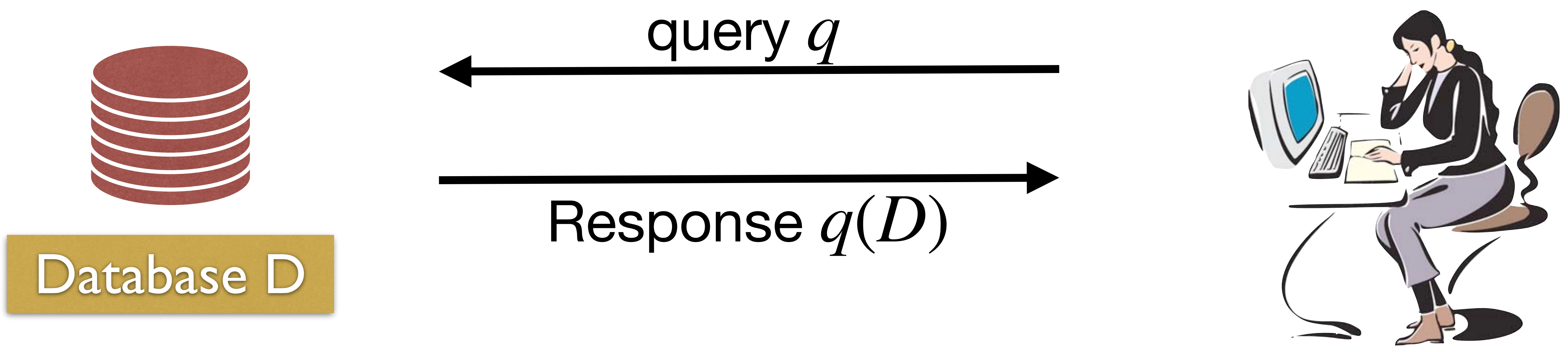
On Linear Time Decidability of Differential Privacy for Programs with Unbounded Inputs

Joint work with Rohit Chadha and Prasad Sistla

Mahesh Viswanathan, IARCS Verification Seminar Series, October 2022

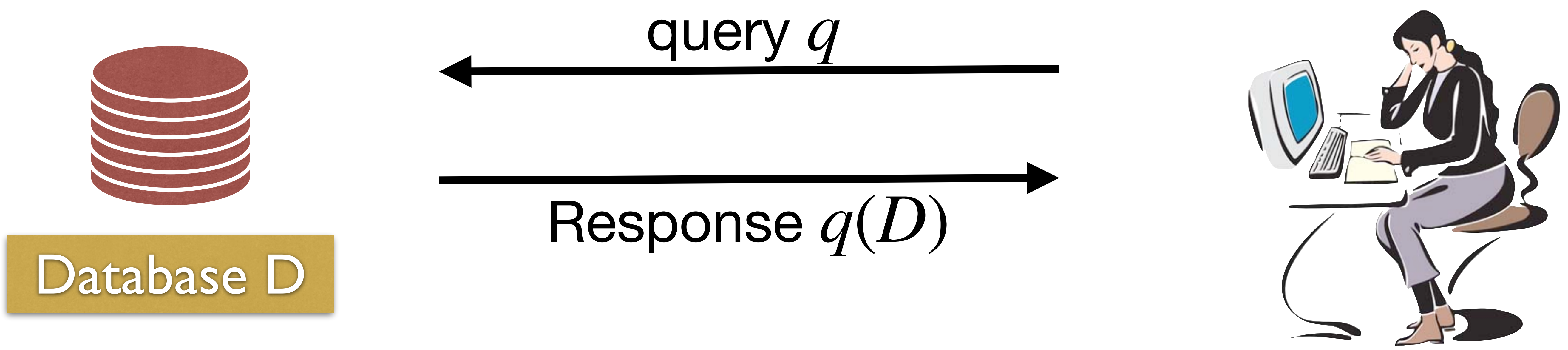
Accessing Sensitive Information

Accessing Sensitive Information



How can a database of personal information about individuals be accessed securely?

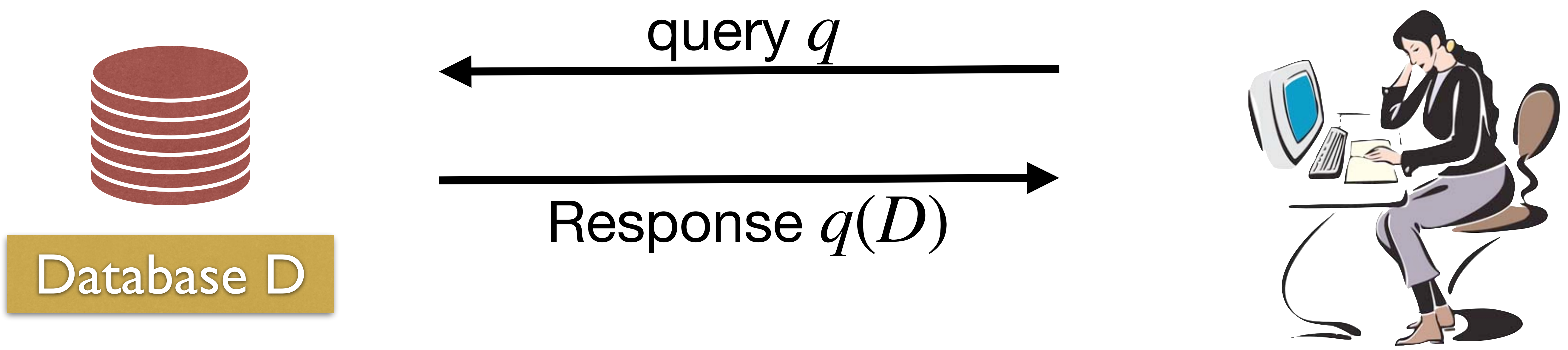
Accessing Sensitive Information



q(D) may reveal private information

How can a database of personal information about individuals be accessed securely?

Accessing Sensitive Information

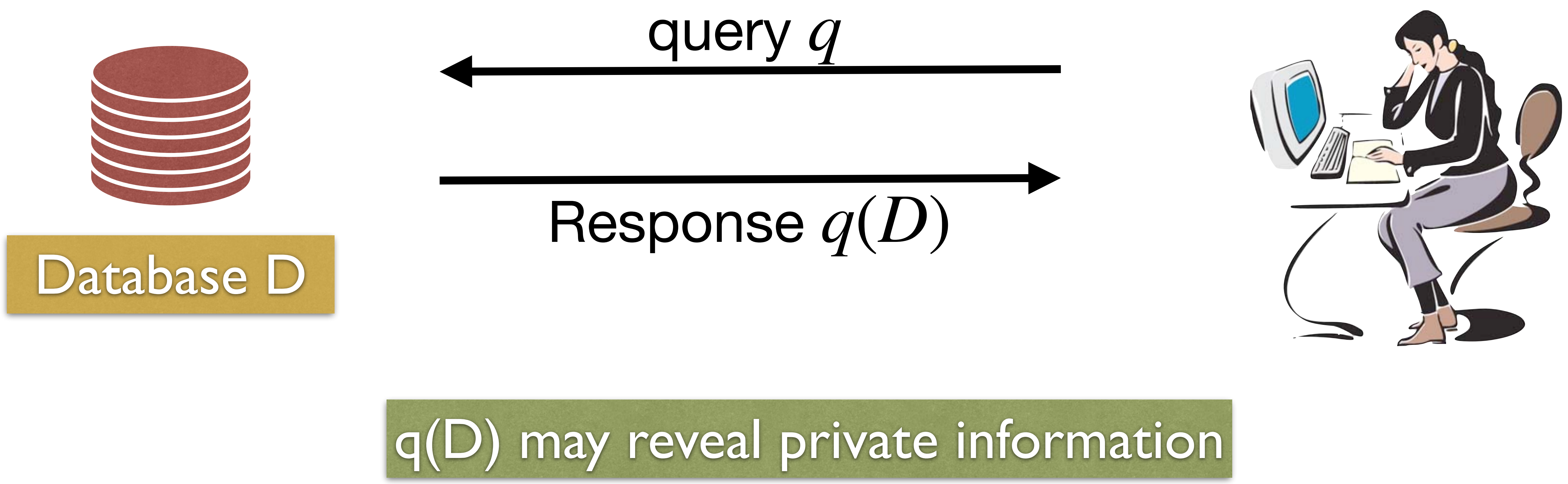


q(D) may reveal private information

How can a database of personal information about individuals be accessed securely?

- Security can be ensured if the data is not allowed to be accessed!

Accessing Sensitive Information

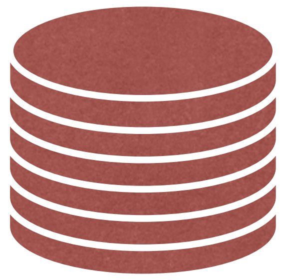


How can a database of personal information about individuals be accessed securely?

- Security can be ensured if the data is not allowed to be accessed!
- [Dinur-Nissim 2003] Entire database can be reconstructed using answers to a few random aggregate queries

Differential Privacy Framework

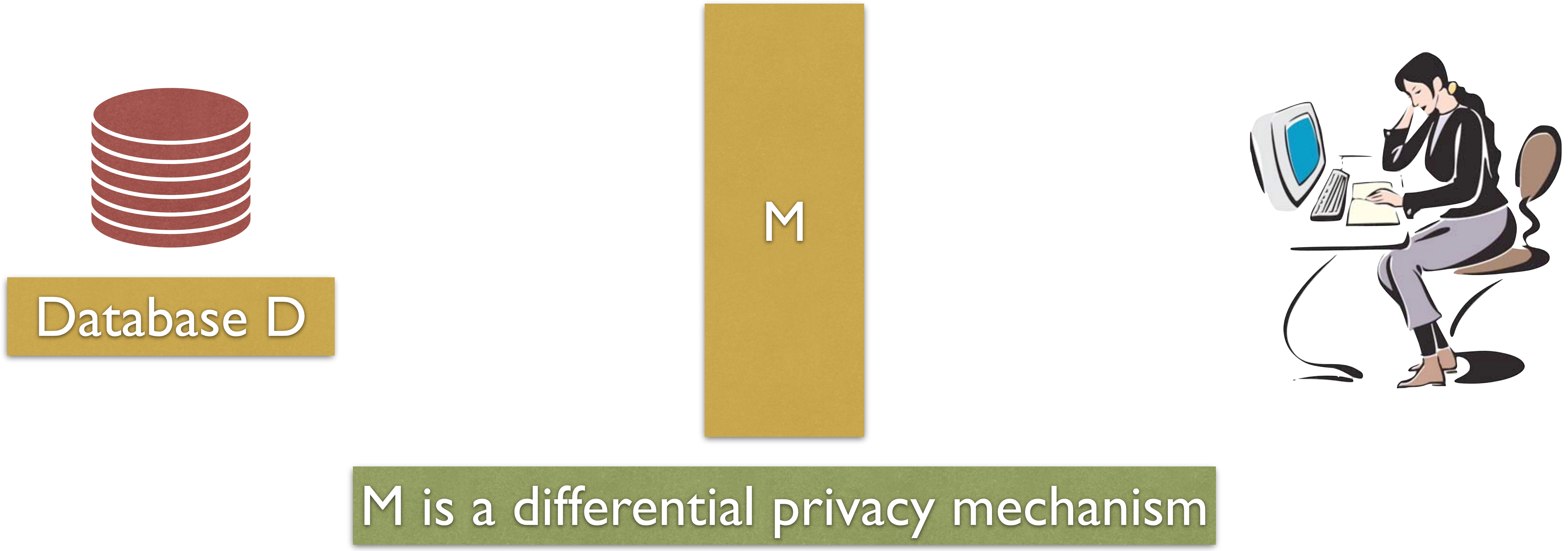
Dwork, McSherry, Nissim, Smith 2006



Database D

Differential Privacy Framework

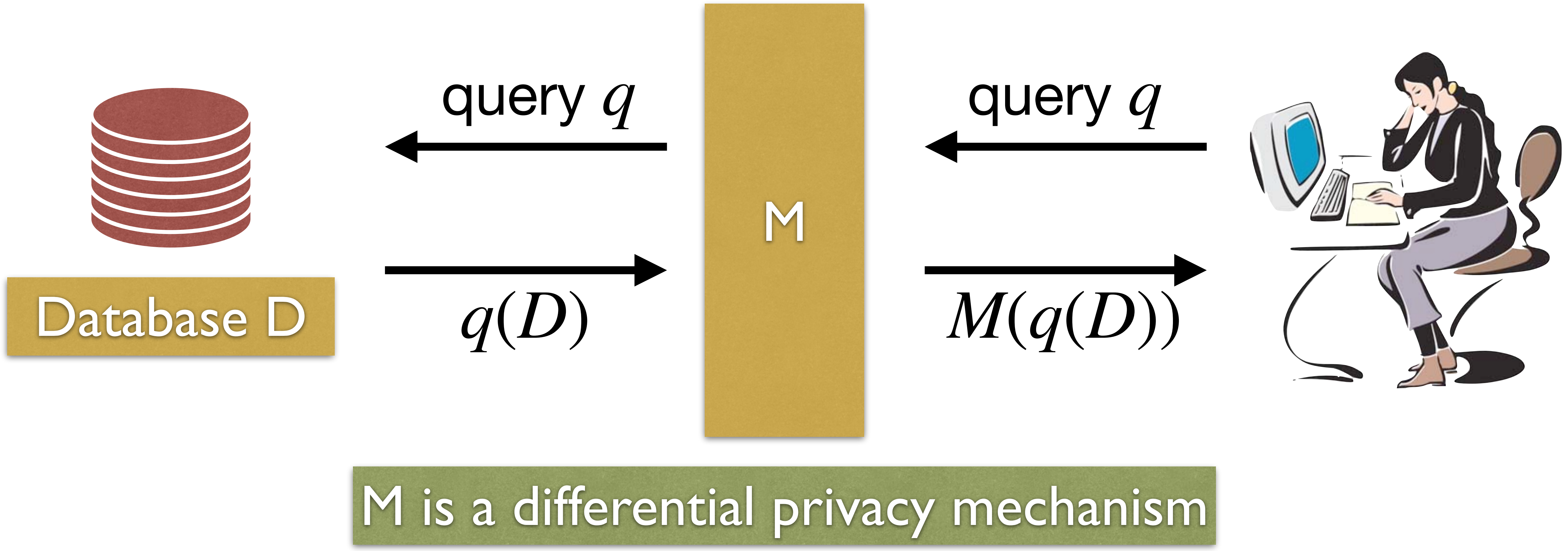
Dwork, McSherry, Nissim, Smith 2006



Interaction between database and user meditated by an algorithm M

Differential Privacy Framework

Dwork, McSherry, Nissim, Smith 2006

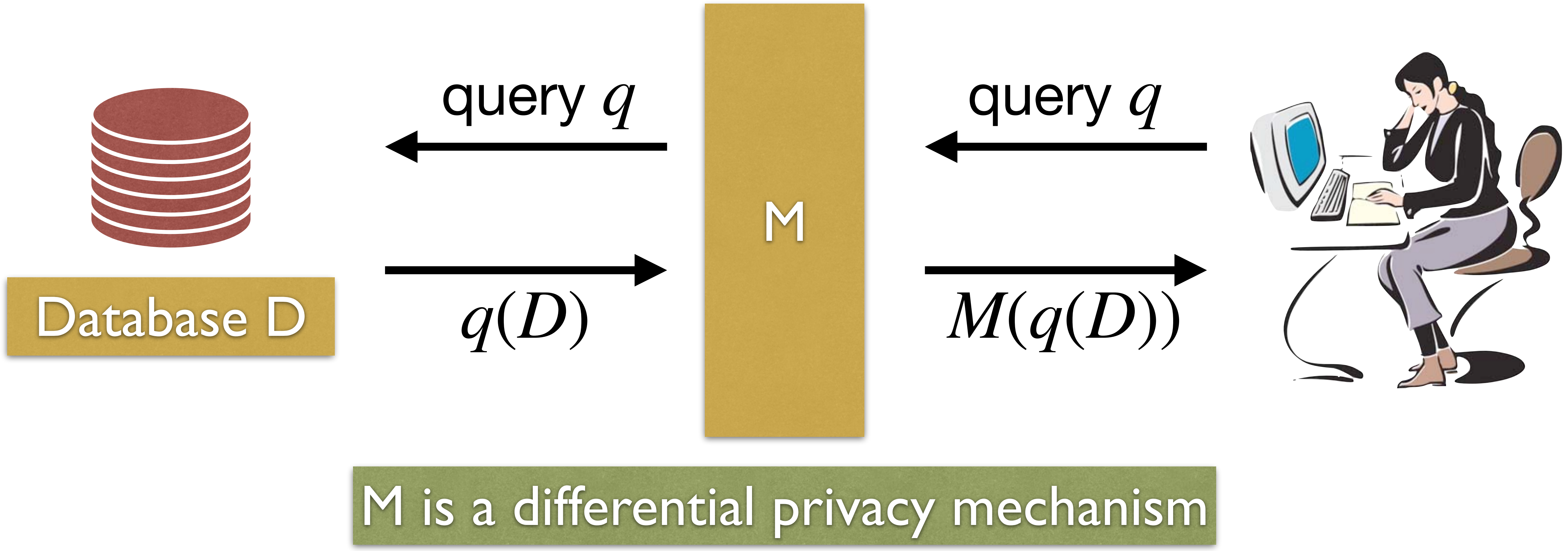


Interaction between database and user meditated by an algorithm M

- Add noise to $q(D)$ and share the noisy response $M(q(D))$

Differential Privacy Framework

Dwork, McSherry, Nissim, Smith 2006



Interaction between database and user meditated by an algorithm M

- Add noise to $q(D)$ and share the noisy response $M(q(D))$
- Trade "accuracy" for "privacy"

