KnoBAB

Making Logic Fast

Giacomo Bergami

Newcastle University, England, United Kingdom

LT$_f$@AAAI-SSS, 29th of March, 2023

Who Are We?

Assistant Prof. (Lecturer)

Dr Giacomo Bergami
[ʤaːkomo beːrgami]

PhD Student

Mr Samuel Appleby

Full Professor

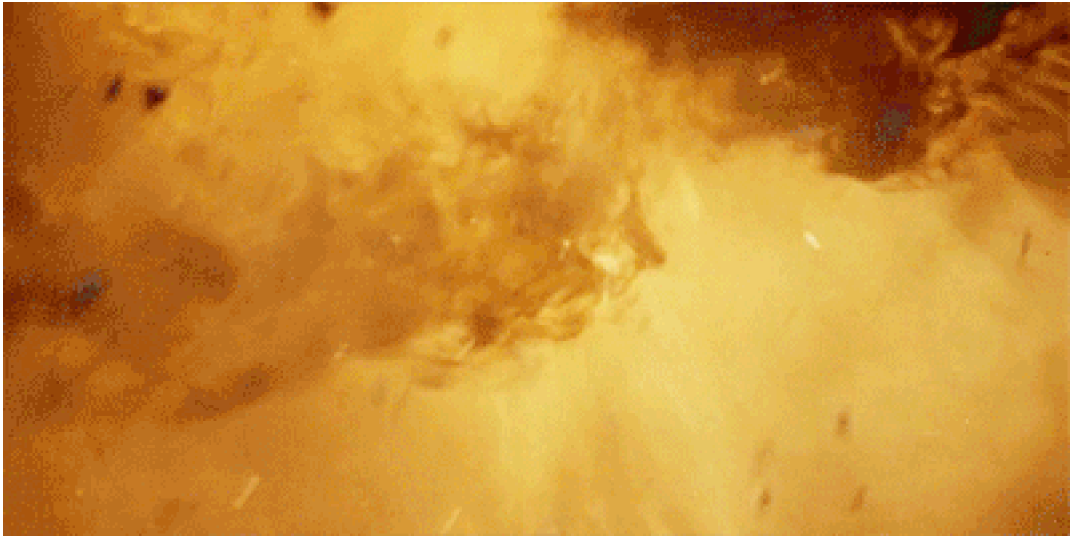
Prof Graham Morgan

References

- Samuel Appleby, Giacomo Bergami, and Graham Morgan.
  Running temporal logical queries on the relational model.
  In *IDEAS'22*, pages 134–143. ACM, 2022.
- Samuel Appleby, Giacomo Bergami, and Graham Morgan.
  Quickening data-aware conformance checking through temporal algebras.
  In *Information (Switzerland)*, volume 14, page 60, 2023.
- Samuel Appleby, Giacomo Bergami, and Graham Morgan.
  Enhancing declarative temporal model mining in relational databases: A preliminary study.
  In *IDEAS'23 [In Press]*, 2023.

And now. . . (1971)

And now. . . (1971)



What is KnoBAB? (1/2)

- Bridging the gap between theoretical research and practical use case scenarios.
- To the best of our knowledge, current approaches in the field have at least one of the following shortcomings:
  1. Do not reap big data algorithms for carrying out computations efficiently
  2. Most of the time, the users are forced to “reduce” a data-aware domain to a dataless one to carry out business process mining operations (e.g., trace repairs or alignments).
  3. KnoBAB shows that it is possible to import a lot of good practices from database field to the benefit of a larger community.
  4. Either assume one fixed declarative language (*Process Query Language*), or are mainly GUI-driven (*ProM*, *RuM*).

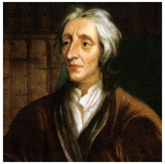
What is KnoBAB? (2/2)

In order to perform optimizations, we performed the following assumptions:

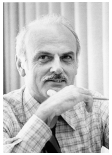
- We are not interested in empty traces, as they have no events!
- Traces' payloads can be represented as a distinctive event (`__trace_payload`) appearing at the beginning of the trace.
- Differently from the current standard, keys are not duck typed, but are associated to only one specific data-type!
- Boolean and Integer representation are mapped into double precision floating points.
- For each database, we consider a minimum and maximum possible string value length.

What is xtLTL$_f$?

