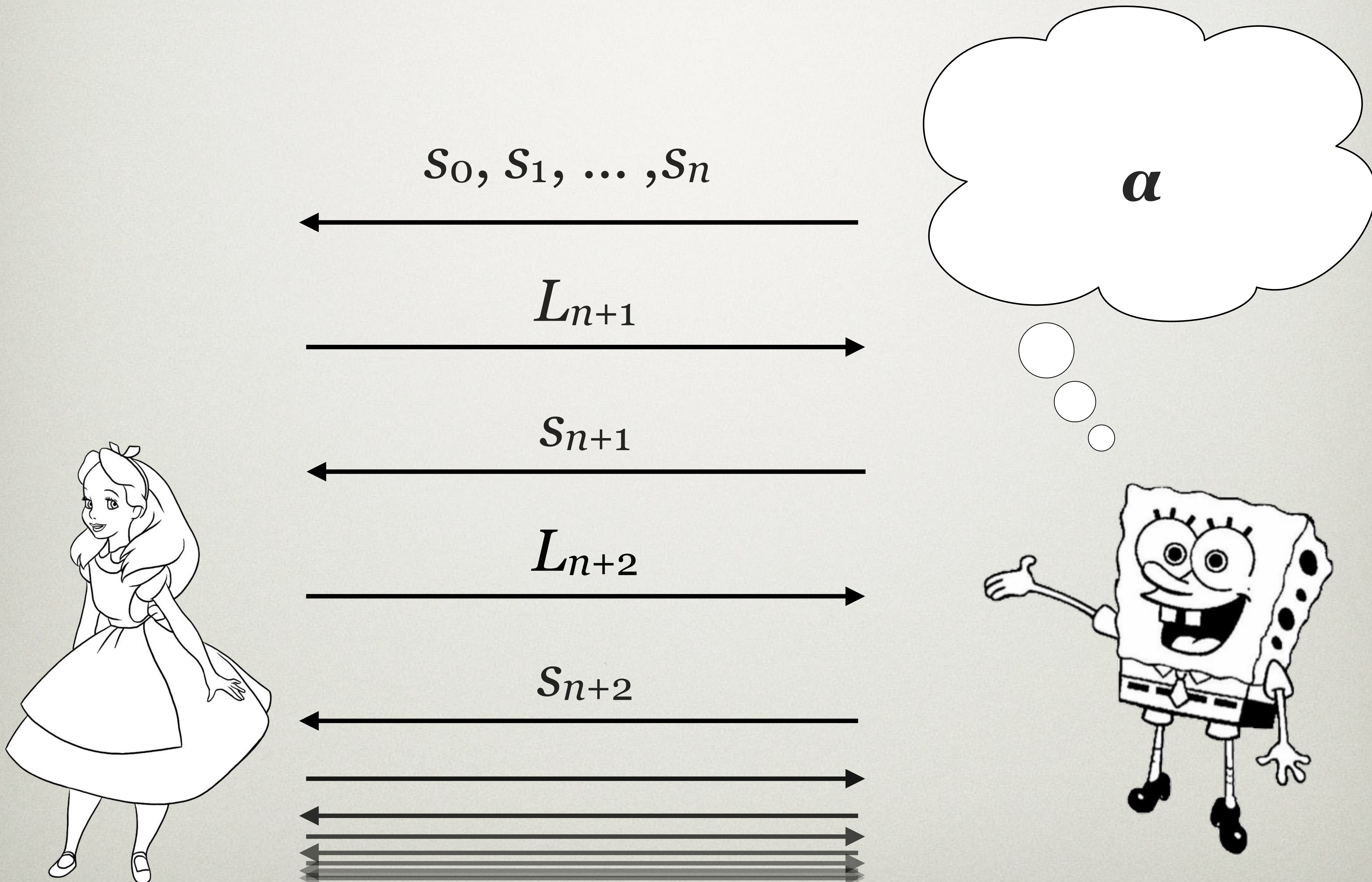
$$L_{n+1}$$

$$\longrightarrow$$

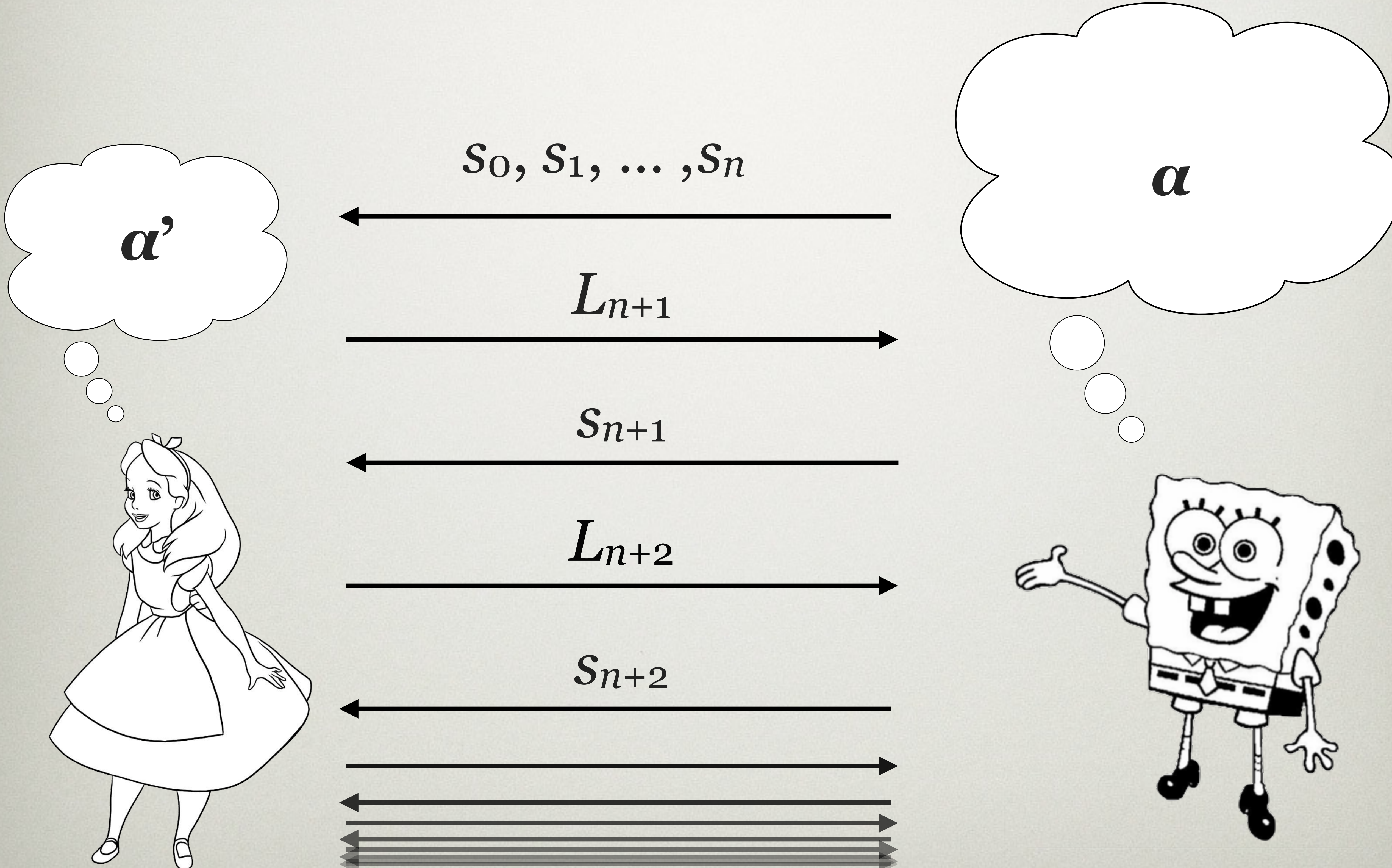
$$s_{n+1}$$

$$\longleftarrow$$





$s_0, s_1, \ldots, s_n$
α
L_{n+1}
s_{n+1}
L_{n+2}
s_{n+2}



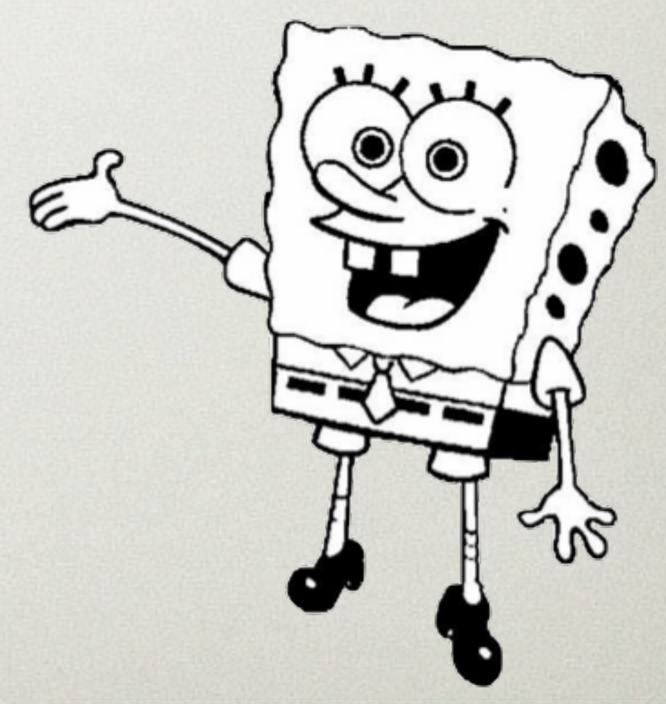
α'
$s_0, s_1, \ldots, s_n$
α
L_{n+1}
s_{n+1}
L_{n+2}
s_{n+2}

Shoup's technique applied to MDL

$$s_i = \xi(L_i(\boldsymbol{\alpha}))$$

$$\boldsymbol{\alpha}$$

$$\xrightarrow{\;\;L_i\;\;}$$



```
if
```
$L_i(\boldsymbol{\alpha})=L_j(\boldsymbol{\alpha})$ `for` $\exists j<i$