Randomized Response

An Example

Randomized Response

An Example

- **Goal:** Determine how many people in a population smoke

Randomized Response

An Example

- **Goal:** Determine how many people in a population smoke
- In response to a question “Do you smoke?”, each person is advised to answer as follows

Randomized Response

An Example

- **Goal:** Determine how many people in a population smoke
- In response to a question “Do you smoke?”, each person is advised to answer as follows
    - Toss a fair coin

Randomized Response

An Example

- **Goal:** Determine how many people in a population smoke
- In response to a question “Do you smoke?”, each person is advised to answer as follows
    - Toss a fair coin
    - If the result is “tails” answer truthfully

Randomized Response

An Example

- **Goal:** Determine how many people in a population smoke
- In response to a question “Do you smoke?”, each person is advised to answer as follows
  - Toss a fair coin
  - If the result is “tails” answer truthfully
  - If the result is “heads”, toss another coin. If the second coin toss is “heads” answer “Yes”, else answer “No”

Randomized Response

An Example

- **Goal:** Determine how many people in a population smoke
- In response to a question “Do you smoke?”, each person is advised to answer as follows
  - Toss a fair coin
  - If the result is “tails” answer truthfully
  - If the result is “heads”, toss another coin. If the second coin toss is “heads” answer “Yes”, else answer “No”
- “Privacy” arises from the plausible deniability of any outcome

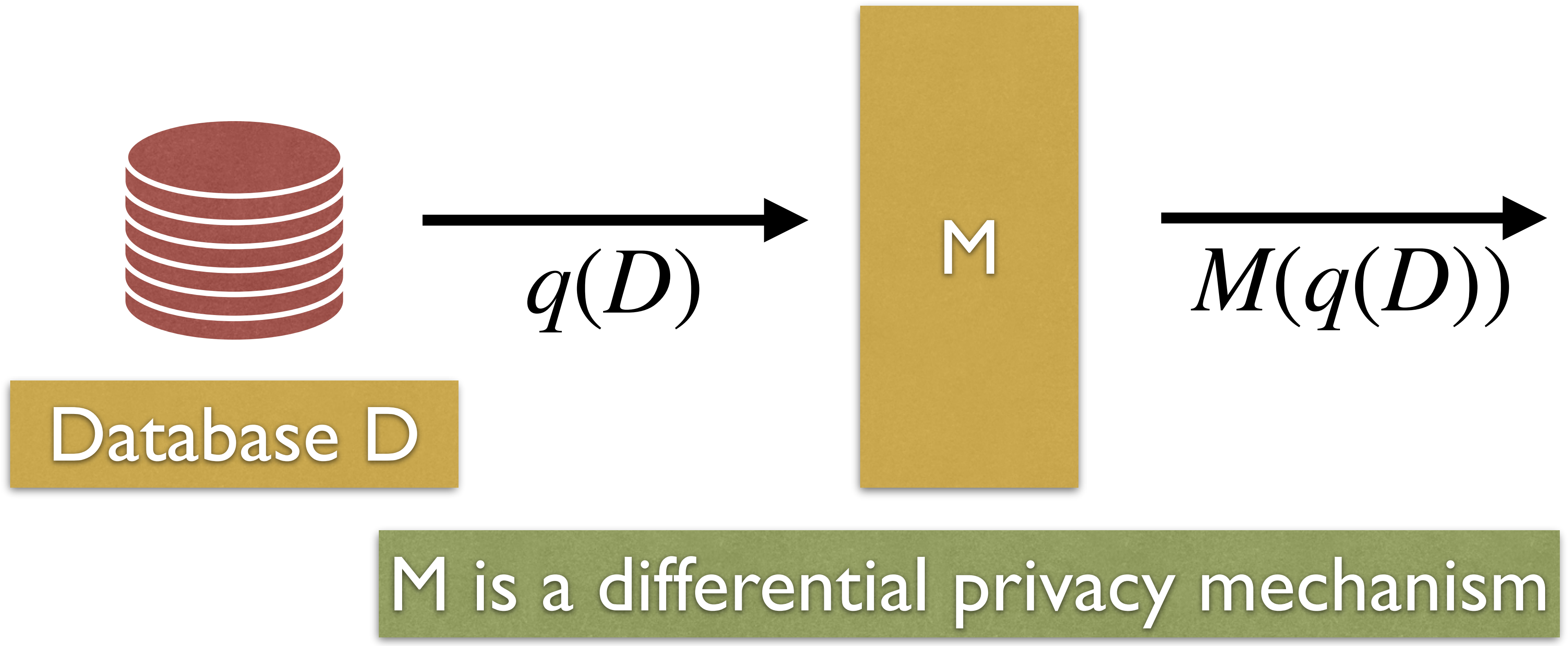
Randomized Response

An Example

- **Goal:** Determine how many people in a population smoke
- In response to a question “Do you smoke?”, each person is advised to answer as follows
  - Toss a fair coin
  - If the result is “tails” answer truthfully
  - If the result is “heads”, toss another coin. If the second coin toss is “heads” answer “Yes”, else answer “No”
- “Privacy” arises from the plausible deniability of any outcome
- If p is the fraction of smokers in a population, then the expected number of “Yes” responses is $(1/4)(1-p) + (3/4)p = (1/4) + (p/2)$

Differential Privacy

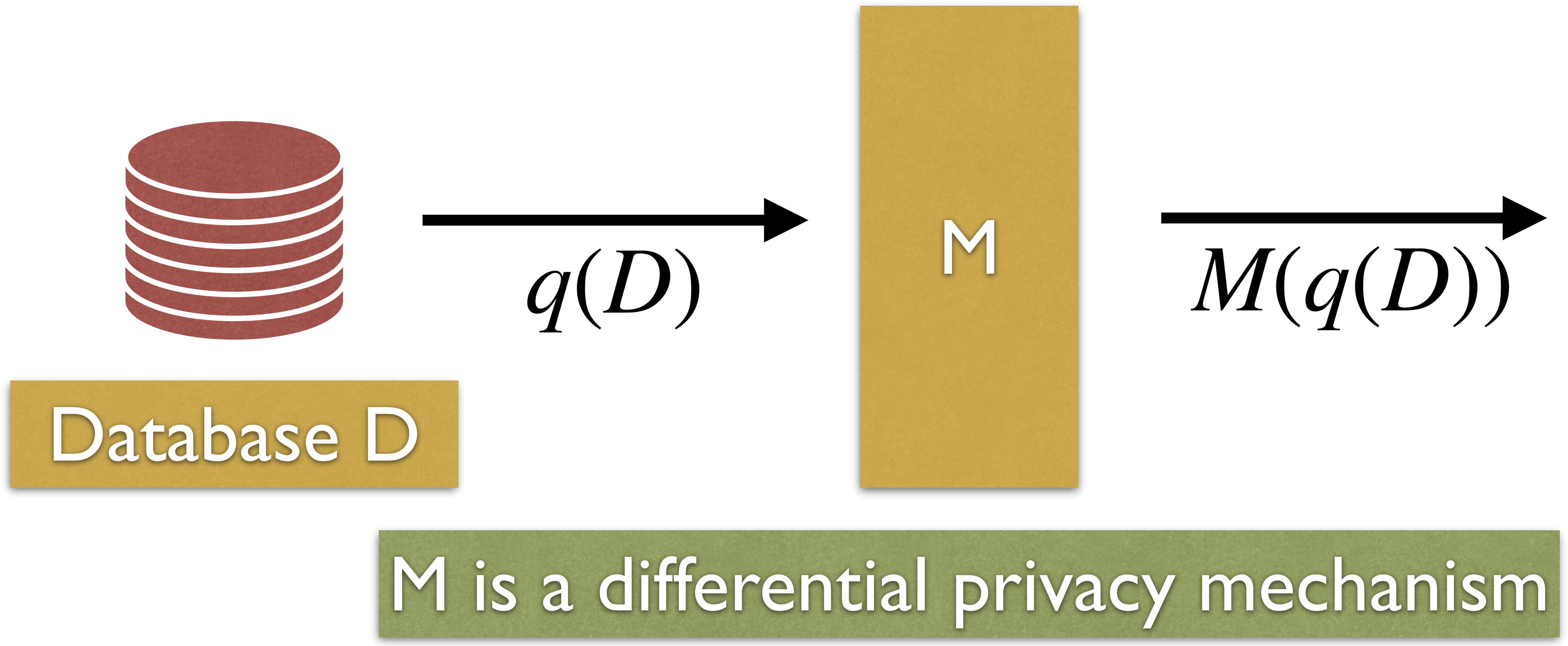
Definition [Dwork, McSherry, Nissim, Smith 2006]



Differential Privacy

Definition [Dwork, McSherry, Nissim, Smith 2006]

Privacy of x: "Behavior of M on database D and $D + x$ is similar."

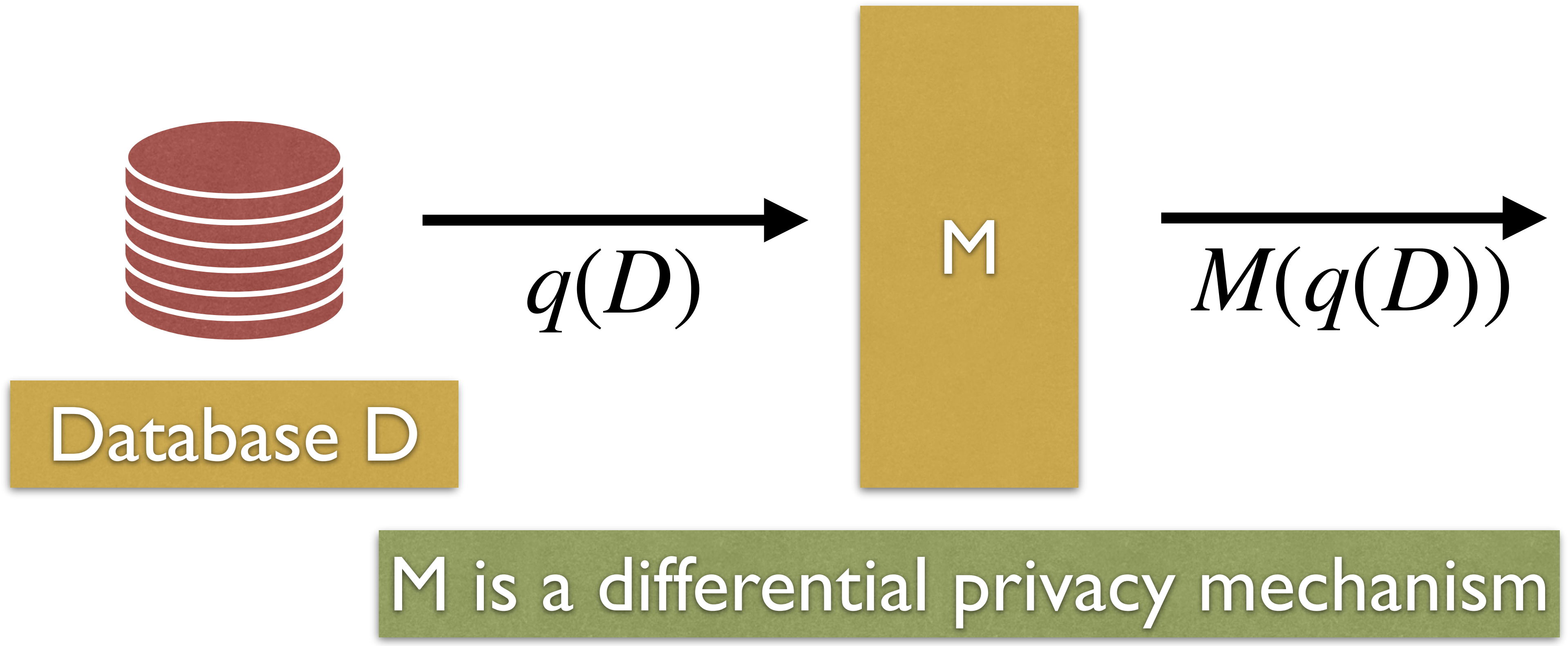


Differential Privacy

Definition [Dwork, McSherry, Nissim, Smith 2006]

Privacy of x: "Behavior of M on database D and $D + x$ is similar."

Program $M(\epsilon)$: Depends on privacy budget ϵ

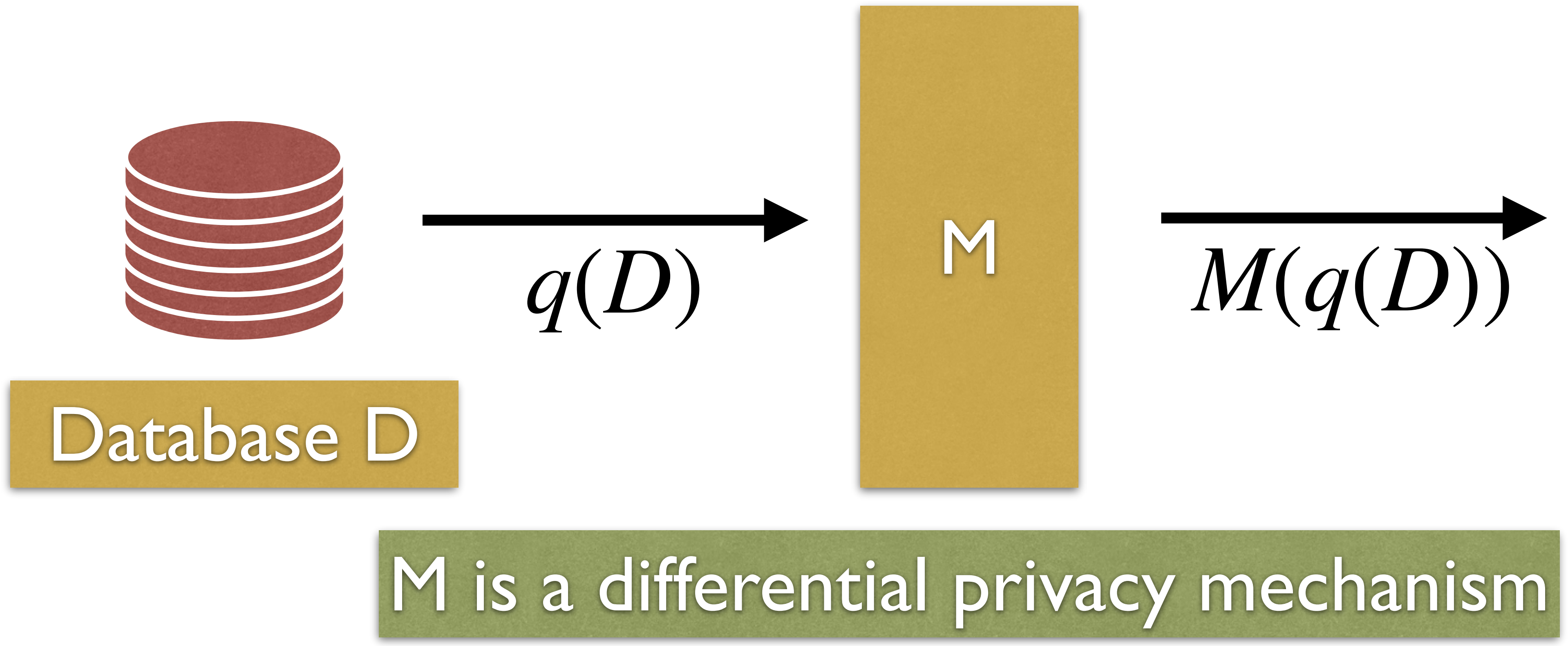


Differential Privacy

Definition [Dwork, McSherry, Nissim, Smith 2006]

Privacy of x: "Behavior of M on database D and $D + x$ is similar."

