

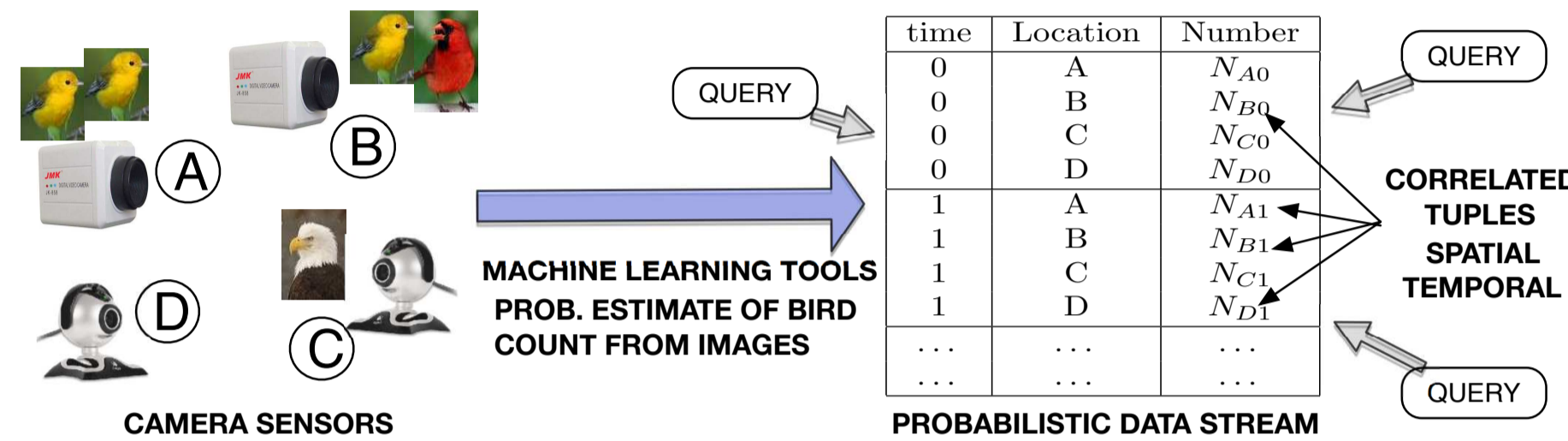
EFFICIENT QUERY EVALUATION ON TEMPORALLY CORRELATED PROBABILISTIC STREAMS

Bhargav Kanagal and Amol Deshpande
University of Maryland, College Park

1. Motivation

Correlated probabilistic streams commonly occur in a large variety of applications.

A **Habitat Monitoring Application**



Problem: Execute *continuous* queries over *probabilistic streams*

Example queries:

- What is the weekly average of the bird count at location A ?
- What is the likelihood that A has more birds than B for the past week ?
- List all days in which A has more than 100 birds

Issues:

- Correlations significantly influence results of query evaluation
- SQL query semantics are ambiguous when results are probabilistic sequences

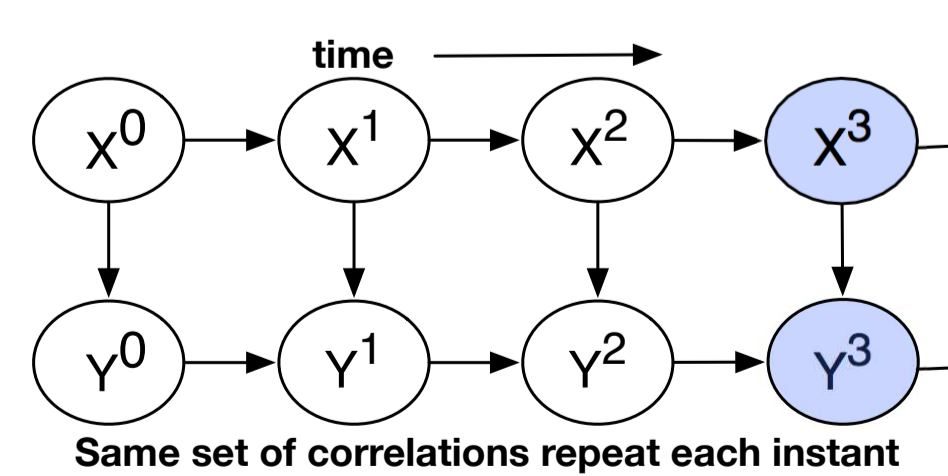
2. Previous work

- Probabilistic databases (*Mystiq*, *Trio*, *Orion*, *MayBMS*) focus on static data and not streams, also cannot handle these complex correlations
- *Lahar*, *Caldera* are applicable to probabilistic streams, but focus on pattern identification queries

3. Our Approach

Observation: Many naturally occurring probabilistic streams are both *structured* and *Markovian* in nature.