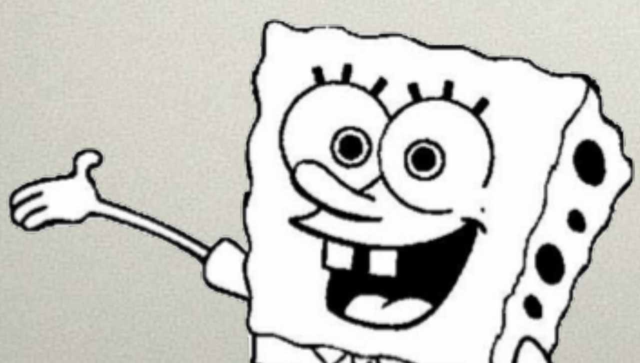
$s_i \leftarrow s_j$

```
else
```

$s_i \leftarrow \{0,1\}^t \setminus \{s_0, \cdots, s_{i-1}\}$

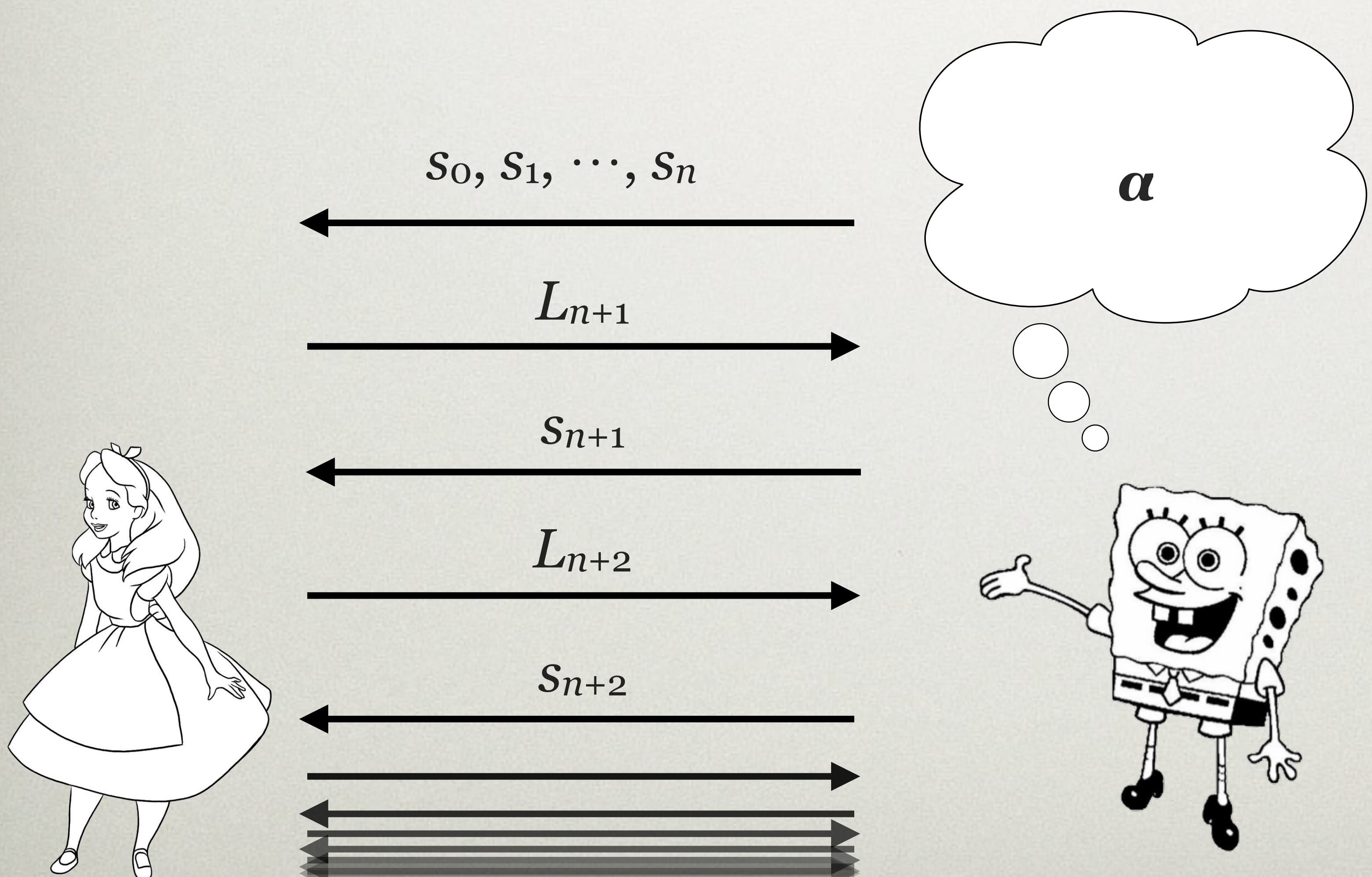
$s_i=\xi(L_i(\boldsymbol{\alpha}))$

$\boldsymbol{\alpha}$

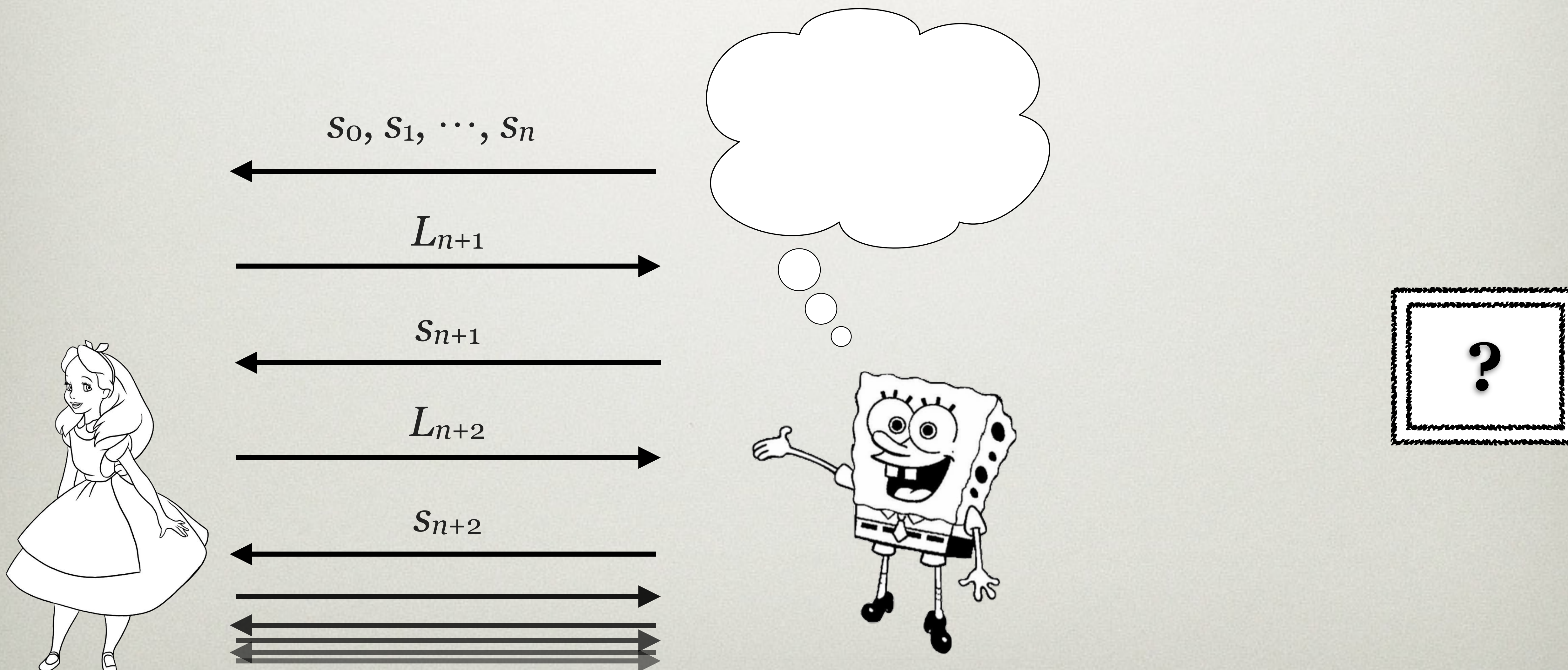


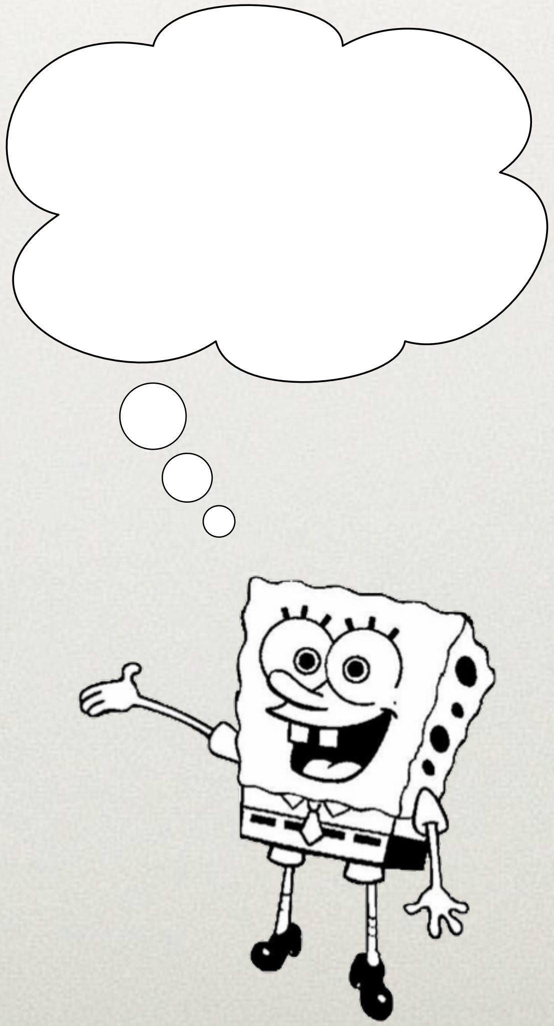
Shoup's technique

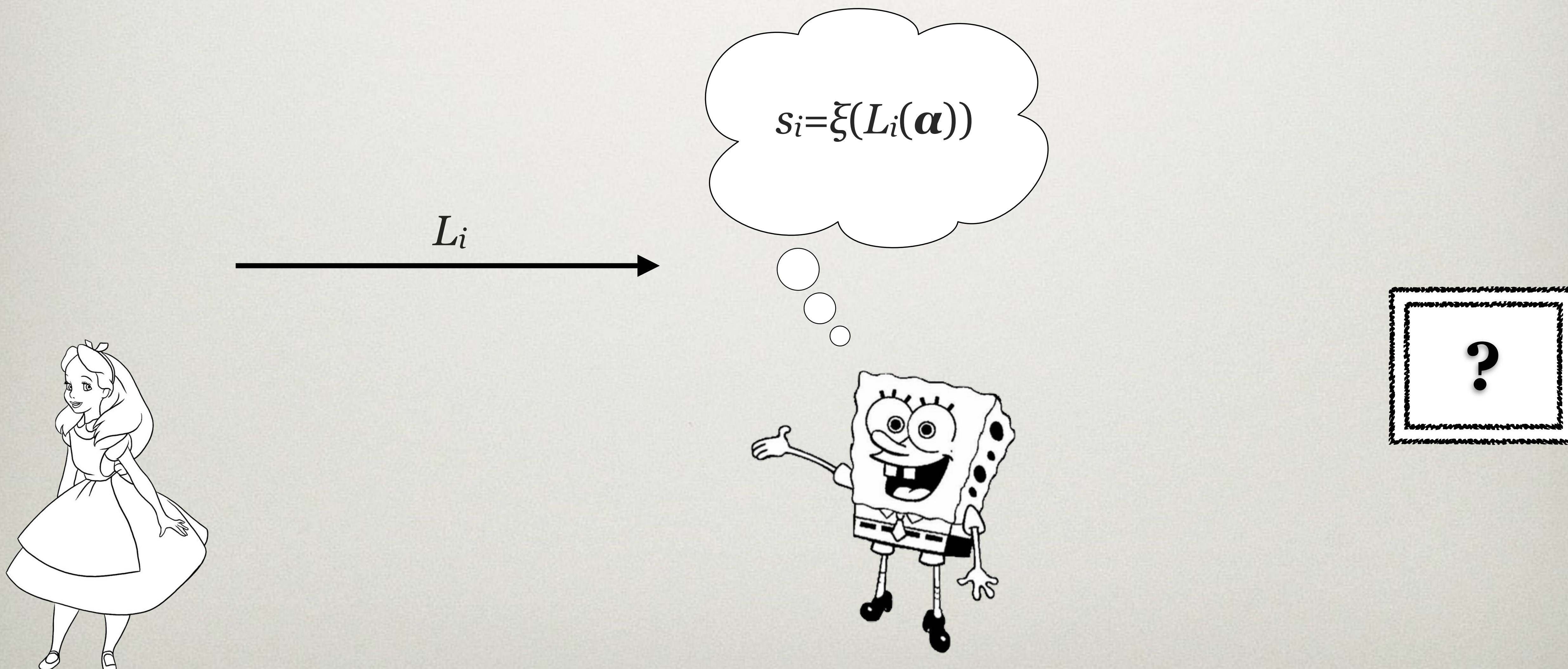
- Game G0: the game describing the original problem