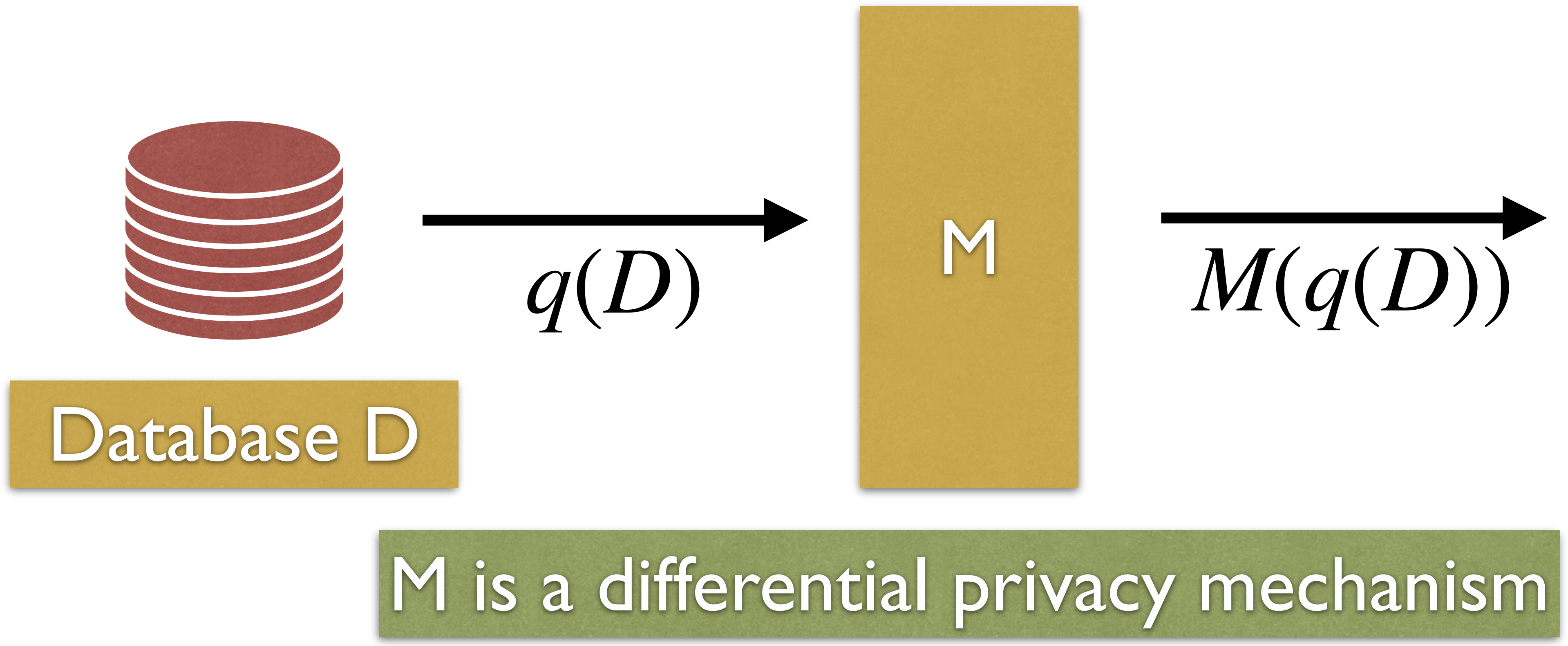
Program $M(\epsilon)$: Depends on privacy budget ϵ



Input to M: $q(D)$ sequence of numbers, answers to aggregate queries on D

Differential Privacy

Definition [Dwork, McSherry, Nissim, Smith 2006]



Privacy of x: "Behavior of M on database D and $D + x$ is similar."

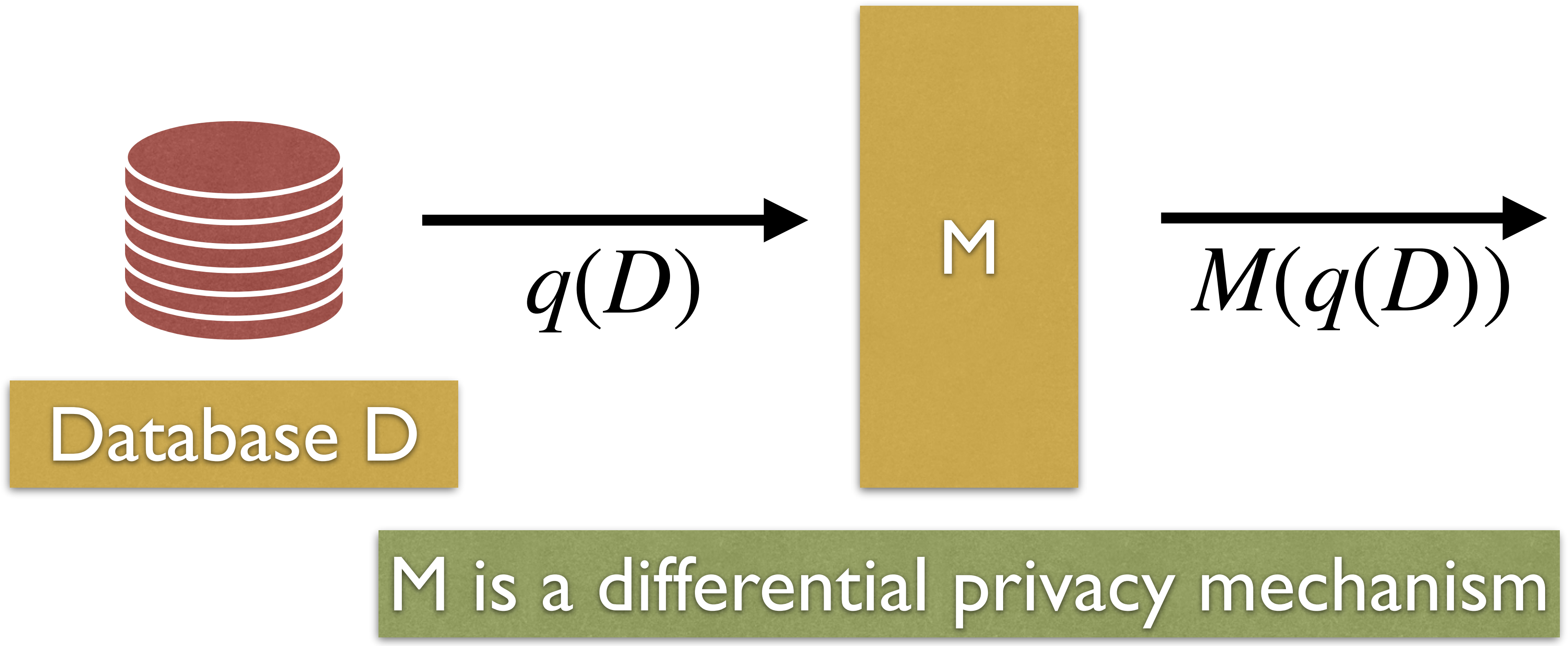
Program $M(\epsilon)$: Depends on privacy budget ϵ

Input to M: $q(D)$ sequence of numbers, answers to aggregate queries on D

Adjacency: Inputs q_1 and q_2 are adjacent if $|q_1[i] - q_2[i]| \leq 1$.

Differential Privacy

Definition [Dwork, McSherry, Nissim, Smith 2006]



Privacy of x: "Behavior of M on database D and $D + x$ is similar."

Program $M(\epsilon)$: Depends on privacy budget ϵ

Input to M: $q(D)$ sequence of numbers, answers to aggregate queries on D

Adjacency: Inputs q_1 and q_2 are adjacent if $|q_1[i] - q_2[i]| \leq 1$.

Definition: M is $d\epsilon$-differentially private if for any pair of adjacent inputs q_1, q_2 and each subset S of outputs

$$\Pr(M(q_1) \in S) \leq e^{d\epsilon} \Pr(M(q_2) \in S)$$

Sparse Vector Technique (SVT)

An Example

Sparse Vector Technique (SVT)

An Example

```
Input: query answers Q[1..n] and threshold T
```

Sparse Vector Technique (SVT)

An Example

```
Input: query answers Q[1..n] and threshold T
Output: first i such that Q[i] > T
```

Sparse Vector Technique (SVT)

An Example

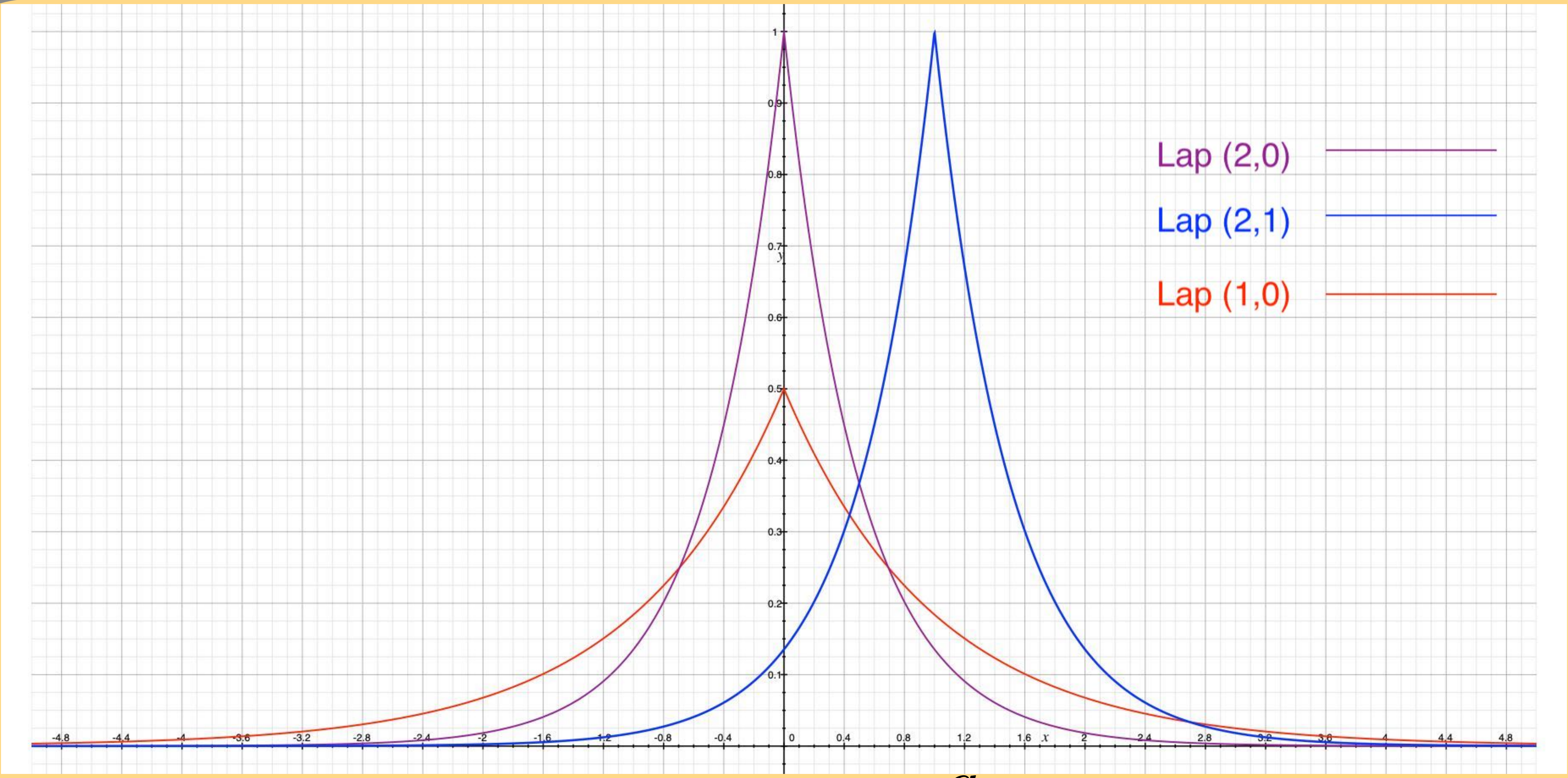
```
Input: query answers Q[1..n] and threshold T
Output: first i such that Q[i] > T
noisyT = Lap(ε/2,T)
```

Sparse Vector Technique (SVT)

An Example

```
Input: query answers Q[1..n] and threshold T
Output: first i such that Q[i] > T
```

noisyT = Lap($\frac{\epsilon}{2}$,T)



pdf of Lap(a, μ) is $\frac{a}{2}e^{-a|x-\mu|}$

Sparse Vector Technique (SVT)

An Example

Input: query answers Q[1..n] and threshold T

Output: first i such that Q[i] > T

noisyT = Lap($\frac{\epsilon}{2}$,T)

for i = 1 to n

  noisyQ = Lap($\frac{\epsilon}{4}$,Q[i])

  if noisyQ $\geq$ noisyT

    output $\top$; exit

  else

    output $\bot$

Sparse Vector Technique (SVT)

An Example

Input: query answers Q[1..n] and threshold T

Output: first i such that Q[i] > T

noisyT = Lap($\frac{\epsilon}{2}$,T)

for i = 1 to n

  noisyQ = Lap($\frac{\epsilon}{4}$,Q[i])

  if noisyQ $\geq$ noisyT

    output $\top$; exit

  else

    output $\perp$

Sparse Vector Technique (SVT)

An Example

Input: query answers Q[1..n] and threshold T

Output: first i such that Q[i] > T

```
noisyT = Lap(ε/2,T)
for i = 1 to n
  noisyQ = Lap(ε/4,Q[i])
  if noisyQ ≥ noisyT
    output ⊤; exit
  else
    output ⊥
```

Sparse Vector Technique (SVT)

An Example

```
Input: query answers Q[1..n] and threshold T
Output: first i such that Q[i] > T
noisyT = Lap(ϵ/2,T)
for i = 1 to n
  noisyQ = Lap(ϵ/4,Q[i])
  if noisyQ ≥ noisyT
    output ⊤; exit
  else
    output ⊥
```

$\forall \epsilon$, for all adjacent inputs q_1, q_2, for all $o \in \bot^* \top$

$\Pr[\mathrm{SVT}(q_1) = o] \leq e^{\epsilon} \Pr[\mathrm{SVT}(q_2) = o]$

Ensuring Privacy is subtle

An Example

```
Input: queries Q[1..n] and threshold T
Output: first i such that Q[i] > T
noisyT = Lap(ε/2,T)
for i = 1 to n
  noisyQ = Lap(ε/4,Q[i])
  if noisyQ ≥ noisyT
    output ⊤; exit
  else
    output ⊥
```

Ensuring Privacy is subtle

An Example

```
Input: queries Q[1..n] and threshold T
Output: first i such that Q[i] > T
noisyT = Lap(ε/2,T)
for i = 1 to n
  noisyQ = Lap(ε/4,Q[i])
  if noisyQ ≥ noisyT
    output ⊤; exit
  else
    output ⊥
```

ϵ-differentially private for all ϵ.

