DEPARTMENT OF COMPUTER SCIENCE
UNIVERSITY OF OXFORD
LogicBlox

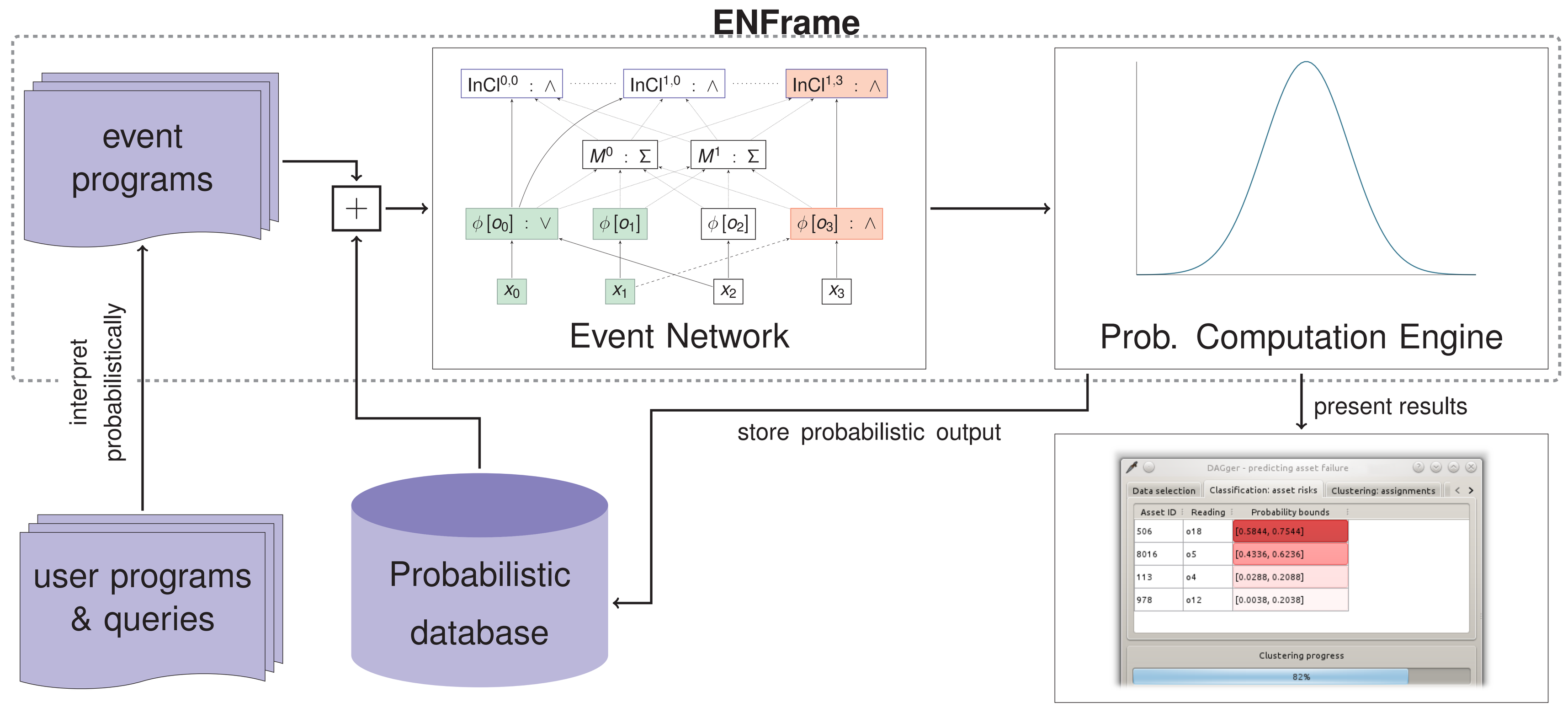
ENFrame = Programs + Queries / Probabilistic Data

Dan Olteanu & Sebastiaan J. van Schaik

Big Uncertain Data (BUDA), June 2014

ENFrame at a glance

- Users write code in a subset of Python + relational queries.
- User code is oblivious to the probabilistic nature of the data: ENFrame interprets the code and runs it on probabilistic data.
- We already tested ENFrame on several algorithms, e.g., k-medoids clustering and k-nearest neighbour classification applied to probabilistic data representing query results.