- Game G1: modified game where secret exponents are chosen *at the end*
- Proving G1 is hard is trivial
- Difference between G0 & G1: Schwartz-Zippel lemma



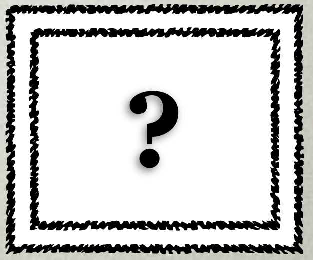
$s_0, s_1, \cdots, s_n$
L_{n+1}
s_{n+1}
L_{n+2}
s_{n+2}
α


$s_0, s_1, \cdots, s_n$
L_{n+1}
s_{n+1}
L_{n+2}
s_{n+2}
?



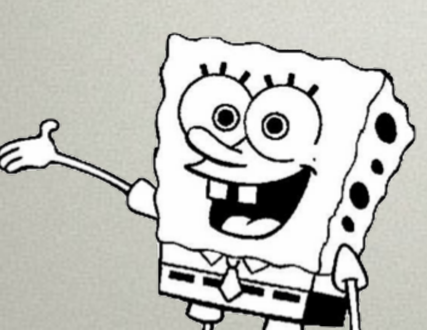

$s_i=\xi(L_i(\boldsymbol{\alpha}))$
L_i
?