Ensuring Privacy is subtle

An Example

```
Input: queries Q[1..n] and threshold T
Output: first i such that Q[i] > T
noisyT = Lap(ϵ/2,T)
for i = 1 to n
  noisyQ = Lap(ϵ/4,Q[i])
  if noisyQ ≥ noisyT
    output ⊤; exit
  else
    output ⊥
```

ϵ-differentially private for all ϵ.

```
Input: queries Q[1..n] and threshold T
Output: first i such that Q[i] > T
noisyT = Lap(ϵ/2,T)
output noisyT
for i = 1 to n
  noisyQ = Lap(ϵ/4,Q[i])
  if noisyQ ≥ noisyT
    output ⊤; exit
  else
    output ⊥
```

Ensuring Privacy is subtle

An Example

```
Input: queries Q[1..n] and threshold T
Output: first i such that Q[i] > T
noisyT = Lap(ε/2,T)
for i = 1 to n
  noisyQ = Lap(ε/4,Q[i])
  if noisyQ ≥ noisyT
    output ⊤; exit
  else
    output ⊥
```

ϵ-differentially private for all ϵ.

```
Input: queries Q[1..n] and threshold T
Output: first i such that Q[i] > T
noisyT = Lap(ε/2,T)
output noisyT
for i = 1 to n
  noisyQ = Lap(ε/4,Q[i])
  if noisyQ ≥ noisyT
    output ⊤; exit
  else
    output ⊥
```

Ensuring Privacy is subtle

An Example

```
Input: queries Q[1..n] and threshold T
Output: first i such that Q[i] > T
noisyT = Lap(ε/2,T)
for i = 1 to n
  noisyQ = Lap(ε/4,Q[i])
  if noisyQ ≥ noisyT
    output ⊤; exit
  else
    output ⊥
```

ϵ-differentially private for all ϵ.

```
Input: queries Q[1..n] and threshold T
Output: first i such that Q[i] > T
noisyT = Lap(ε/2,T)
output noisyT
for i = 1 to n
  noisyQ = Lap(ε/4,Q[i])
  if noisyQ ≥ noisyT
    output ⊤; exit
  else
    output ⊥
```

For all d, not $d\epsilon$-differentially private.

[BCJSV20] Given a program $M(\epsilon)$, the problem of determining if it is $d\epsilon$-differentially private is undecidable.

This Talk

This Talk

An automaton model to describe some differential privacy algorithms

- The model processes an (unbounded) stream of real-valued query answers (input) and produces a stream of outputs (finite or real-valued)
- Some well-known algorithms captured by the model

This Talk

An automaton model to describe some differential privacy algorithms

- The model processes an (unbounded) stream of real-valued query answers (input) and produces a stream of outputs (finite or real-valued)
- Some well-known algorithms captured by the model

Identify necessary and sufficient conditions when an automaton is differentially private

- Necessary and sufficient conditions can be checked in linear time
- Algorithm certifies differential privacy or produces counter-examples

Prior Work

- Constructing Privacy proofs: [Reed, Pierce 2010], [Gaboardi, Haeberlen, Hsu, Narayan, Pierce 2013], [Barthe, Kopf, Olmedo, Zanella-Beguelin 2013], [Barthe, Gaboardi, Gregoire, Hsu, Strub 2016], [Zhang, Kifer 2017], [Albarghouthi, Hsu 2018], [Wang, Ding, Wang, Kifer, Zhang 2019], [de Amorim, Gaboardi, Hsu, Katsumata 2019]

- Discovering privacy bugs: [Ding, Wang, Wang, Zhang, Kifer 2018], [Bichsel, Gehr, Drechsler-Cohen, Tsankov, Vechev 2018], [Wang, Ding, Kifer, Zhang 2020]

- Decision Procedures: [Barthe, Chadha, Jagannath, Sistla, V. 2019]

- **This talk:** Decision procedure for unbounded inputs

Differentially Private Automata (DiPA)

Overview

Differentially Private Automata (DiPA)

Overview

Parametric automata with finitely many control states and 2 variables: storage variable x and sampling variable insample. In each step:

Differentially Private Automata (DiPA)

Overview

Parametric automata with finitely many control states and 2 variables: storage variable x and sampling variable insample. In each step:

- A number is sampled from the Laplace distribution whose parameters depend on the current state and is stored in insample.

Differentially Private Automata (DiPA)

Overview

Parametric automata with finitely many control states and 2 variables: storage variable x and sampling variable insample. In each step:

- A number is sampled from the Laplace distribution whose parameters depend on the current state and is stored in insample.
- Depending on the current state, a real number is read from input and added to insample.

Differentially Private Automata (DiPA)

Overview

Parametric automata with finitely many control states and 2 variables: storage variable x and sampling variable insample. In each step:

- A number is sampled from the Laplace distribution whose parameters depend on the current state and is stored in insample.
- Depending on the current state, a real number is read from input and added to insample.
- If an input is read, then control state is changed based on a comparison between insample and stored value x. The transition has an output.

Differentially Private Automata (DiPA)

Overview

Parametric automata with finitely many control states and 2 variables: storage variable x and sampling variable insample. In each step:

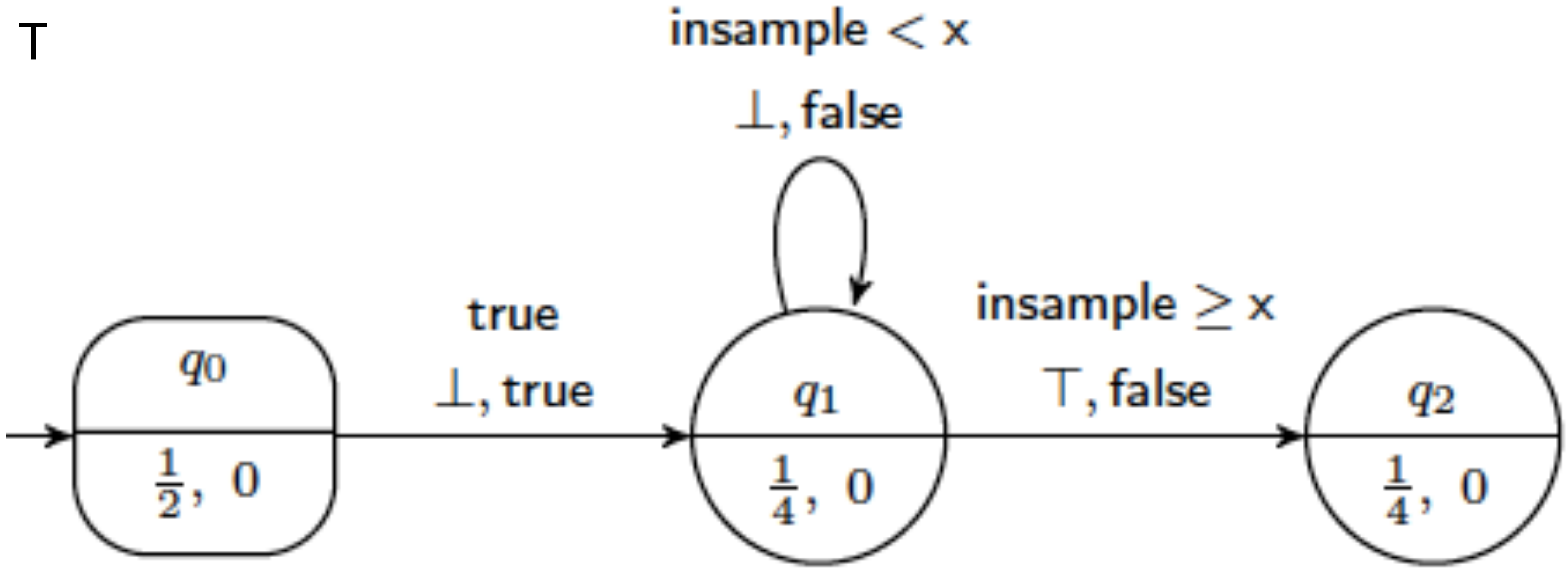
- A number is sampled from the Laplace distribution whose parameters depend on the current state and is stored in insample.
- Depending on the current state, a real number is read from input and added to insample.
- If an input is read, then control state is changed based on a comparison between insample and stored value x. The transition has an output.
- Based on the transition, the stored value x maybe be updated to insample.

DiPA: An Example

```
Input: Q[1..n]
Output: first i s.t. Q[i] > T
noisyT = Lap(ε/2,T)
for i = 1 to n
  noisyQ = Lap(ε/4,Q[i])
  if noisyQ ≥ noisyT
    output ⊤; exit
  else
    output ⊥
```

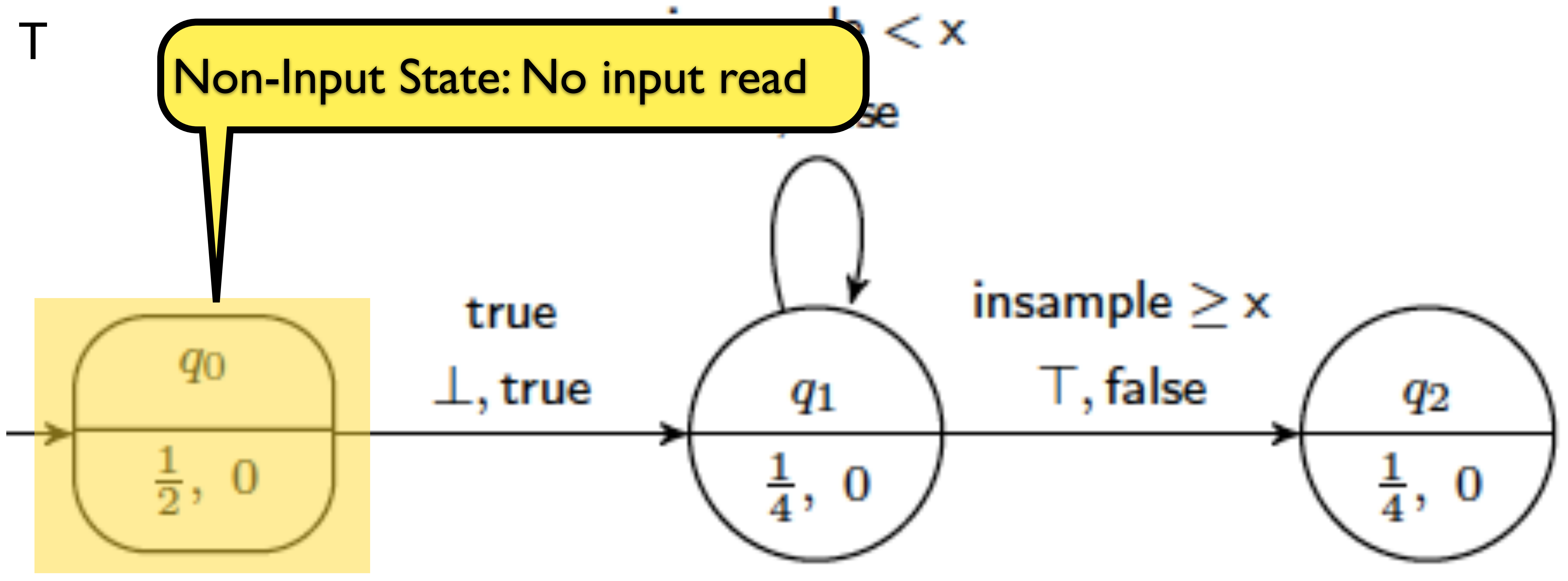
DiPA: An Example

```
Input: Q[1..n]
Output: first i s.t. Q[i] > T
noisyT = Lap(ε/2,T)
for i = 1 to n
  noisyQ = Lap(ε/4,Q[i])
  if noisyQ ≥ noisyT
    output ⊤; exit
  else
    output ⊥
```



DiPA: An Example

```
Input: Q[1..n]
Output: first i s.t. Q[i] > T
noisyT = Lap(ε/2,T)
for i = 1 to n
  noisyQ = Lap(ε/4,Q[i])
  if noisyQ ≥ noisyT
    output ⊤; exit
  else
    output ⊥
```



DiPA: An Example

Input: Q[1..n]
Output: first i s.t. Q[i] > T

noisyT = Lap($\frac{\epsilon}{2}$,T)

for i = 1 to n

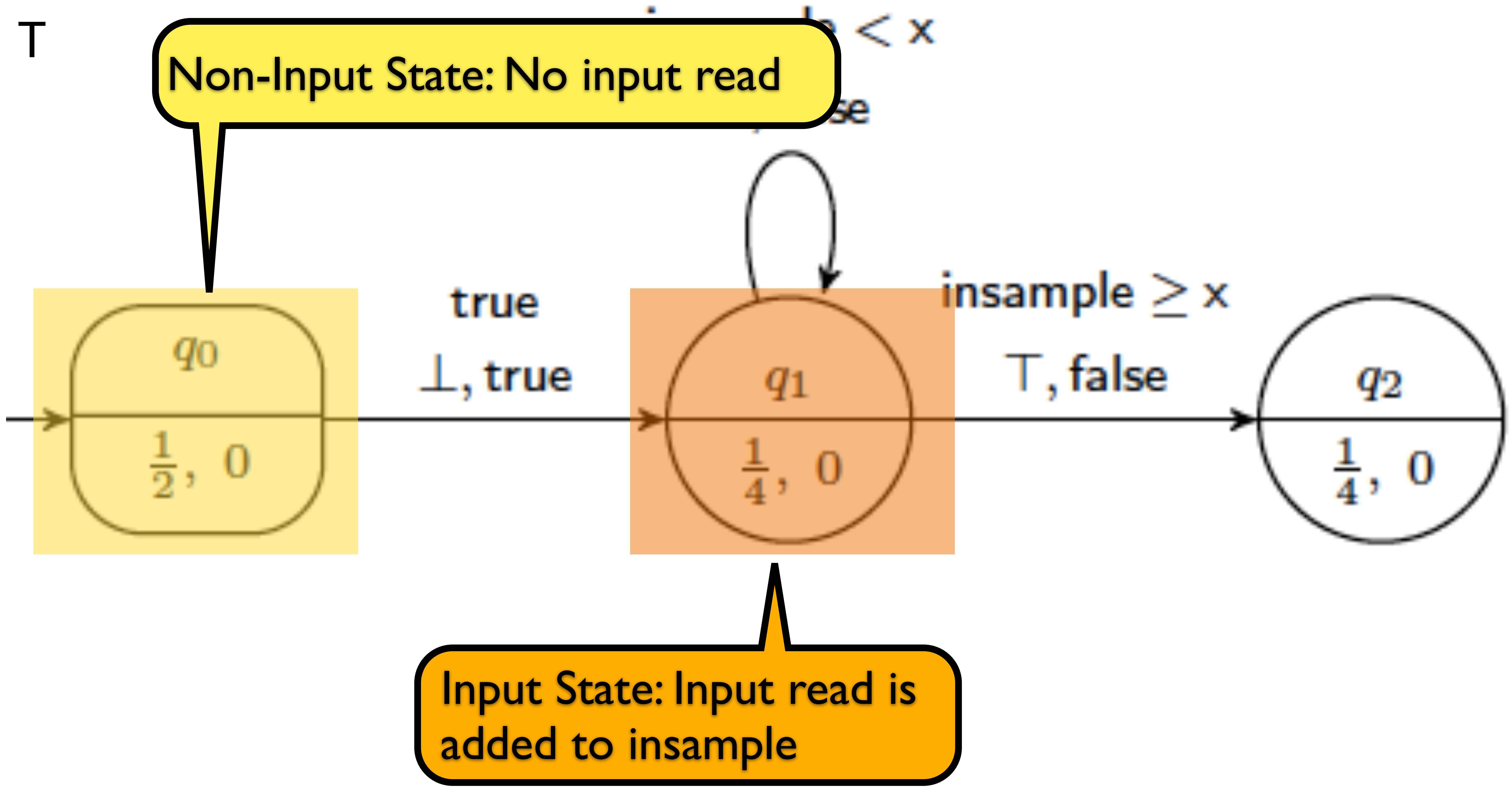
  noisyQ = Lap($\frac{\epsilon}{4}$,Q[i])

  if noisyQ $\geq$ noisyT

    output $\top$; exit

  else

    output $\perp$



DiPA: An Example

Input: Q[1..n]