- Despite LTL$_f$ describes a way to calculate the satisfiability of a trace starting from its beginning and providing temporal quantification on specific events, any relational algebra works in the opposite way, from the data being accessed.
  - $\Rightarrow$ next ($\bigcirc\varphi$, $X\varphi$) must be evaluated from φ towards the preceding event, if any.
- Walking on the footsteps of Declare, we might assume to have activation and target condition to be tested. Differently from LTL$_f$, we also need to explicitly express data correlation conditions.

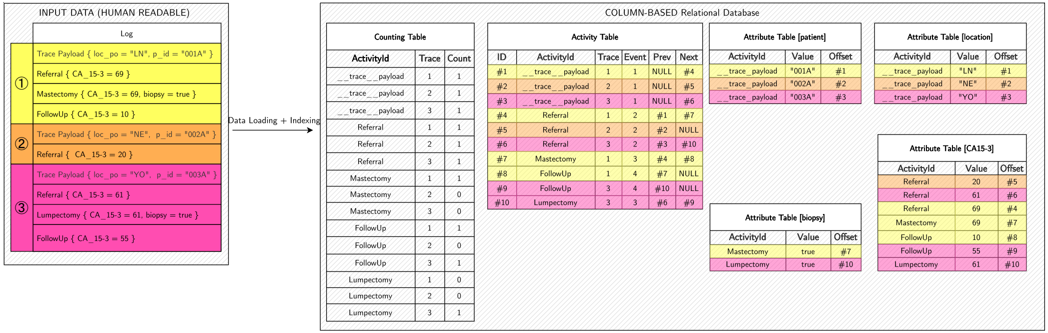


"Logic is the anatomy of thought"
John Locke (1632-1704)



Edgar Codd (1923-2003)

I. Column-Store Data Representation



INPUT DATA (HUMAN READABLE)

	Log
①	Trace Payload { loc_po = "LN", p_id = "001A" }
	Referral { CA_15-3 = 69 }
	Mastectomy { CA_15-3 = 69, biopsy = true }
	FollowUp { CA_15-3 = 10 }
②	Trace Payload { loc_po = "NE", p_id = "002A" }
	Referral { CA_15-3 = 20 }
③	Trace Payload { loc_po = "YO", p_id = "003A" }
	Referral { CA_15-3 = 61 }
	Lumpectomy { CA_15-3 = 61, biopsy = true }
	FollowUp { CA_15-3 = 55 }

Data Loading + Indexing

COLUMN-BASED Relational Database

Counting Table

ActivityId	Trace	Count
__trace__payload	1	1
__trace__payload	2	1
__trace__payload	3	1
Referral	1	1
Referral	2	1
Referral	3	1
Mastectomy	1	1
Mastectomy	2	0
Mastectomy	3	0
FollowUp	1	1
FollowUp	2	0
FollowUp	3	1
Lumpectomy	1	0
Lumpectomy	2	0
Lumpectomy	3	1

Activity Table

ID	ActivityId	Trace	Event	Prev	Next
#1	__trace__payload	1	1	NULL	#4
#2	__trace__payload	2	1	NULL	#5
#3	__trace__payload	3	1	NULL	#6
#4	Referral	1	2	#1	#7
#5	Referral	2	2	#2	NULL
#6	Referral	3	2	#3	#10
#7	Mastectomy	1	3	#4	#8
#8	FollowUp	1	4	#7	NULL
#9	FollowUp	3	4	#10	NULL
#10	Lumpectomy	3	3	#6	#9

Attribute Table [patient]

ActivityId	Value	Offset
__trace_payload	"001A"	#1
__trace_payload	"002A"	#2
__trace_payload	"003A"	#3

Attribute Table [location]

ActivityId	Value	Offset
__trace_payload	"LN"	#1
__trace_payload	"NE"	#2
__trace_payload	"YO"	#3

Attribute Table [biopsy]

ActivityId	Value	Offset
Mastectomy	true	#7
Lumpectomy	true	#10

Attribute Table [CA15-3]

ActivityId	Value	Offset
Referral	20	#5
Referral	61	#6
Referral	69	#4
Mastectomy	69	#7
FollowUp	10	#8
FollowUp	55	#9
Lumpectomy	61	#10

Column-Base storage work under the assumption that it is always possible to decompose a relation $\Re(\texttt{id}, A_1, \dots, A_n)$ into n tables $\Re_i(\texttt{id}, A_i)$ such that $\bowtie_{1\leq i\leq n}\Re_i = \Re$

For the result representation, we relax the 1NF for representing the result, an ordered sequence of nested records $\langle i, j, L\rangle$ entailing that the j-th event of the i-th trace satisfies the conditions of ρ. L lists all the data activation, target, and Θ correlation conditions being witnessed.