$s_i=\xi(L_i(\boldsymbol{\alpha}))$


?

```
if $L_i(\boldsymbol{\alpha})=L_j(\boldsymbol{\alpha})$ for $\exists j<i$
        $s_i \leftarrow s_j$
else
        $s_i \leftarrow \{0,1\}^t \setminus \{s_0, \cdots, s_{i-1}\}$
```

$s_i=\xi(L_i(\boldsymbol{\alpha}))$



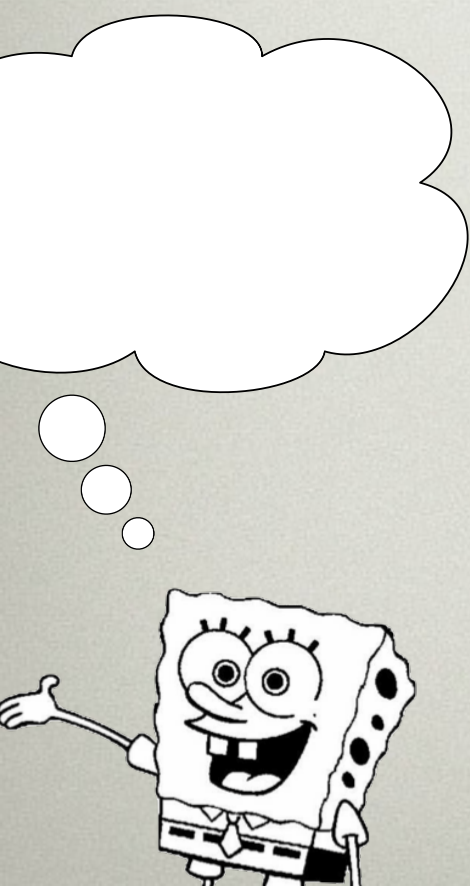
?

if $L_i = L_j$ for $\exists j<i$

$s_i \leftarrow s_j$

else

$s_i \leftarrow \{0,1\}^t \setminus \{s_0, \cdots, s_{i-1}\}$



?