Output: first i s.t. Q[i] > T

noisyT = Lap($\frac{\epsilon}{2}$,T)

for i = 1 to n

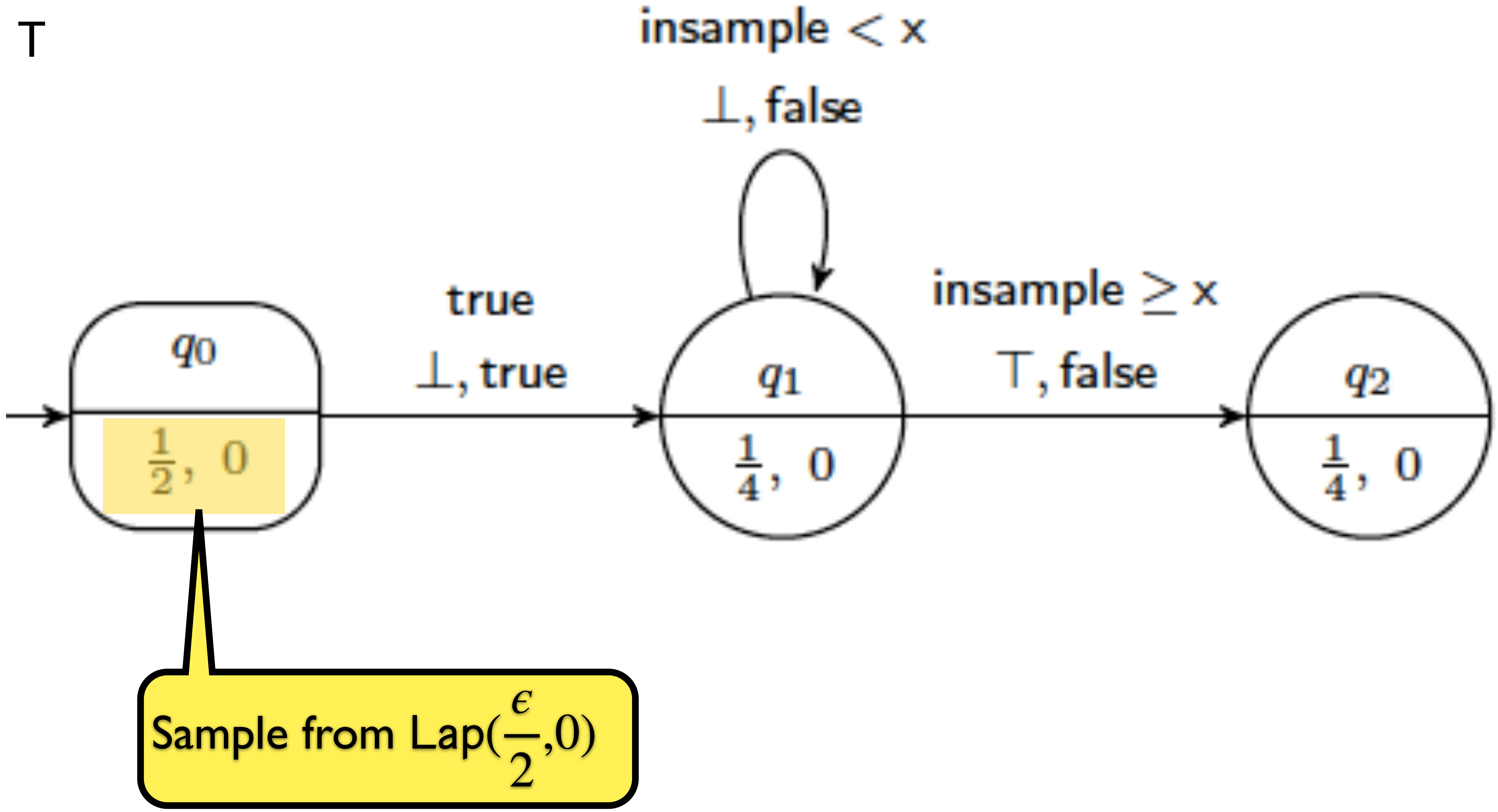
  noisyQ = Lap($\frac{\epsilon}{4}$,Q[i])

  if noisyQ ≥ noisyT

    output ⊤; exit

  else

    output ⊥

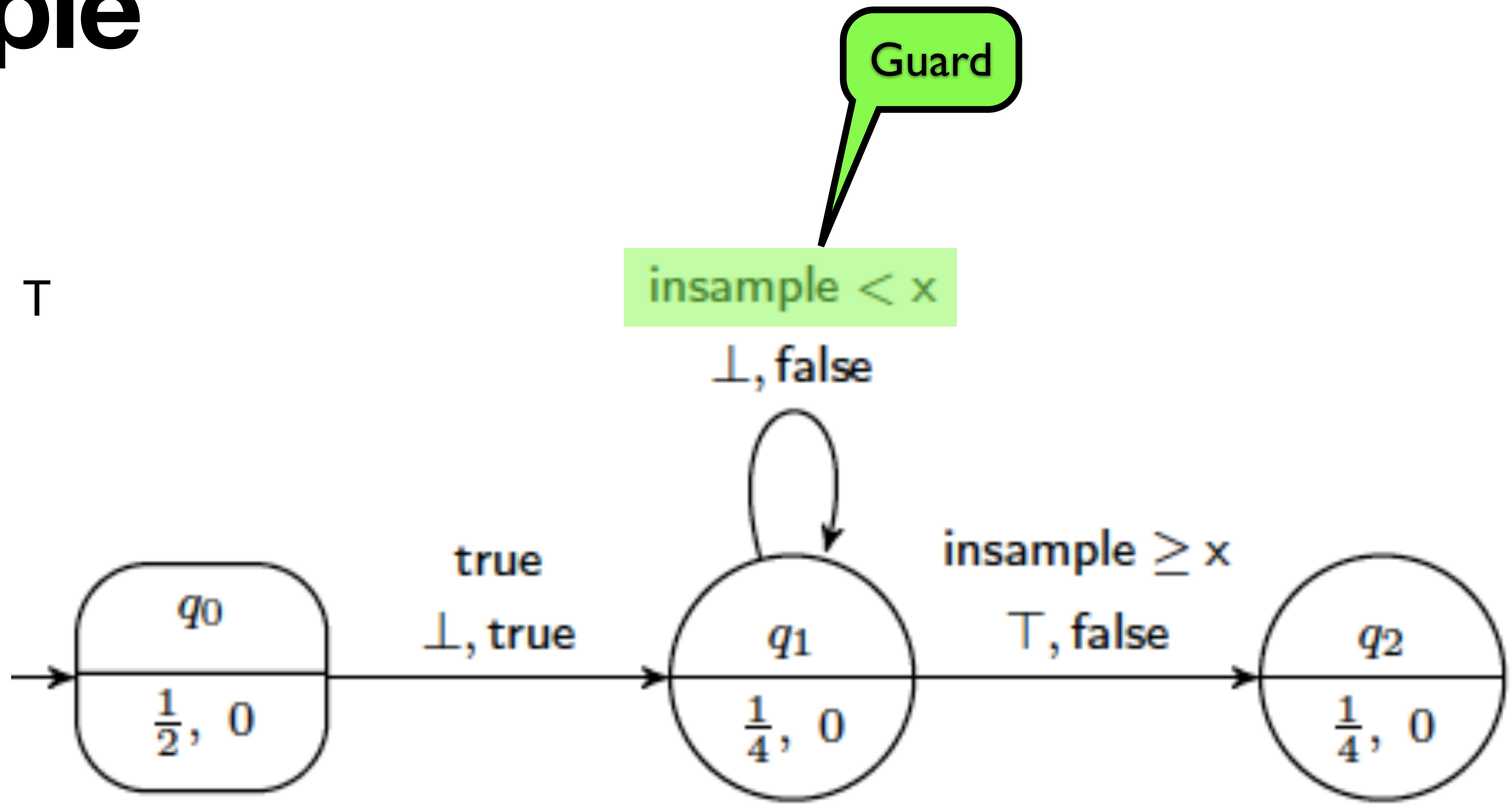


DiPA: An Example

Input: Q[1..n]

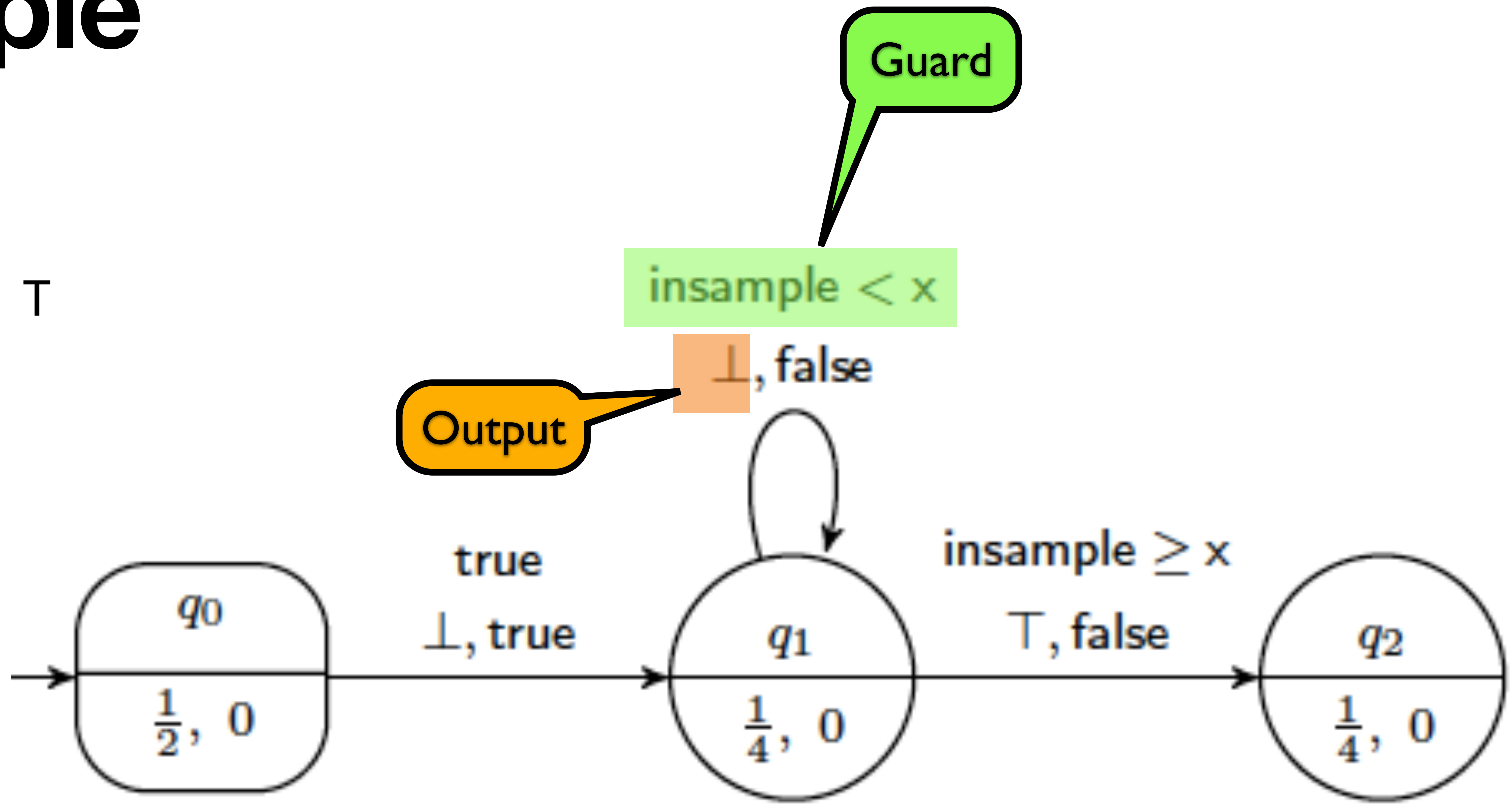
Output: first i s.t. Q[i] > T

```
noisyT = Lap(ε/2,T)
for i = 1 to n
  noisyQ = Lap(ε/4,Q[i])
  if noisyQ ≥ noisyT
    output T; exit
  else
    output ⊥
```



DiPA: An Example

```
Input: Q[1..n]
Output: first i s.t. Q[i] > T
noisyT = Lap(ε/2,T)
for i = 1 to n
  noisyQ = Lap(ε/4,Q[i])
  if noisyQ ≥ noisyT
    output ⊤; exit
  else
    output ⊥
```



DiPA: An Example

Input: Q[1..n]

Output: first i s.t. Q[i] > T

noisyT = Lap($\frac{\epsilon}{2}$,T)

for i = 1 to n

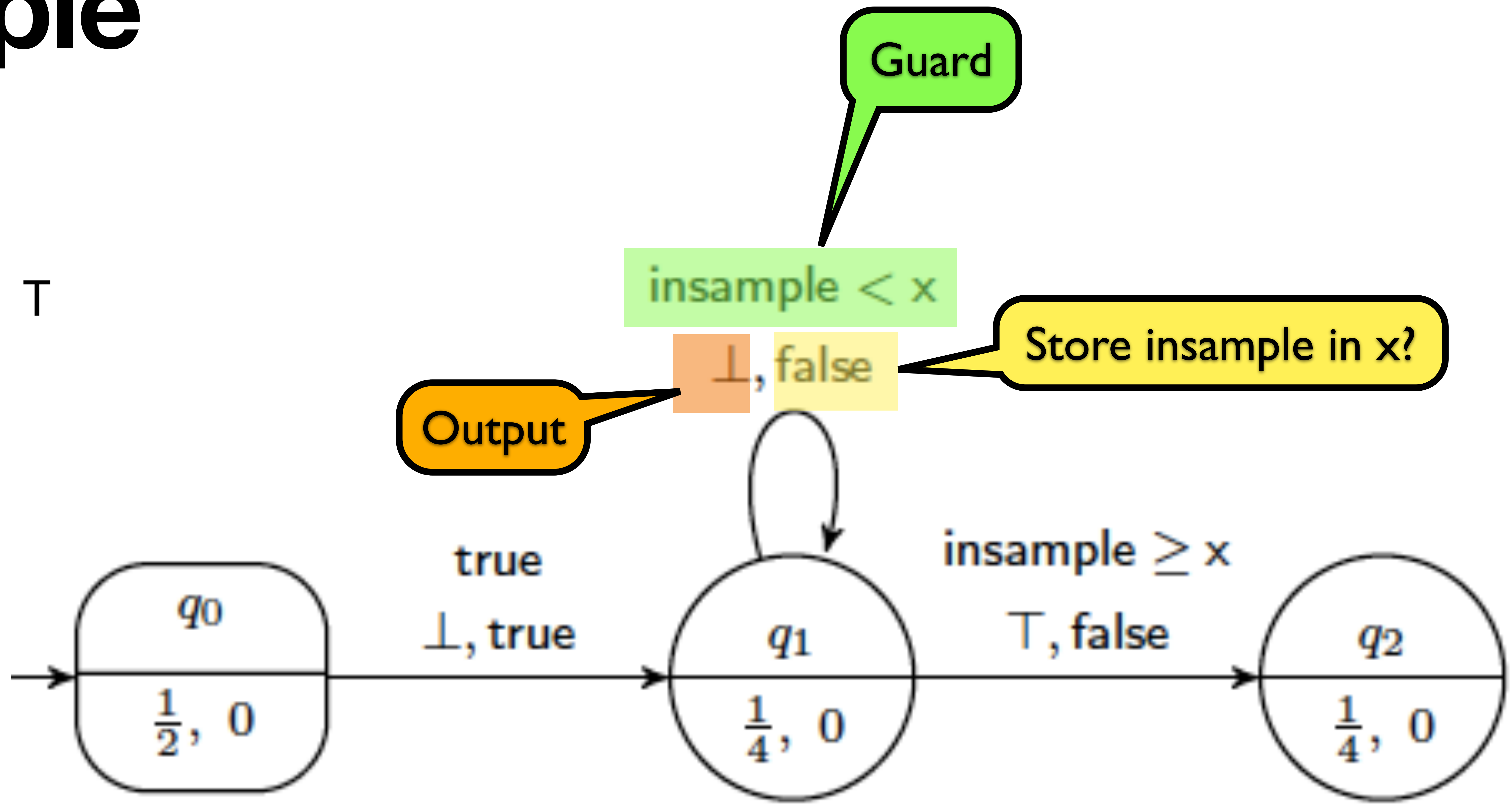
  noisyQ = Lap($\frac{\epsilon}{4}$,Q[i])