- **Structured:** Same set of correlations and independences repeat across time
- **Markovian:** The values of variables at time $t + 1$ is independent of those at time $t - 1$ given their values at time t



We exploit this to:

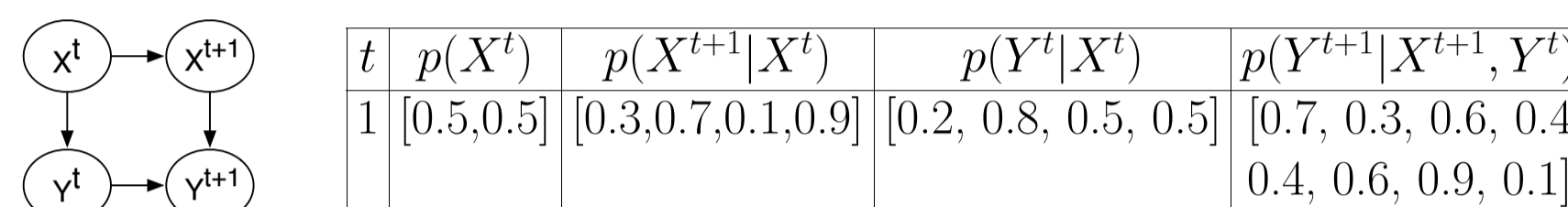
- Design **compact, lightweight data structures** to represent and modify them
- Enable **incremental query processing** using the iterator framework

A Markov sequence as a PGM

4. Markov sequence

- **Special case** of a probabilistic sequence. Completely determined by specifying successive joint distributions for all time instants $p(V_t, V_{t+1})$

- **Efficient Representation:** We represent Markov sequences using a combination of the schema graph and the actual data



(a) Schema graph (b) Distributions representing the Markov sequence

- **Efficient Query Processing:** Only transmit these numbers between operators. Only operators with the knowledge of the schema can interpret these numbers.

5. Formal Semantics

Possible World Semantics (+ small modification for *sequences*)

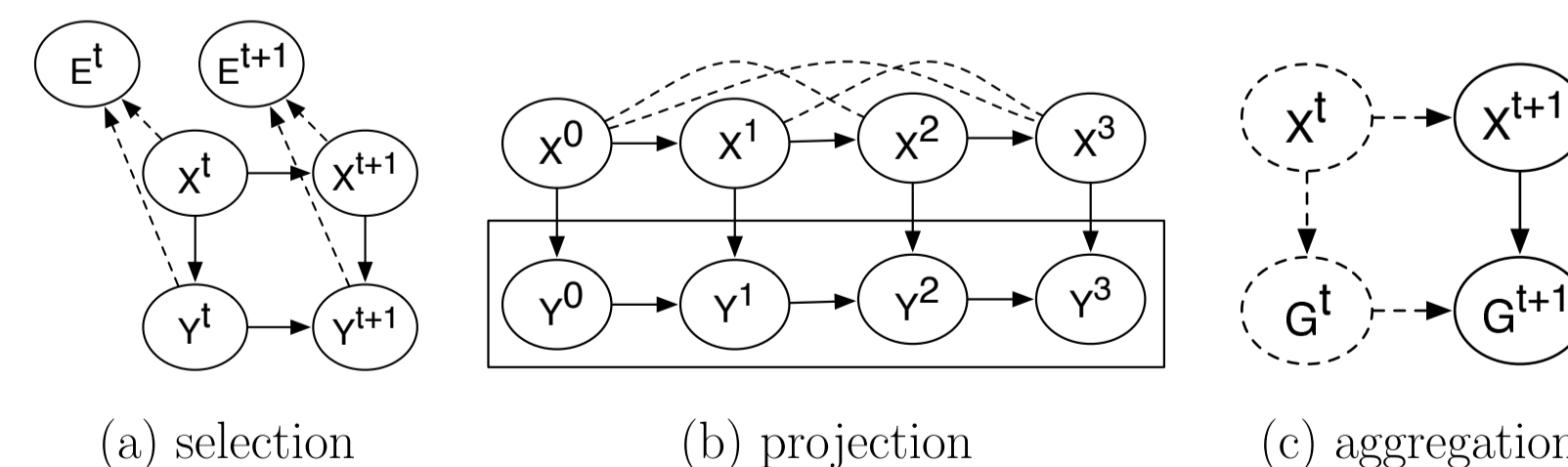
- Operators: *Select*, *Project*, *Joins*, *Aggregate*, *Windowing*
- *MAP*, *ML*: Convert a Markovian sequence into a deterministic sequence
- The set of Markov sequences is not closed under these operators, i.e., some operators return non-Markovian sequences (*Projection*, *Windowing*)
- *Op* is safe for input schema $I \Leftrightarrow Op(I)$ is a Markovian sequence

6. Operator Design

- **Schema routine:** Output schema is computed based on the input schema
- **get_next() routine:** Operates on each of the input data tuples to compute the output tuples.