G0 & G1

- G0 and G1 differ iff
  $\exists i<j,\ L_i \neq L_j \wedge L_i(\boldsymbol{\alpha})=L_j(\boldsymbol{\alpha})$
- $\Pr[\exists i<j,\ L_i \neq L_j \wedge L_i(\boldsymbol{\alpha})=L_j(\boldsymbol{\alpha})]$
  $\leq (q+n+1)^2/2p$
- Success probability for G1: $1/p^n$

Shoup's technique for MDL

- Success probability of a solver $\leq p^{-n} + (q+n+1)^2/2p$
- Meaningless if $q=p^{1/2}$
  - But we want $q=\Omega(n^{1/2}p^{1/2})$

MDL with hyperplane queries

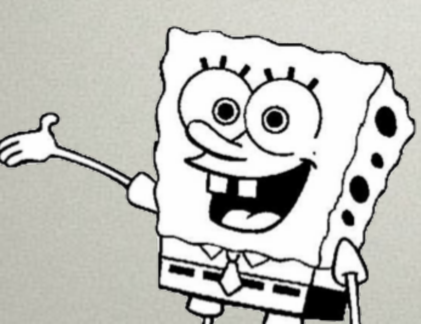
if $L_i(\boldsymbol{\alpha})=L_j(\boldsymbol{\alpha})$ for $\exists j<i$

$s_i \leftarrow s_j$

else

$s_i \leftarrow \{0,1\}^t \setminus \{s_0, \cdots, s_{i-1}\}$

$s_i=\xi(L_i(\boldsymbol{\alpha}))$



?

Is $L_i(\boldsymbol{\alpha})=L_j(\boldsymbol{\alpha})$?

if for $\exists j<i$

$s_i \leftarrow s_j$

else

$s_i \leftarrow \{0,1\}^t \setminus \{s_0, \cdots, s_{i-1}\}$

?