  if noisyQ $\geq$ noisyT

    output ⊤; exit

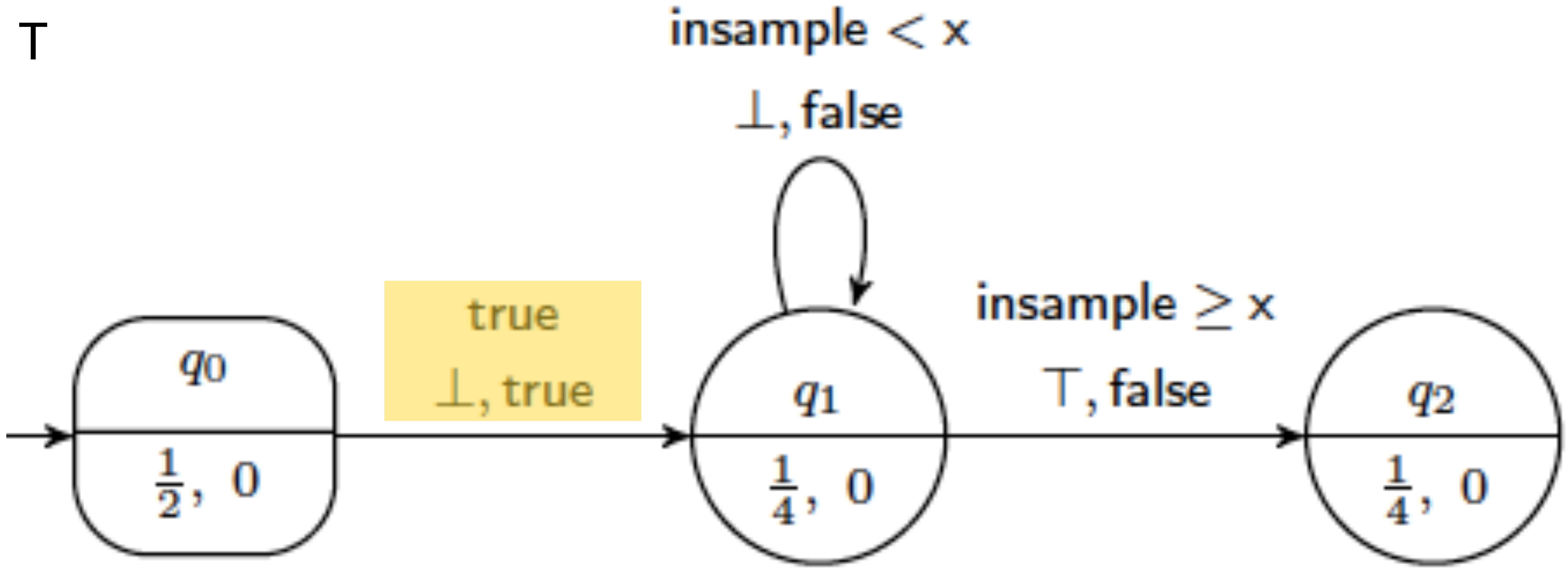
  else

    output ⊥



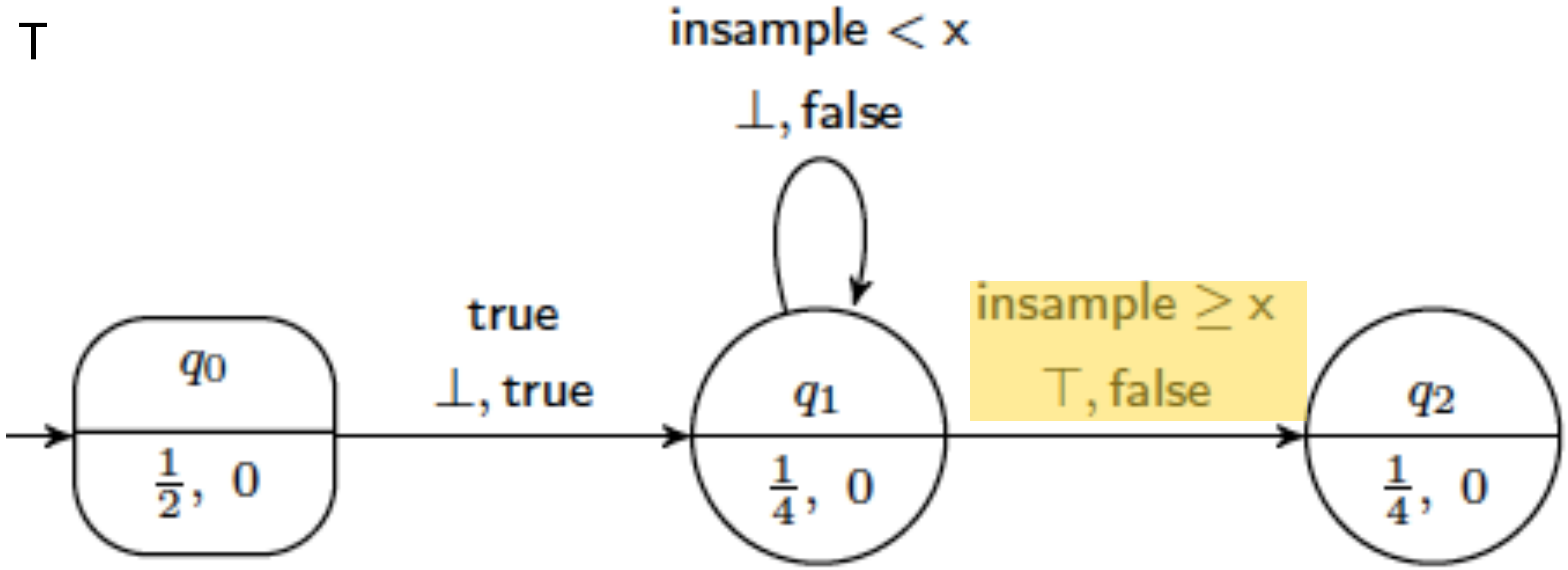
DiPA: An Example

```
Input: Q[1..n]
Output: first i s.t. Q[i] > T
noisyT = Lap(ε/2,T)
for i = 1 to n
  noisyQ = Lap(ε/4,Q[i])
  if noisyQ ≥ noisyT
    output ⊤; exit
  else
    output ⊥
```



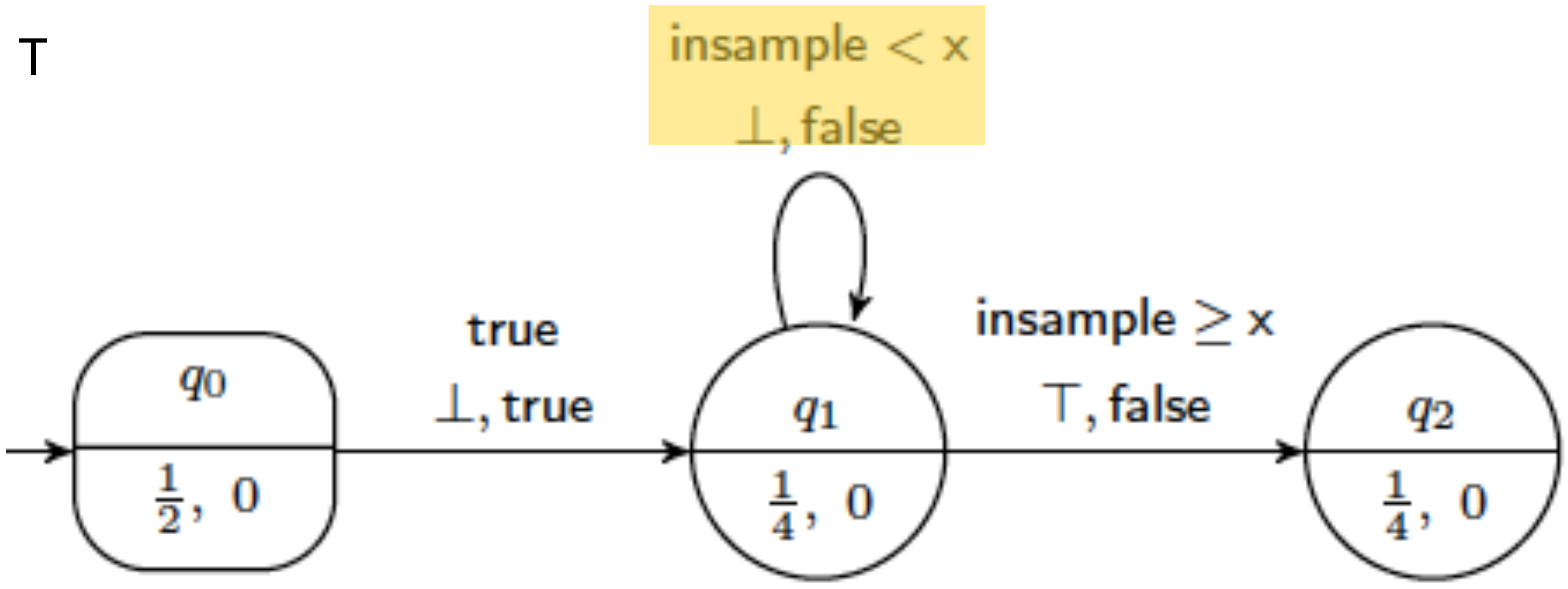
DiPA: An Example

```
Input: Q[1..n]
Output: first i s.t. Q[i] > T
noisyT = Lap(ε/2,T)
for i = 1 to n
  noisyQ = Lap(ε/4,Q[i])
  if noisyQ ≥ noisyT
    output ⊤; exit
  else
    output ⊥
```



DiPA: An Example

```
Input: Q[1..n]
Output: first i s.t. Q[i] > T
noisyT = Lap(ε/2,T)
for i = 1 to n
  noisyQ = Lap(ε/4,Q[i])
  if noisyQ ≥ noisyT
    output ⊤; exit
  else
    output ⊥
```



DiPA: Terms and Assumptions

DiPA: Terms and Assumptions

Assignment Transition: One where insample is assigned to x

DiPA: Terms and Assumptions

Assignment Transition: One where insample is assigned to x

Initialization: Initial state has only one transition which is an assignment transition and guard is true

DiPA: Terms and Assumptions

Assignment Transition: One where insample is assigned to x

Initialization: Initial state has only one transition which is an assignment transition and guard is true

Determinacy and Output Distinction: Transitions out of any state have disjoint guards with distinct outputs

DiPA: Terms and Assumptions

Assignment Transition: One where insample is assigned to x

Initialization: Initial state has only one transition which is an assignment transition and guard is true

Determinacy and Output Distinction: Transitions out of any state have disjoint guards with distinct outputs

- Knowing the start state and the sequence of outputs, determines the sequence of transitions taken

DiPA: Terms and Assumptions

Assignment Transition: One where insample is assigned to x

Initialization: Initial state has only one transition which is an assignment transition and guard is true

Determinacy and Output Distinction: Transitions out of any state have disjoint guards with distinct outputs

- Knowing the start state and the sequence of outputs, determines the sequence of transitions taken

Paths: Sequence of consecutive transitions augmented with inputs

DiPA: Terms and Assumptions

Assignment Transition: One where insample is assigned to x

Initialization: Initial state has only one transition which is an assignment transition and guard is true

Determinacy and Output Distinction: Transitions out of any state have disjoint guards with distinct outputs

- Knowing the start state and the sequence of outputs, determines the sequence of transitions taken

Paths: Sequence of consecutive transitions augmented with inputs

- For path ρ, $\mathsf{inseq}(\rho)$ is the sequence of inputs and $\mathsf{outseq}(\rho)$ is the sequence of outputs.

DiPA: Terms and Assumptions

Assignment Transition: One where insample is assigned to x

Initialization: Initial state has only one transition which is an assignment transition and guard is true

Determinacy and Output Distinction: Transitions out of any state have disjoint guards with distinct outputs

- Knowing the start state and the sequence of outputs, determines the sequence of transitions taken

Paths: Sequence of consecutive transitions augmented with inputs

- For path ρ, inseq(ρ) is the sequence of inputs and outseq(ρ) is the sequence of outputs.

Probability: For path ρ from initial state, Pr$[\epsilon_0, \rho]$ is the probability when security parameter is ϵ_0

Computational Problem and Results

Computational Problem and Results

$d\epsilon$-Differential Privacy: DiPA $\mathscr{A}$ is $d\epsilon$-differentially private (for $d > 0$) if for all $\epsilon > 0$, and any two paths ρ_1, ρ_2 starting from the initial state such that outseq$(\rho_1) =$ outseq(ρ_1) and adjacent inseq(ρ_1) and inseq(ρ_2), Pr$[\epsilon, \rho_1] \leq e^{d\epsilon}$ Pr$[\epsilon, \rho_2]$.

Computational Problem and Results

$d\epsilon$-Differential Privacy: DiPA $\mathscr{A}$ is $d\epsilon$-differentially private (for $d > 0$) if for all $\epsilon > 0$, and any two paths ρ_1, ρ_2 starting from the initial state such that $\mathsf{outseq}(\rho_1) = \mathsf{outseq}(\rho_1)$ and adjacent $\mathsf{inseq}(\rho_1)$ and $\mathsf{inseq}(\rho_2)$, $\Pr[\epsilon, \rho_1] \leq e^{d\epsilon} \Pr[\epsilon, \rho_2]$.

Differential Privacy: DiPA $\mathscr{A}$ is differentially private if $\exists d \,.\, \mathscr{A}$ is $d\epsilon$-differentially private.

Computational Problem and Results

$d\epsilon$-Differential Privacy: DiPA $\mathscr{A}$ is $d\epsilon$-differentially private (for $d > 0$) if for all $\epsilon > 0$, and any two paths ρ_1, ρ_2 starting from the initial state such that $\mathsf{outseq}(\rho_1) = \mathsf{outseq}(\rho_1)$ and adjacent $\mathsf{inseq}(\rho_1)$ and $\mathsf{inseq}(\rho_2)$, $\Pr[\epsilon, \rho_1] \leq e^{d\epsilon} \Pr[\epsilon, \rho_2]$.

Differential Privacy: DiPA $\mathscr{A}$ is differentially private if $\exists d\,.\,\mathscr{A}$ is $d\epsilon$-differentially private.

Theorem: Given $\mathscr{A}$, there is a linear-time algorithm that can determine if $\mathscr{A}$ is differentially private.

Rest of the talk: Assume output alphabet is a finite set

Leaking Cycle

Cycle: Path that starts and ends in the same control state

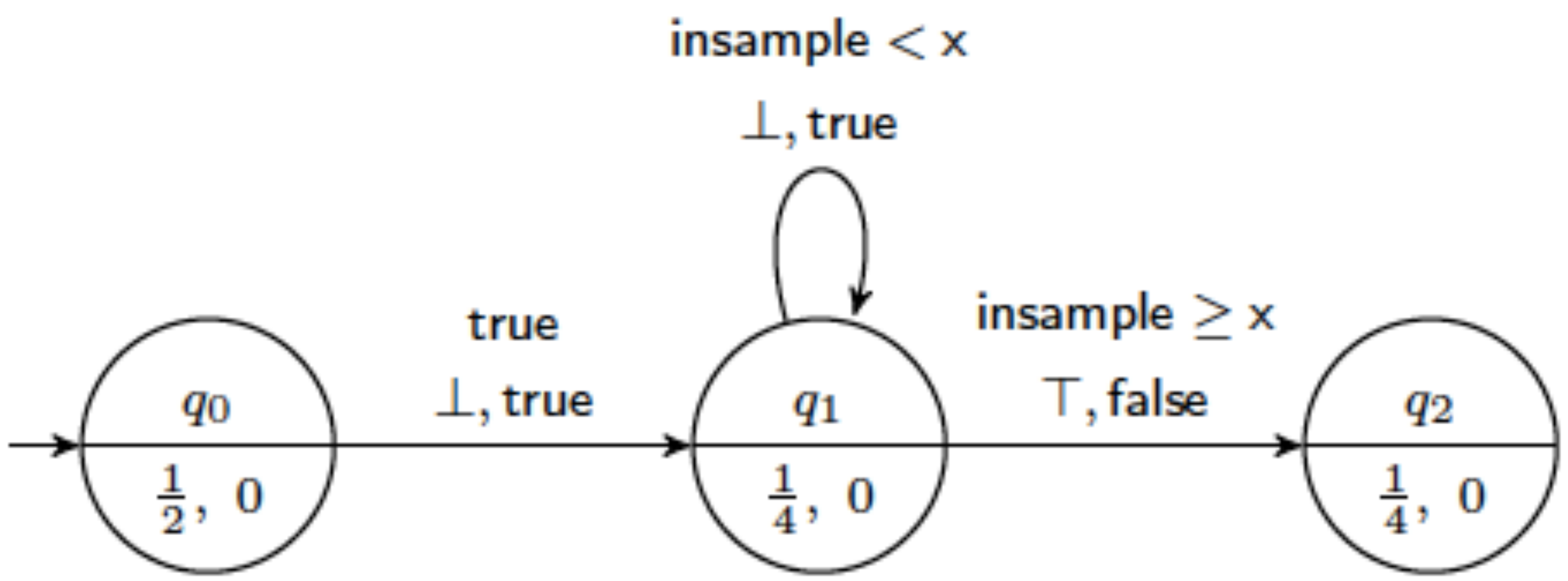
Leaking Cycle: A cycle that has an assignment transition and a transition with a guard that references value stored in x

Leaking Cycle

Cycle: Path that starts and ends in the same control state

Leaking Cycle: A cycle that has an assignment transition and a transition with a guard that references value stored in x

Example: Consider a program that outputs $\perp$ as long as queries are ordered in descending order, and outputs $\top$ and stops on encountering the first pair in wrong order.