Selection: Add new boolean random variable to each slice; Always safe

Projection: Eliminate all variables not of interest; Can be unsafe for some schemas



- **Join:** Combine the schemas, concatenate the tuples; Always safe
- **Aggregation:** Maintain the joint distribution of the aggregate variable along with the other variables in the time slice; Always safe

$$p(X^{t+1}, G^{t+1}) = \sum_{X^t, G^t} p(X^t, G^t) p(X^{t+1}|X^t) p(G^{t+1}|G^t, X^{t+1})$$

Operator Design (contd)

Windowing:

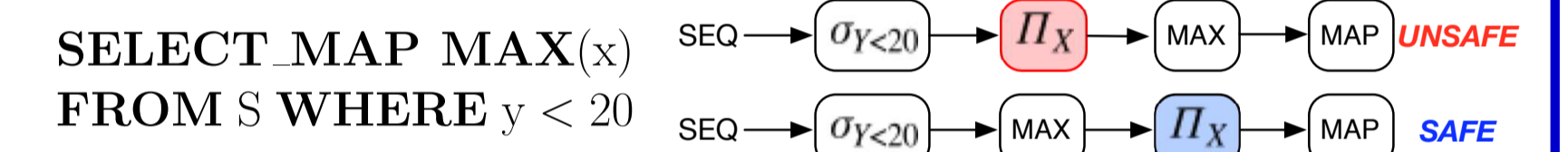
- Sliding window operator is always unsafe (We use approximations)
- For the special case when shift = window size, we can reduce the windowing operation to a projection

ML: Simply marginalize the joint distribution

MAP: Viterbi-style dynamic programming

7. Query Planning Algorithm

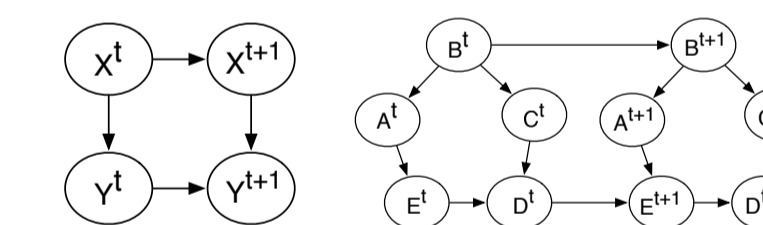
- Unsafe Operators: Projection, Window operator reduced to a projection
- Query Planning = Determining the *correct position for Projection*
- Strategy: Pull up Projection operator until it is safe. If no safe position for projection, check its parent: ML \Rightarrow we can determine a safe plan, MAP \Rightarrow we cannot determine a safe plan



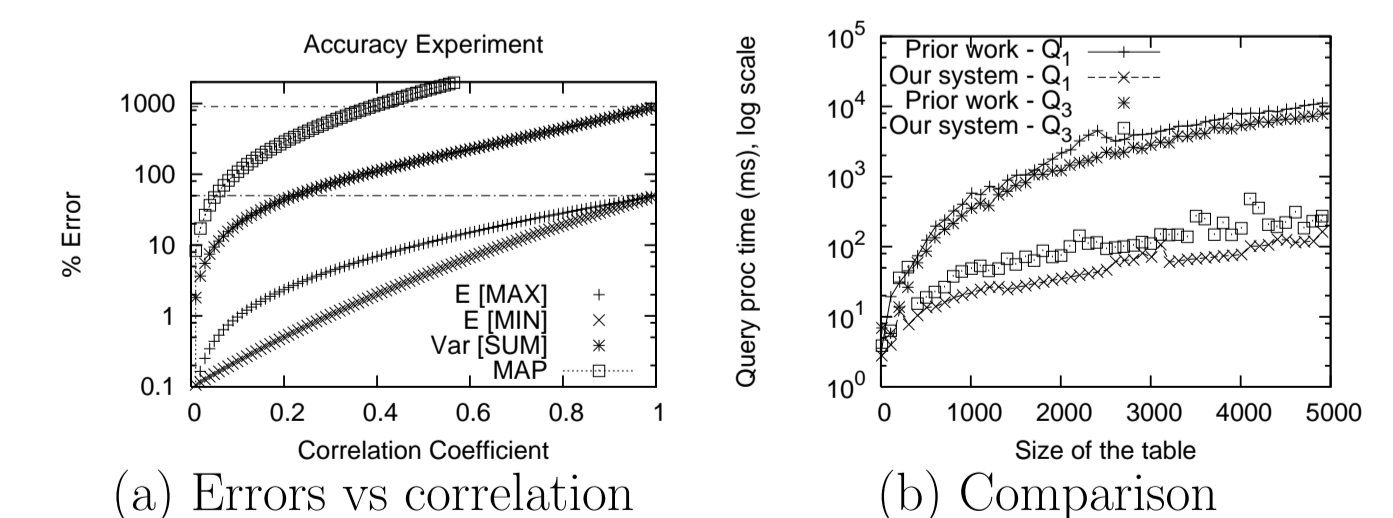
- The query planning algorithm is *sound* and *complete*

8. Results

Markov sequence generator: Generate Markov sequences for different schemas with correlations ranging from (indep., perfectly correlated)



- Capturing and reasoning about temporal correlations is critical for obtaining accurate query answers
- Our incremental operators can process up to 500 tuples per second, for domain sizes below 200
- Our get_next() query processing framework that exploits the structure in Markov sequences is much more efficient than previous generic approaches



(a) Errors vs correlation (b) Comparison

Q1: **SELECT MAP Agg(A) FROM S**

Q3: **SELECT MAP MAX(A) FROM S[size,size]**