Key Technical Features of ENFrame

- Language to express probabilistic events that capture arbitrary correlations in the input data and in the output as induced by program traces.
- Sequential and parallel algorithms for exact and approximate probabilistic computation of user programs.
- Quality metric for probabilistic computation using, e.g., ENFrame, naïve and sampling-based methods.

Exact and Approximate Probability Computation

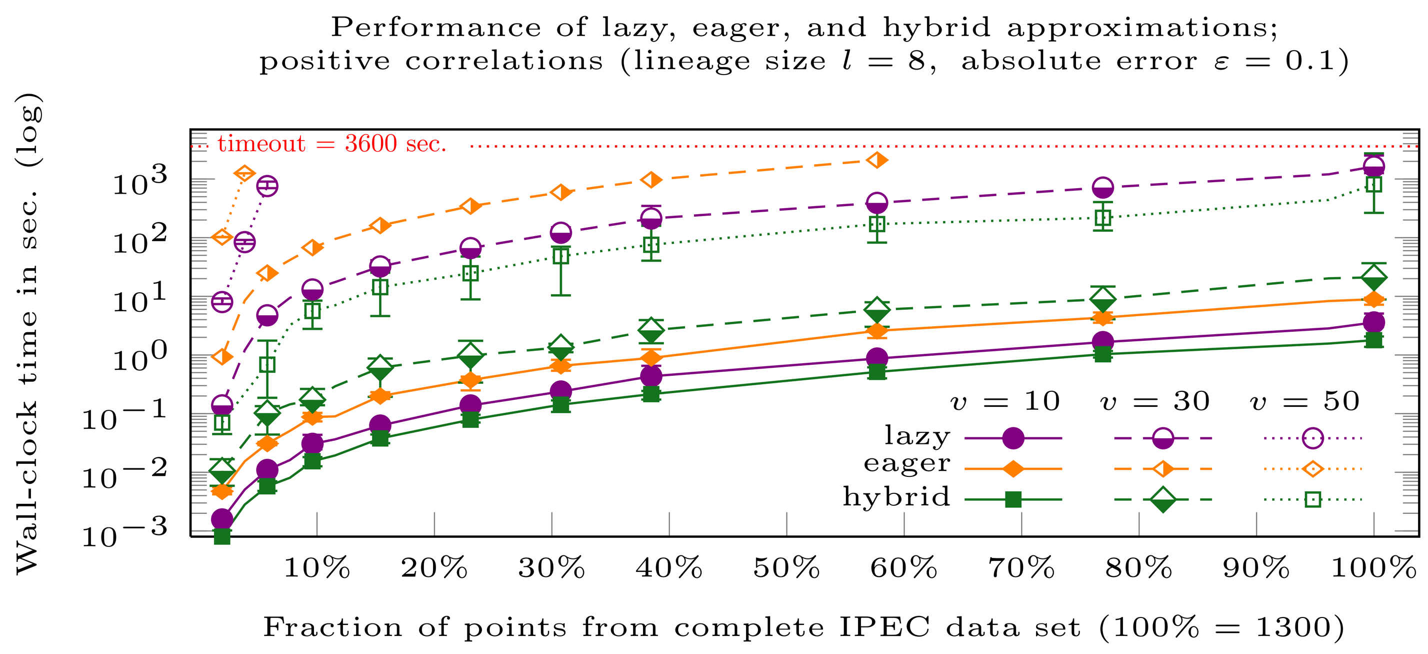
- ENFrame can compute, e.g., the probability that two objects are in the same cluster (k-medoids), or that an object is assigned to a class (k-NN).
- Main idea: exhaustive/partial structural decomposition of an entire event network.
  - Outcome equivalent to clustering in every possible world.
  - Multiple pruning strategies for approximate probability computation with error guarantees.
  - Efficient parallel computation on multi-cores.