if for $\exists j<i$

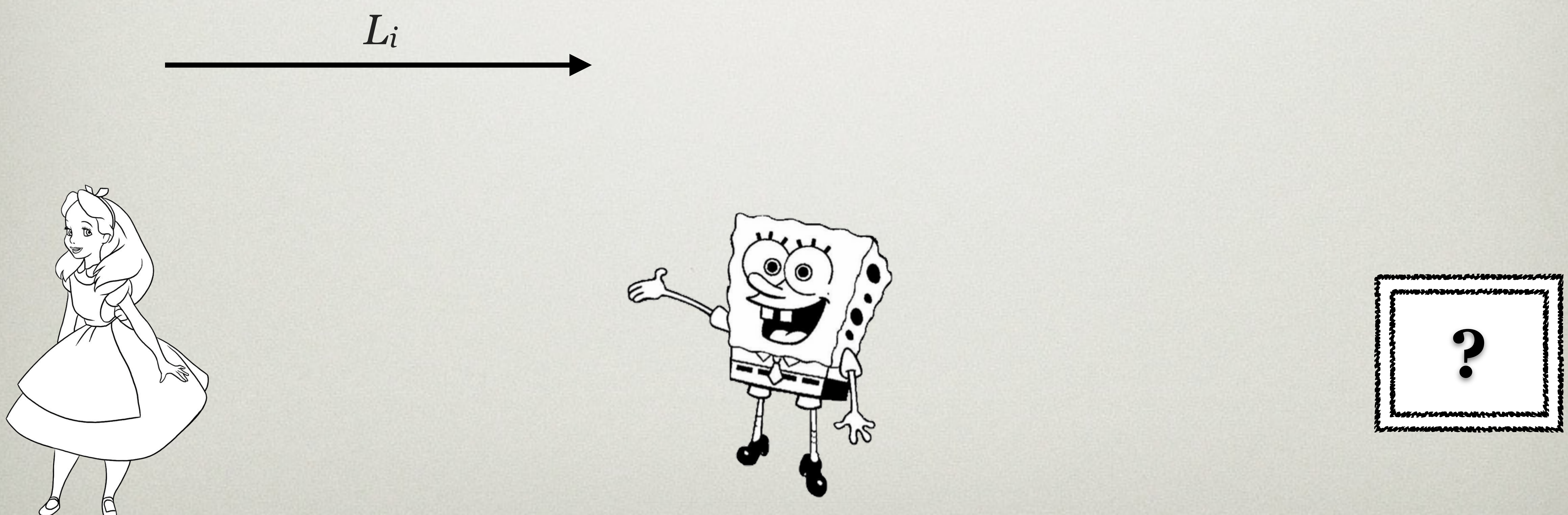
$s_i \leftarrow s_j$

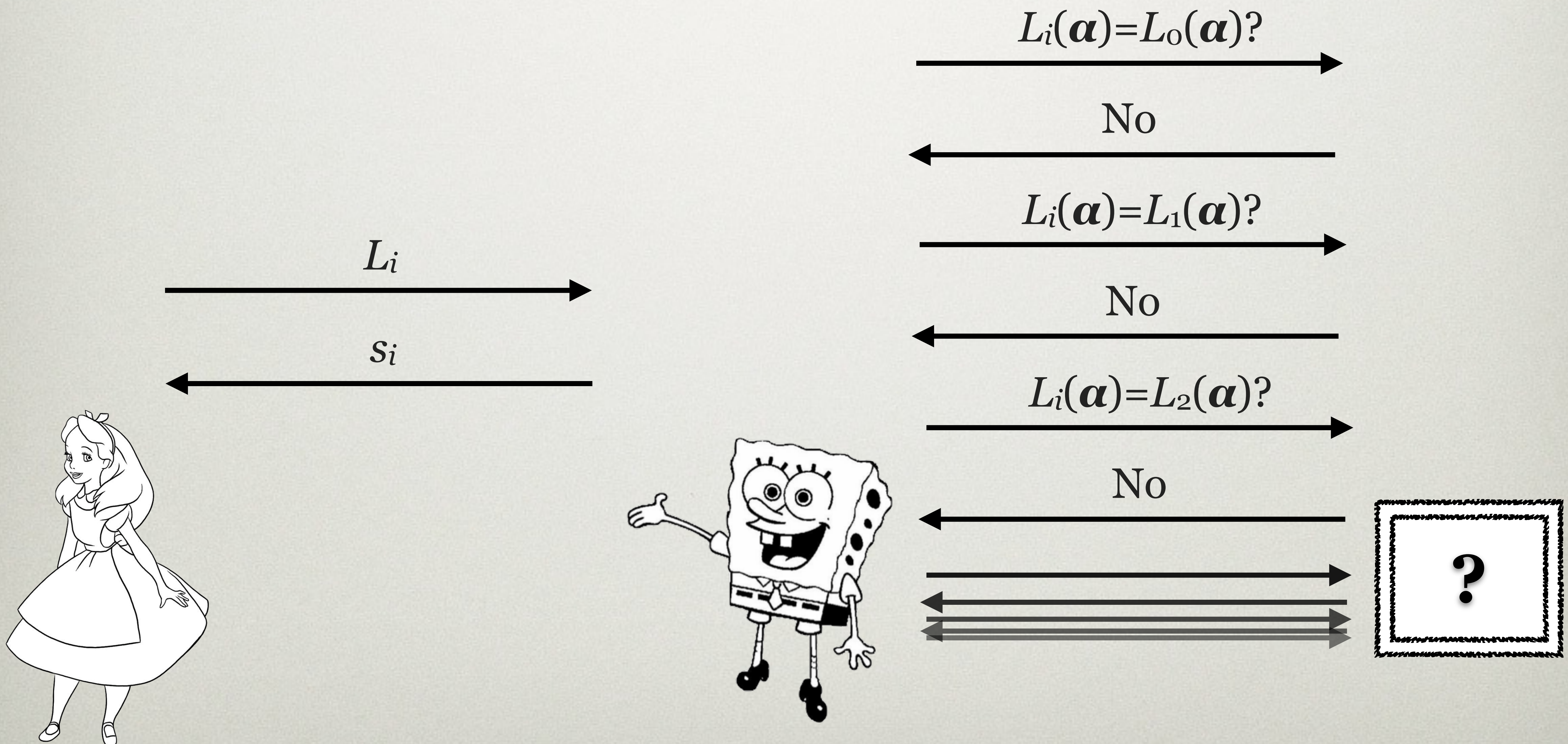
else

$s_i \leftarrow \{0,1\}^t \setminus \{s_0, \cdots, s_{i-1}\}$

Yes

?


L_i
?


L_i
s_i
$L_i(\boldsymbol{\alpha})=L_0(\boldsymbol{\alpha})$?
No
$L_i(\boldsymbol{\alpha})=L_1(\boldsymbol{\alpha})$?
No
$L_i(\boldsymbol{\alpha})=L_2(\boldsymbol{\alpha})$?
No
?

SHQ problem

- Search-by-Hyperplane-Queries (SHQ)
  - Correctly guess a hidden point $\boldsymbol{\alpha} \in \mathbb{Z}_p{}^n$
  - The solver can make up to q adaptive *hyperplane queries*
    - "Is $\boldsymbol{\alpha} \in H$?" for a hyperplane $H \subseteq \mathbb{Z}_p{}^n$
    - A hyperplane H can be described by an equation $a_1 X_1 + \ldots + a_n X_n = b$

MDL with hyperplane queries

- MDL game can be simulated perfectly, if the challenger has ability to decide if the hidden exponents $\boldsymbol{\alpha}=(\alpha_1, \ldots, \alpha_n)$ lie on a given hyperplane *H* or not
- Any MDL solver *A* can be turned into a SHQ solver *B* with the same success probability
  - No. of queries: $q \to (q+n+1)^2/2$

MDL via SHQ

- Any lower bound for SHQ yields a lower bound for MDL
- Lower bound $q=\Omega(np)$ for SHQ → lower bound $q=\Omega((np)^{1/2})$ for MDL

Search by Hyperplane Queries (SHQ)

Twenty questions

- "Is it an animal?"
- If each question reduces the search space by half, then you can find the hidden point within
  $q = \log_2 p^n = n \log_2 p$ queries

Twenty questions

- A hyperplane query is a terribly bad question to ask in a game of twenty questions
  - Too thin!

Brute-force solver

- A 'brute-force' SHQ solver
  - Makes queries of form $X_i = j$ for $i = 1, \ldots, n$ and $j = 1, \ldots, p-1$
  - If $\boldsymbol{\alpha}$ is on $X_i = j$ for some j, then the ith coordinate of $\boldsymbol{\alpha}$ is j
  - Otherwise, the ith coordinate is 0
  - $q = n(p-1)$ is enough to find $\boldsymbol{\alpha}$ in the worst case

Main results

- We can show that $q = \Omega(np)$ in the average case
  - (Also, $q \geq n(p\text{-}1)$ in the worst case)

Twenty questions

- A: a SHQ solver making exactly q queries and deterministic
  - $\boldsymbol{H}=(H_1, \ldots, H_q)$: queries made by A
  - $\boldsymbol{b}=(b_1, \ldots, b_q)$: answers received by A
- Then, the success prob. is bounded by (no. of all possible $\boldsymbol{b}$s)$/p^n$
  - So far, nothing about hyperplanes

Intuition

- No. of possible ***b***s would be at most 2^q, so the prob. is bounded by $2^q/p^n$
- But, it would be very hard to get 1 as an answer: lower Hamming weight
  - No. of possible ***b***s should be much smaller than 2^q
  - But, is it?