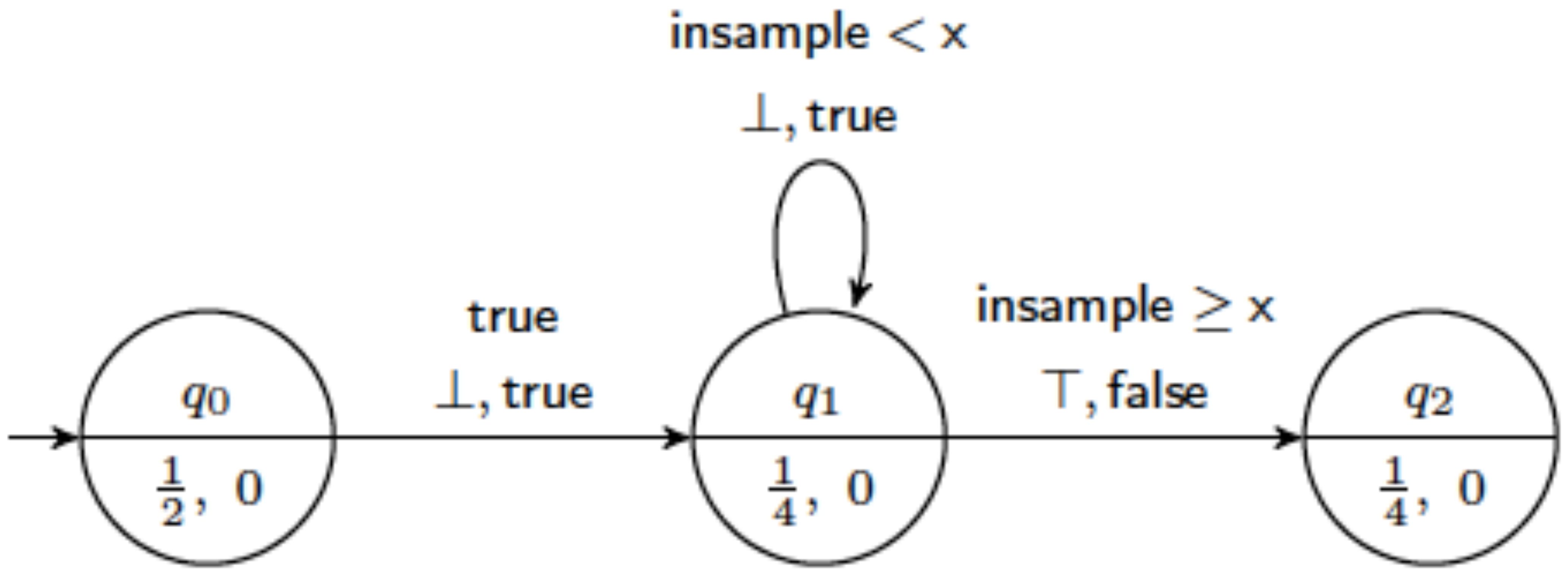


Leaking Cycle

Cycle: Path that starts and ends in the same control state

Leaking Cycle: A cycle that has an assignment transition and a transition with a guard that references value stored in x

Example: Consider a program that outputs $\perp$ as long as queries are ordered in descending order, and outputs $\top$ and stops on encountering the first pair in wrong order.



Self loop on q_1 is a leaking cycle.

Leaking Pair

Leaking Pair: A pair of cycles (C_1, C_2) connected by a path ρ such that

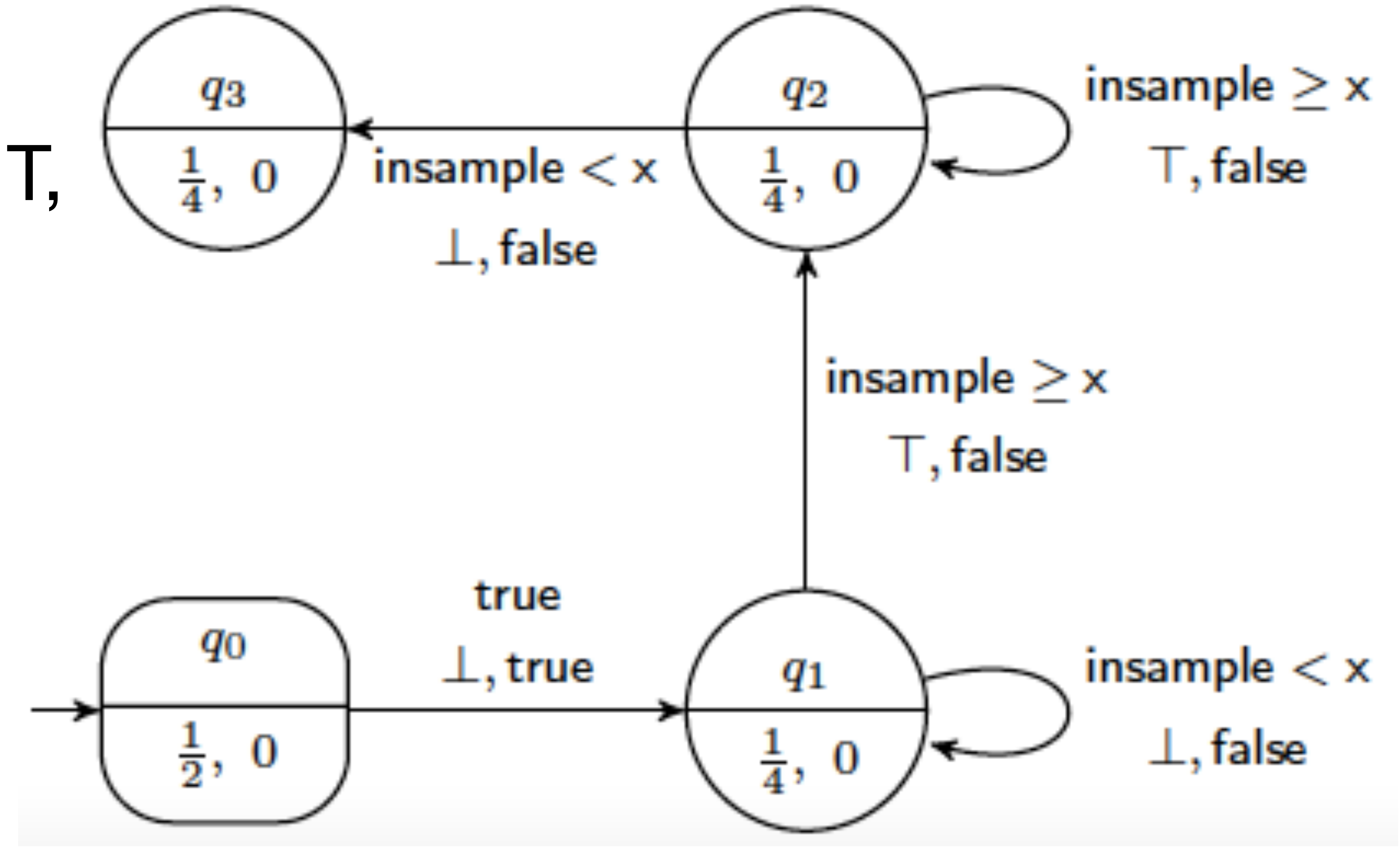
- C_1 and C_2 have no assignment transitions,
- C_1 has a transition with guard insample < x and C_2 has a transition with guard insample ≥ x (or vice versa)
- Assignment transitions in ρ ensure that value in x is at least (or at most) what it is in C_1

Leaking Pair

Leaking Pair: A pair of cycles (C_1, C_2) connected by a path ρ such that

- C_1 and C_2 have no assignment transitions,
- C_1 has a transition with guard insample < x and C_2 has a transition with guard insample $\geq$ x (or vice versa)
- Assignment transitions in ρ ensure that value in x is at least (or at most) what it is in C_1

Example: First checks if queries are less than threshold T, then if they are greater than T, and then stops at first query that is less T again.



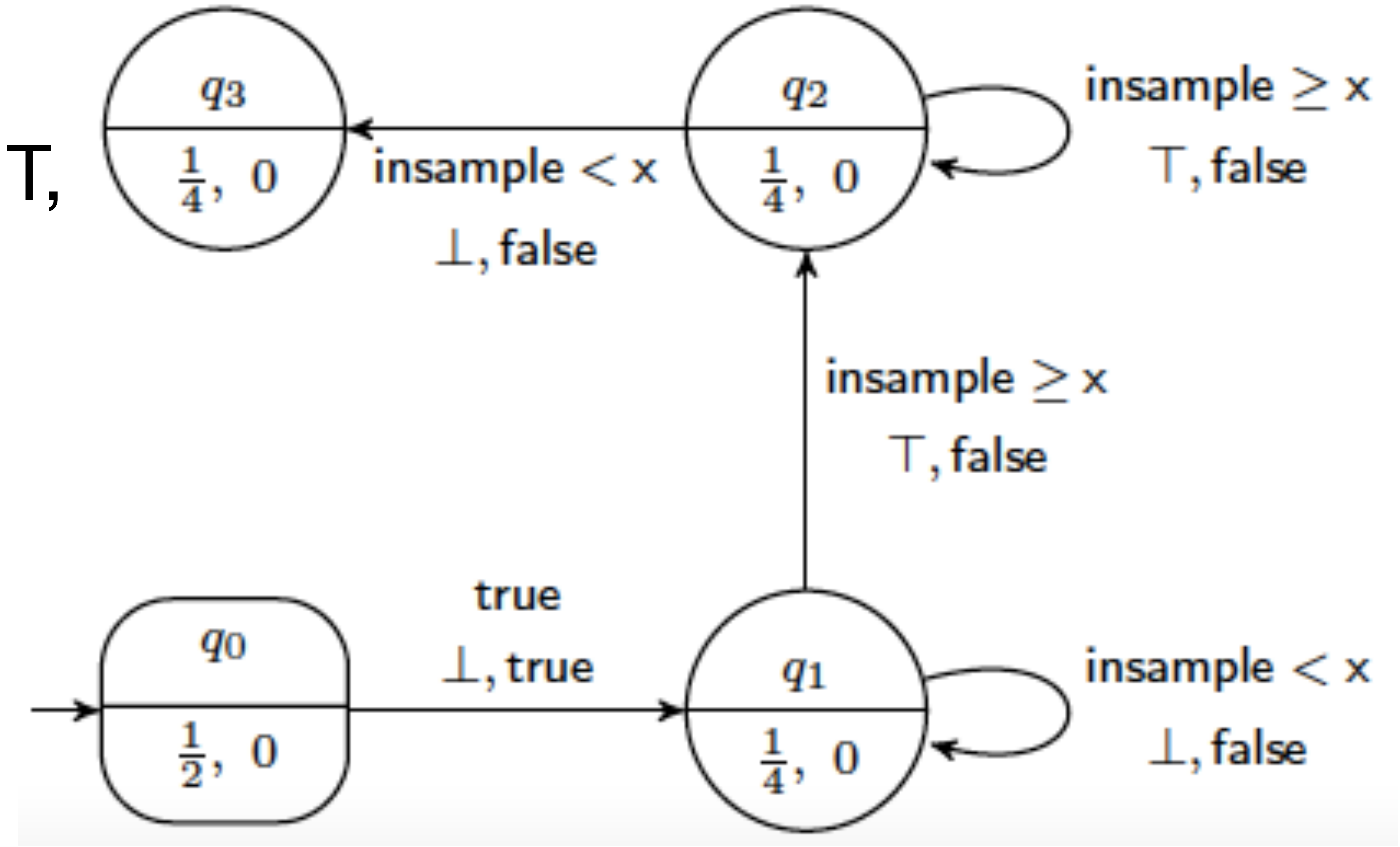
Leaking Pair

Leaking Pair: A pair of cycles (C_1, C_2) connected by a path ρ such that

- C_1 and C_2 have no assignment transitions,
- C_1 has a transition with guard insample < x and C_2 has a transition with guard insample ≥ x (or vice versa)
- Assignment transitions in ρ ensure that value in x is at least (or at most) what it is in C_1

Example: First checks if queries are less than threshold T, then if they are greater than T, and then stops at first query that is less T again.

Self loop on q_1 and self loop on q_2 form a leaking pair.



Characterizing Differentially Private DiPA

Characterizing Differentially Private DiPA

Well-formed DiPA: $\mathscr{A}$ is well formed if it has no reachable leaking cycle or leaking pair of cycles.

Characterizing Differentially Private DiPA

Well-formed DiPA: $\mathscr{A}$ is well formed if it has no reachable leaking cycle or leaking pair of cycles.

Theorem: $\mathscr{A}$ is differentially private if and only if it is well formed.

Not well formed DiPA $\Rightarrow$ not differentially private

Not well formed DiPA $\Rightarrow$ not differentially private

Observation 1: For any path ρ, if the means of samples are set to be consistent with the guards along a path, then for sufficiently large ϵ, $\text{Pr}[\epsilon, \rho] \geq 3/4$.

Not well formed DiPA $\Rightarrow$ not differentially private

Observation 1: For any path ρ, if the means of samples are set to be consistent with the guards along a path, then for sufficiently large ϵ, $\Pr[\epsilon, \rho] \geq 3/4$.

Observation 2: Leaking cycle or pair identify a pair of repeatable transitions such that the value sampled in one transition must be less than the other.

Not well formed DiPA $\Rightarrow$ not differentially private

Observation 1: For any path ρ, if the means of samples are set to be consistent with the guards along a path, then for sufficiently large ϵ, Pr$[\epsilon, \rho] \geq 3/4$.

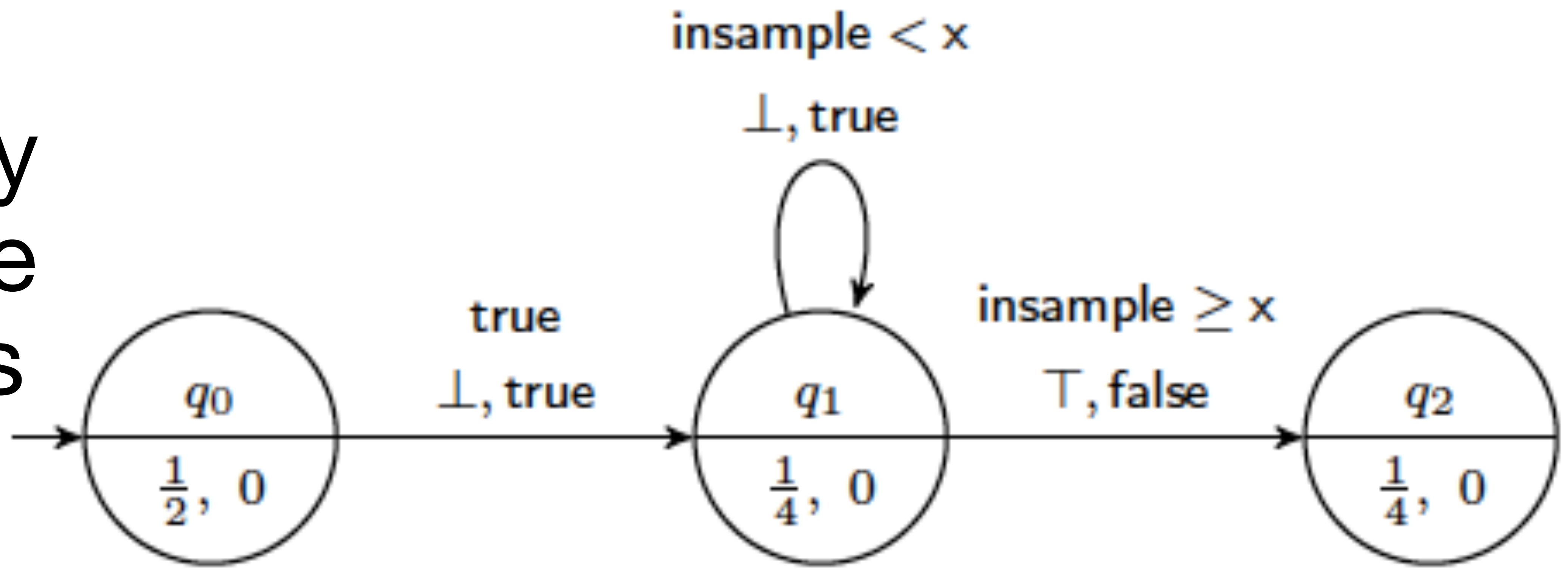
Observation 2: Leaking cycle or pair identify a pair of repeatable transitions such that the value sampled in one transition must be less than the other.