Event Language, Event Programs, and Event Network

- **Event program**:
  - ENFrame's probabilistic interpretation of the user program is captured by *events*.
- **Event language**:
  - Event = random variables, conditioned values, Boolean formulas over events, arithmetic operations over events.
- Variables of type T from user program become discrete random variables with pdfs over values in T.
- c-values: values (numbers/vectors) conditioned on events.
  - For a number v and variable Φ:
    c-value $\Phi \otimes v$ evaluates to v if Φ is true, or 0 otherwise.
  - c-values can be summed and compared:
    $\Phi_1 \otimes v_1 + \ldots + \Phi_n \otimes v_n \quad \leq \quad \Psi_1 \otimes w_1 + \ldots + \Psi_m \otimes w_m$
  - Application example: sum of distances between objects.
- **Event Network**: Joint representation of interconnected events.

Experimental Evaluation

Comparison of approximation algorithms for k-medoids ($k = 3$), input data with positive correlations and varying number of variables (v). Further experiments in [vSOF14].



Example: User Program and Event Program for k-medoids Clustering

```
#Initialisation phase: Select k cluster medoids (centres)
(O, n) = loadData()          # list and number of objects
(k, iter) = loadParams()     # number of clusters and iterations
M = init()                   # initialise medoids

for it in range(0,iter):     # clustering iterations
  #Assignment phase: assign objects to closest medoid
  InCl = [None] * k
  for i in range(0,k):
   InCl[i] = [None] * n
   for l in range(0,n):
    InCl[i][l] = reduce_and(
       [(dist(O[l],M[i]) <= dist(O[l],M[j])) for j in range(0,k)])
  InCl = breakTies2(InCl)    # each object is in exactly one cluster

  #Update phase: Select new cluster medoids
  DistSum = [None] * k
  for i in range(0,k):
   DistSum[i] = [None] * n
   for l in range(0,n):
    DistSum[i][l] = reduce_sum(
      [dist(O[l],O[p]) for p in range(0,n) if InCl[i][p]])

  Centre = [None] * k
  for i in range(0,k):
   Centre[i] = [None] * n
   for l in range(0,n):
    Centre[i][l] = reduce_and(
      [DistSum[i][l] <= DistSum[i][p] for p in range(0,n)])
  Centre = breakTies1(Centre)  # enforce one Centre per cluster

  M = [None] * k
  for i in range(0,k):
   M[i] = reduce_sum([O[l] for l in range(0,n) if Centre[i][l]])
```

$\forall i$ in $0..n-1$: $O^i \equiv \Phi(o_i) \otimes \vec{o}_i$
$M^0_{-1} \equiv \Phi(o_{\pi(0)}) \otimes \vec{o}_{\pi(0)}; \ldots; M^{k-1}_{-1} \equiv \Phi(o_{\pi(k-1)}) \otimes \vec{o}_{\pi(k-1)}$

$\forall$it in $0..$iter -1 :

$\forall i$ in $0..k-1$:
$\forall l$ in $0..n-1$:
$\text{InCl}^{i,l}_{\text{it}} \equiv \bigwedge_{j=0}^{k-1} [\text{dist}(O^l, M^i_{\text{it}-1}) \leq \text{dist}(O^l, M^j_{\text{it}-1})]$

Encoding of `breakTies2` omitted

$\forall i$ in $0..k-1$:
$\forall l$ in $0..n-1$:
$\text{DistSum}^{i,l}_{\text{it}} \equiv \sum_{p=0}^{n-1} \text{InCl}^{i,p}_{\text{it}} \otimes \text{dist}(O^l, O^p)$

$\forall i$ in $0..k-1$:
$\forall l$ in $0..n-1$:
$\text{Centre}^{i,l}_{\text{it}} \equiv \bigwedge_{p=0}^{n-1} [\text{DistSum}^{i,l}_{\text{it}} \leq \text{DistSum}^{i,p}_{\text{it}}]$

Encoding of `breakTies2` omitted

$\forall i$ in $0..k-1$:
$M^i_{\text{it}} = \sum_{l=0}^{n-1} \text{Centre}^{i,l}_{\text{it}} \wedge O^l$

Challenges Currently under Microscope

- Which additional event language constructs are needed to capture further data analysis tasks?
- Trade-off: functionality (event-based result explanation, sensitivity analysis) vs. performance (coarse events compiled to C++ code)?

[vSOF14] Sebastiaan J. van Schaik, Dan Olteanu, and Robert Fink. *ENFrame: A Platform for Processing Probabilistic Data* (in: EDBT 2014)