Useless queries

- You can get easy, useless 1s, if that's what you want
  - Once you obtain a hyperplane query H with reply 1, then you can *repeat* the same query to get many more 1s
  - No useful info about $\boldsymbol{\alpha}$: meaningless 1s

Useless queries

- If so far A has asked $H_1, \ldots, H_r, H'_1, \ldots, H'_s$ and got 1s for $H_1, \ldots, H_r$, and 0s for $H'_1, \ldots, H'_s$, then A knows that $\boldsymbol{\alpha} \in \cap H_i - \cup H'_j$
- We say that at this point a query H of A is *useless*, if $\cap H_i - \cup H'_j \subseteq H$
  - i.e., when A can be certain that the answer must be 1 without asking

Useful queries

- WLOG, we may assume that A
  - Makes exactly q queries
  - Is deterministic
  - *And, never makes useless queries*
- This prevents A to obtain meaningless 1s as answers

Small Hamming weight

- For such a solver A, there can be only n 1s in the vector $\boldsymbol{b}=(b_1, \ldots, b_q)$

A little geometry

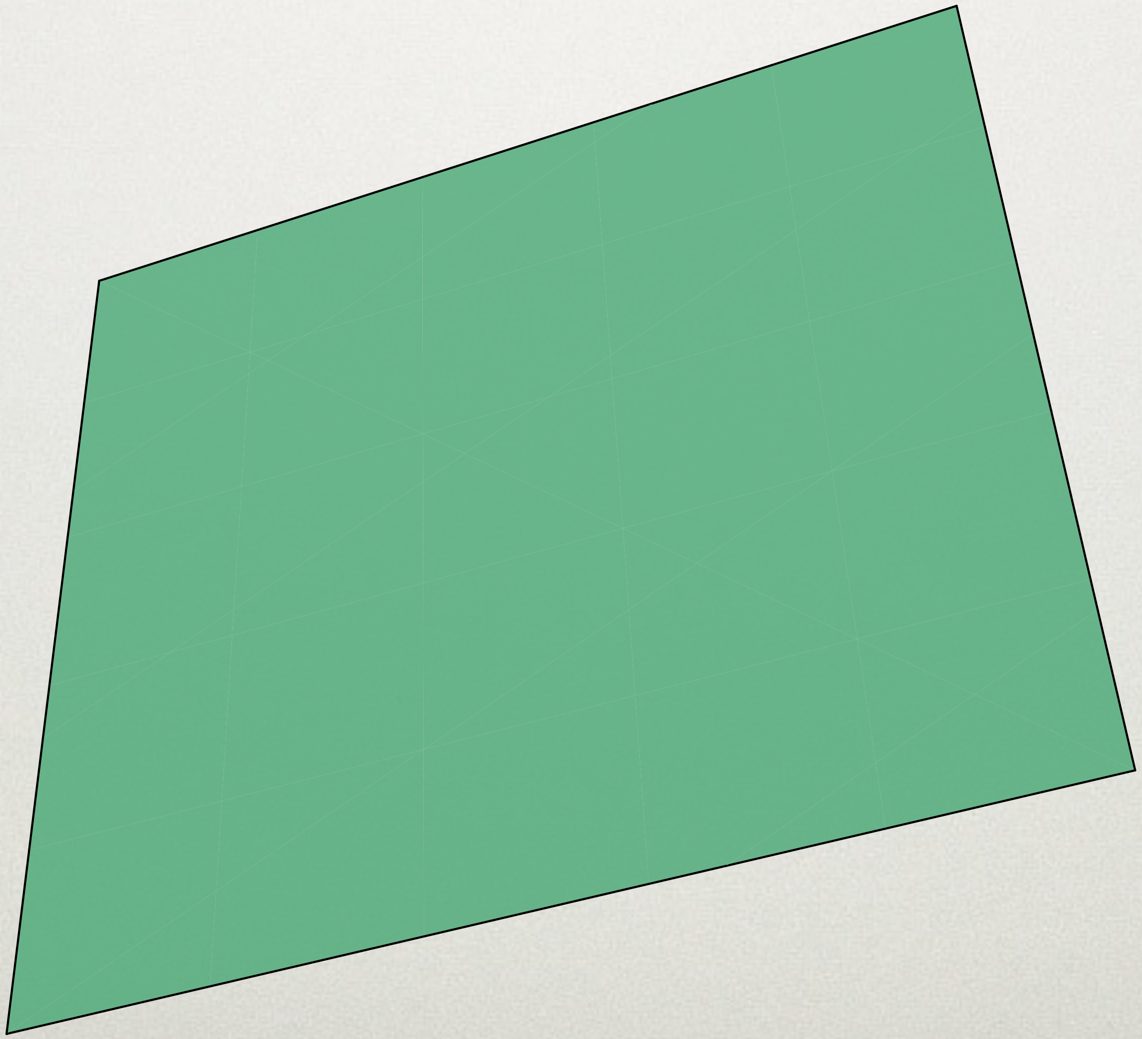
- Assume hyperplane queries H_1, ..., H_m with $\boldsymbol{\alpha} \in \cap H_i$
- Then, $H_1 \cap \cdots \cap H_i \nsubseteq H_{i+1}$ (H_{i+1} is not useless)