Observation 3: If the means of these transitions are set in opposite order of their guards, then the probability of the path corresponding to sufficiently many repetitions of the cycle can be upper bounded by as small a number as desired.

Not well formed DiPA $\Rightarrow$ not differentially private

Observation 1: For any path ρ, if the means of samples are set to be consistent with the guards along a path, then for sufficiently large ϵ, $\Pr[\epsilon, \rho] \geq 3/4$.

Observation 2: Leaking cycle or pair identify a pair of repeatable transitions such that the value sampled in one transition must be less than the other.



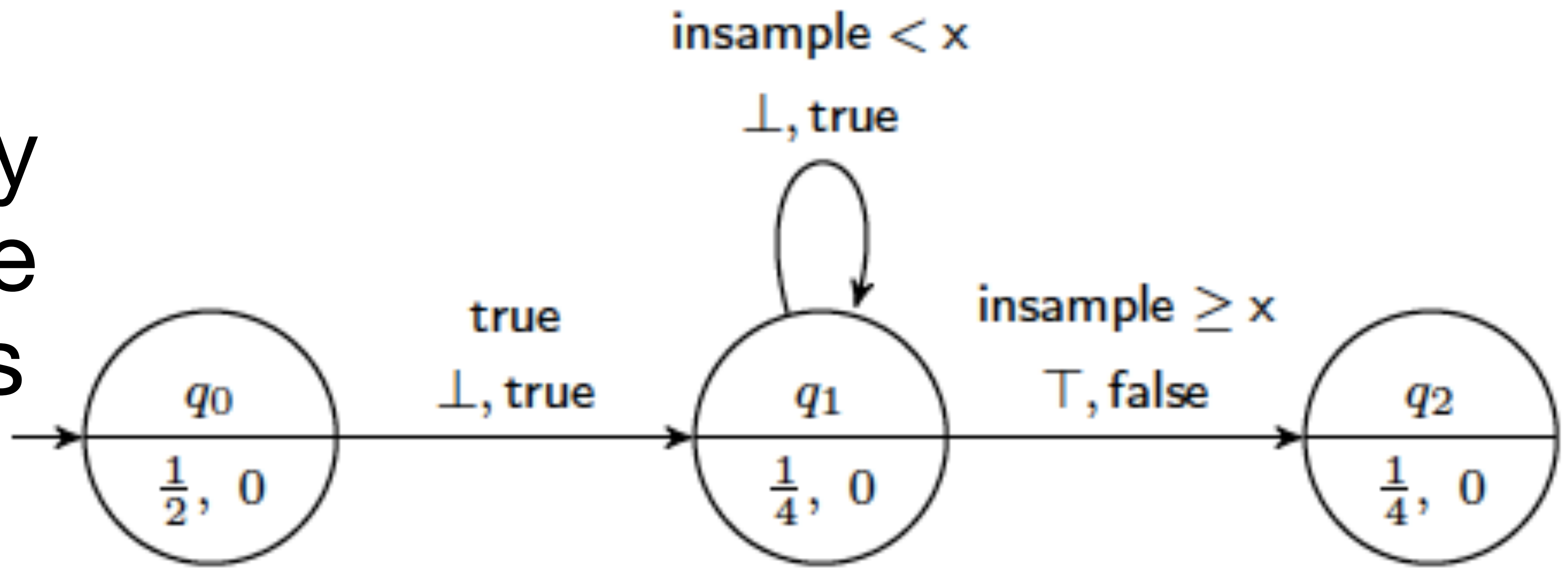
Observation 3: If the means of these transitions are set in opposite order of their guards, then the probability of the path corresponding to sufficiently many repetitions of the cycle can be upper bounded by as small a number as desired.

Not well formed DiPA $\Rightarrow$ not differentially private

Observation 1: For any path ρ, if the means of samples are set to be consistent with the guards along a path, then for sufficiently large ϵ, Pr$[\epsilon, \rho] \geq 3/4$.

- Set $\rho_1^n = q_0 \xrightarrow{0,\perp} q_1 \xrightarrow{-1,\perp} q_1 \xrightarrow{-2,\perp} q_1 \xrightarrow{-3,\perp} q_1 \xrightarrow{-4,\perp} q_1 \cdots$

Observation 2: Leaking cycle or pair identify a pair of repeatable transitions such that the value sampled in one transition must be less than the other.



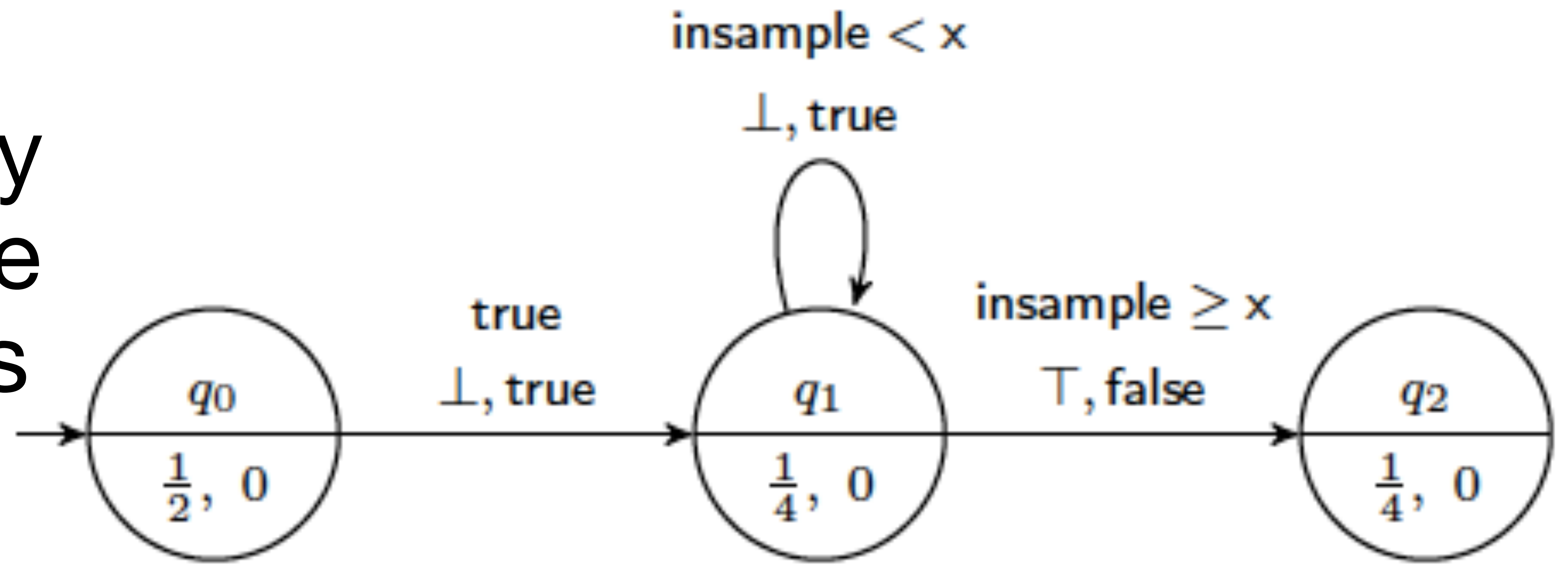
Observation 3: If the means of these transitions are set in opposite order of their guards, then the probability of the path corresponding to sufficiently many repetitions of the cycle can be upper bounded by as small a number as desired.

Not well formed DiPA $\Rightarrow$ not differentially private

Observation 1: For any path ρ, if the means of samples are set to be consistent with the guards along a path, then for sufficiently large ϵ, Pr$[\epsilon, \rho] \geq 3/4$.

- Set $\rho_1^n = q_0 \xrightarrow{0,\perp} q_1 \xrightarrow{-1,\perp} q_1 \xrightarrow{-2,\perp} q_1 \xrightarrow{-3,\perp} q_1 \xrightarrow{-4,\perp} q_1 \cdots$

Observation 2: Leaking cycle or pair identify a pair of repeatable transitions such that the value sampled in one transition must be less than the other.



Observation 3: If the means of these transitions are set in opposite order of their guards, then the probability of the path corresponding to sufficiently many repetitions of the cycle can be upper bounded by as small a number as desired.

- Set $\rho_2^n = q_0 \xrightarrow{0,\perp} q_1 \xrightarrow{-2,\perp} q_1 \xrightarrow{-1,\perp} q_1 \xrightarrow{-4,\perp} q_1 \xrightarrow{-3,\perp} q_1 \cdots$

Well formed $\Rrightarrow$ Differentially Private

Well formed $\Rightarrow$ Differentially Private

Critical Transition: A transition that is not part of a cycle.

Well formed $\Rightarrow$ Differentially Private

Critical Transition: A transition that is not part of a cycle.

Cost of a transition t: If t is not critical, then cost is 0. If t is critical and source state of t is a non-input state, then cost of t is d. Else it is $2d$.

Well formed $\Rightarrow$ Differentially Private

Critical Transition: A transition that is not part of a cycle.

Cost of a transition t: If t is not critical, then cost is 0. If t is critical and source state of t is a non-input state, then cost of t is d. Else it is $2d$.

Weights: Weight of a path ρ is sum of costs of all transitions in ρ. Weight of $\mathscr{A}$ is maximum weight of path in $\mathscr{A}$.

Well formed $\Rightarrow$ Differentially Private

Critical Transition: A transition that is not part of a cycle.

Cost of a transition t: If t is not critical, then cost is 0. If t is critical and source state of t is a non-input state, then cost of t is d. Else it is $2d$.

Weights: Weight of a path ρ is sum of costs of all transitions in ρ. Weight of $\mathscr{A}$ is maximum weight of path in $\mathscr{A}$.

Theorem: A well formed DiPA $\mathscr{A}$ is $\mathsf{wt}(\mathscr{A})\epsilon$-differentially private.

Well formed $\Rightarrow$ Differentially Private

Critical Transition: A transition that is not part of a cycle.

Cost of a transition t: If t is not critical, then cost is 0. If t is critical and source state of t is a non-input state, then cost of t is d. Else it is $2d$.

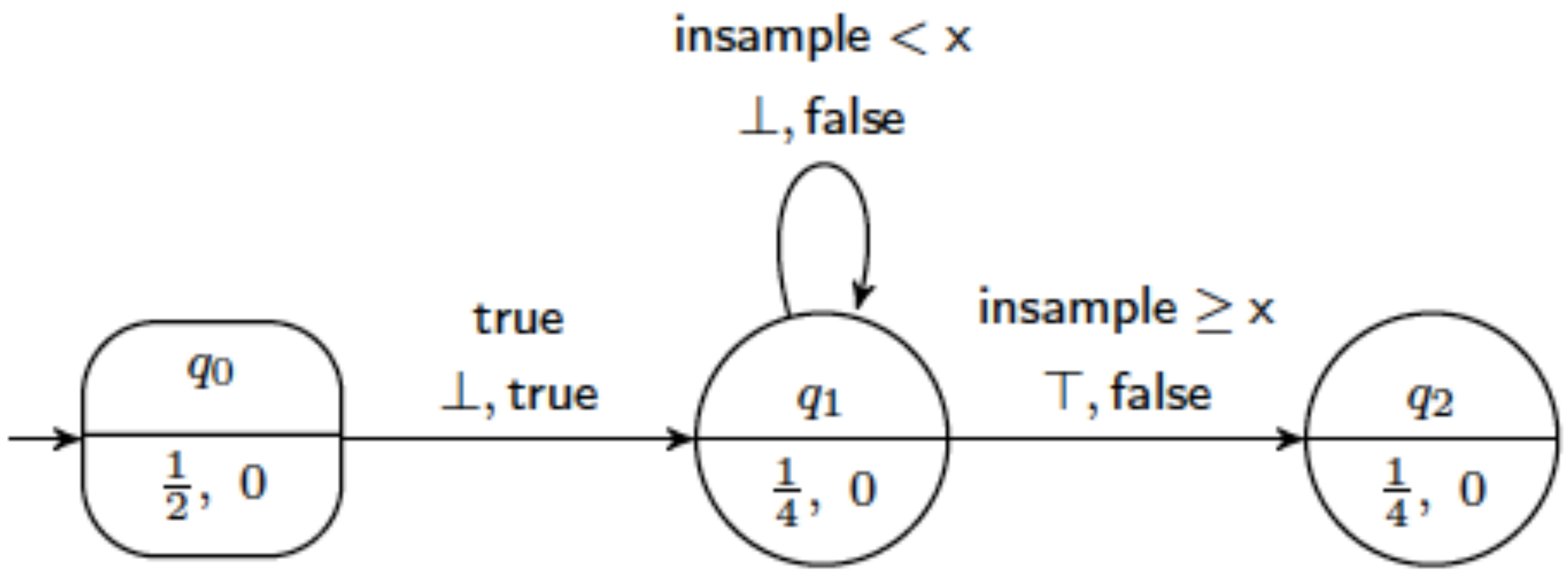
Weights: Weight of a path ρ is sum of costs of all transitions in ρ. Weight of $\mathscr{A}$ is maximum weight of path in $\mathscr{A}$.

Theorem: A well formed DiPA $\mathscr{A}$ is wt$(\mathscr{A})\epsilon$-differentially private.

Example: cost(t_{11}) = 0, cost(t_{01}) = 1/2, cost(t_{12}) = 2*(1/4)=1/2.

wt$(\mathscr{A})$ = 1/2 + 1/2 = 1.

Thus $\mathscr{A}$ is ϵ-differentially private.



Linear time checking of well formedness

Linear time checking of well formedness

One can check if a DiPA is well formed in linear time. If it is well formed, the weight can be computed in linear time, assuming constant time for arithmetic operations.

Linear time checking of well formedness

One can check if a DiPA is well formed in linear time. If it is well formed, the weight can be computed in linear time, assuming constant time for arithmetic operations.

- For example, to check if there is a reachable leaking cycle, check if there is a reachable SCC that has an assignment transition, and a transition whose guard compares the value stored in x.

Real valued outputs

The results on checking differential privacy can be extended to DiPA with real valued outputs.


CONCLUSIONS

In a nutshell …

In a nutshell …

- We presented a automata model that can describe some differential privacy algorithms

In a nutshell …

- We presented a automata model that can describe some differential privacy algorithms
- Checking differential privacy of such algorithms for an unbounded sequence of inputs is decidable in linear time

In a nutshell …

- We presented a automata model that can describe some differential privacy algorithms
- Checking differential privacy of such algorithms for an unbounded sequence of inputs is decidable in linear time
- Algorithm relies on checking conditions on the underlying graph of the automaton

In a nutshell …

- We presented a automata model that can describe some differential privacy algorithms
- Checking differential privacy of such algorithms for an unbounded sequence of inputs is decidable in linear time
- Algorithm relies on checking conditions on the underlying graph of the automaton
  - Parameters of distributions don’t play a role: differential privacy ensured even if these are changed in the algorithm

In a nutshell …

- We presented a automata model that can describe some differential privacy algorithms
- Checking differential privacy of such algorithms for an unbounded sequence of inputs is decidable in linear time
- Algorithm relies on checking conditions on the underlying graph of the automaton
    - Parameters of distributions don’t play a role: differential privacy ensured even if these are changed in the algorithm
    - Computed values of “d” for algorithms match the theoretical bounds known

Possible next steps

- Results can be extended to automata with multiple storage variables and richer guard conditions
- How tight is the value of “d” computed by the algorithm?
- Checking other properties of automata-like models
  - $d\epsilon$-differential privacy
  - Accuracy