A little geometry

- Then, $H_1 \cap \cdots \cap H_i \nsubseteq H_{i+1}$

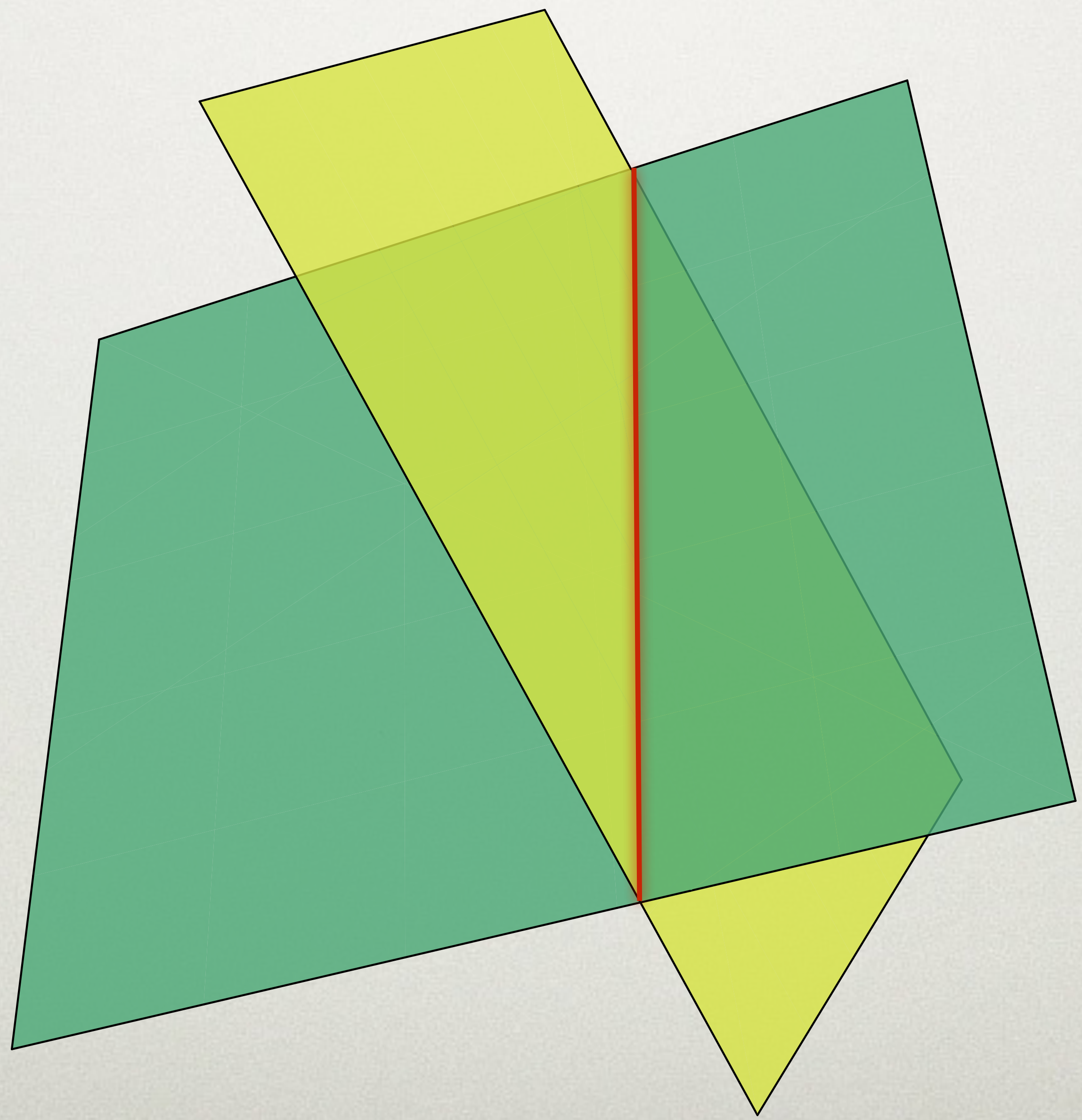
A little geometry

- Then, $H_1 \cap \cdots \cap H_i \nsubseteq H_{i+1}$



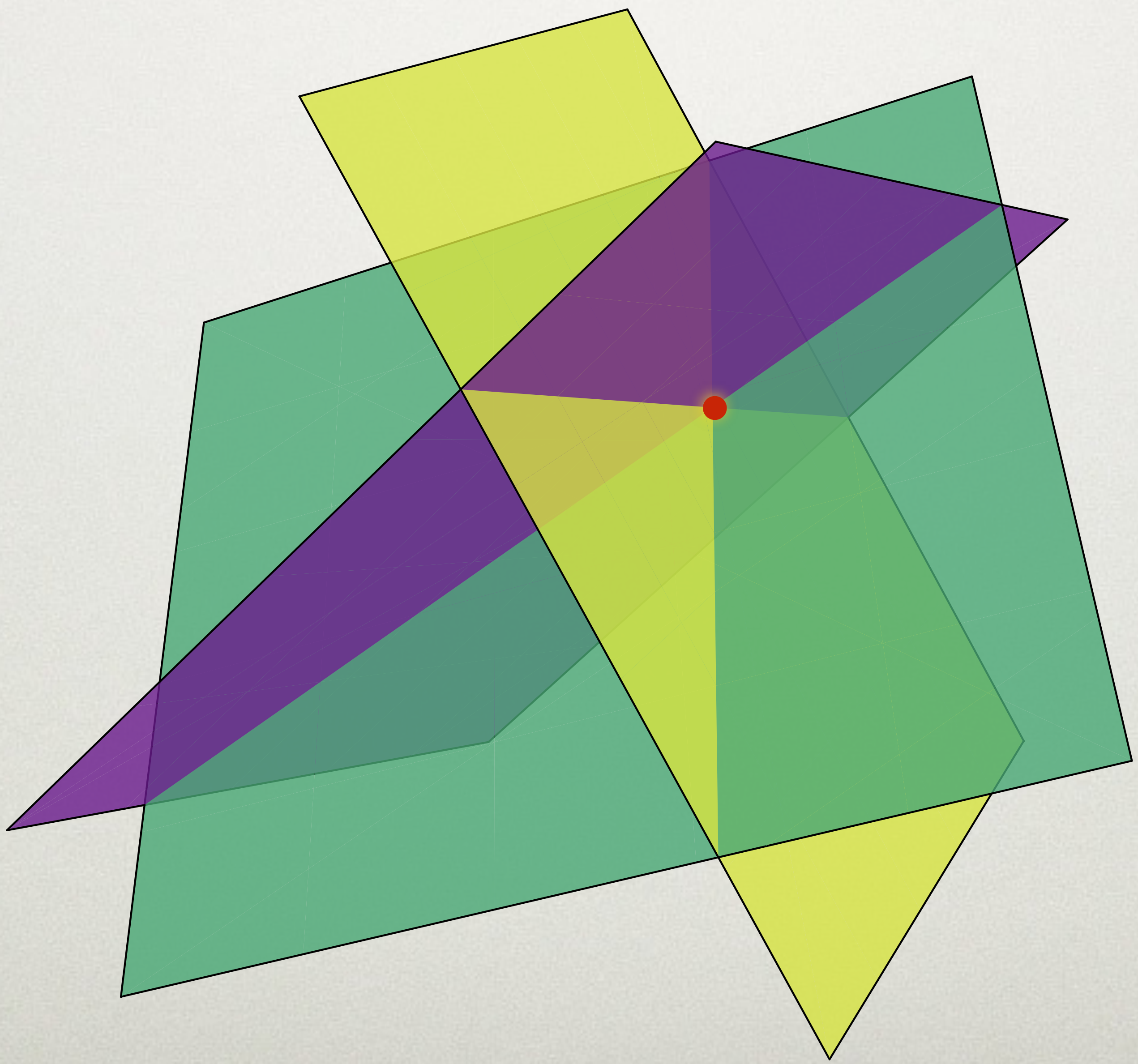
A little geometry

- Then, $H_1 \cap \cdots \cap H_i \nsubseteq H_{i+1}$



A little geometry

- Then, $H_1 \cap \cdots \cap H_i \nsubseteq H_{i+1}$



A little geometry

- Then, $H_1 \cap \cdots \cap H_i \not\subseteq H_{i+1}$
- Each additional hyperplane decrements the dimension of the intersection by 1
- So, $m \leq n$

Finally

- The success probability is bounded by

$$\frac{1}{p^n}\sum_{i=0}^{n}\binom{q}{i} \leq \frac{1}{p^n} + \frac{1}{2}\left(\frac{eq}{np}\right)^n$$

- So, to obtain some constant success probability, $q=\Omega(np)$ is needed
- This implies $q=\Omega((np)^{1/2})$ for MDL

Thank you!