II. Temporal Algebraic Operators ($\mathtt{xtLTL_f}$)

$\mathtt{xtLTL_f}$: Syntax

Given optional A/T fields, an $\mathtt{xtLTL_f}$ expression is defined as follows:

$$\rho ::= \mu | \nu | \beta | \delta$$

$$\mu ::= \mathsf{Activity}_{A/T}^{\mathcal{L},\tau}(\mathsf{a}) | \mathsf{Compound}_{A/T}^{\mathcal{L},\tau}(\mathsf{a}, lower \leq \kappa \leq upper) | \mathsf{First}_{A}^{\mathcal{L},\tau} | \mathsf{Last}_{A}^{\mathcal{L},\tau} | \mathsf{Init}^{\mathcal{L}}(\mathsf{a}) | \mathsf{Ends}^{\mathcal{L}}(\mathsf{a})$$

$$\nu ::= \mathsf{Exists}_n(\rho) | \mathsf{Absence}_n(\rho) | \mathsf{Init}(\rho) | \mathsf{Ends}(\rho) | \mathsf{Next}^{\tau}(\rho) | \mathsf{Globally}^{\tau ?}(\rho) | \mathsf{Future}^{\tau ?}(\rho) | \mathsf{Not}^{\tau ?}(\rho)$$

$$\beta ::= \mathsf{Until}_{\Theta}^{\tau ?}(\rho_1, \rho_2) | \mathsf{And}_{\Theta}^{\tau ?}(\rho_1, \rho_2) | \mathsf{Or}_{\Theta}^{\tau ?}(\rho_1, \rho_2)$$

$$\delta ::= \mathsf{AndFuture}_{\Theta}^{\tau}(\rho_1, \rho_2) | \mathsf{AndGlobally}_{\Theta}^{\tau}(\rho_1, \rho_2) | \mathsf{AndNextGlobally}_{\Theta}^{\tau}(\rho_1, \rho_2)$$

The preliminary results show that $\mathtt{xtLTL_f}$ is at least as expressive as $\mathrm{LTL_f}$ where the distinction between timed and untimed operators perserve the expected "good" properties. We also provided a formalisation of activation, target, and correlation condition for data-aware scenarios.

II. Temporal Algebraic Operators (xtLTL$_f$)

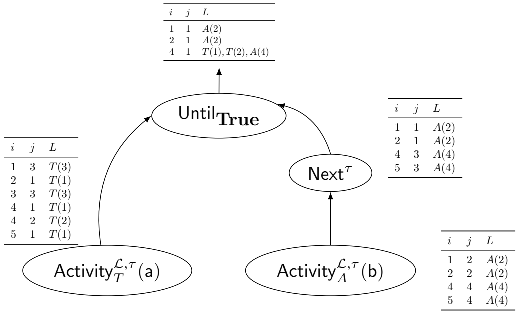
Tutorial

Let's suppose to evaluate $\mathsf{a}\mathcal{U} \bigcirc (\mathsf{b})$ over the following log:

$$\mathcal{L} = \{\mathsf{dbacd}, \mathsf{ab}, \mathsf{cda}, \mathsf{aadb}, \mathsf{addb}\}$$

This can be expressed as

$$\mathsf{Until}_{\mathbf{True}}(\mathsf{Activity}_T^{\mathcal{L},\tau}(\mathsf{a}), \mathsf{Next}^{\tau}(\mathsf{Activity}_A^{\mathcal{L},\tau}(\mathsf{b})))$$



$\mathsf{Until}_{\mathbf{True}}$
Next^{τ}
$\mathsf{Activity}_T^{\mathcal{L},\tau}(\mathsf{a})$
$\mathsf{Activity}_A^{\mathcal{L},\tau}(\mathsf{b})$

i	j	L
1	1	$A(2)$
2	1	$A(2)$
4	1	$T(1), T(2), A(4)$

i	j	L
1	3	$T(3)$
2	1	$T(1)$
3	3	$T(3)$
4	1	$T(1)$
4	2	$T(2)$
5	1	$T(1)$

i	j	L
1	1	$A(2)$
2	1	$A(2)$
4	3	$A(4)$
5	3	$A(4)$

i	j	L
1	2	$A(2)$
2	2	$A(2)$
4	4	$A(4)$
5	4	$A(4)$

III. Conjunctive and Aggregation Queries

Without the need of changing the operators' semantics nor the results being returned, the same operators can express over the same "model" all of the following queries, by simply changing the root node of the query plan:

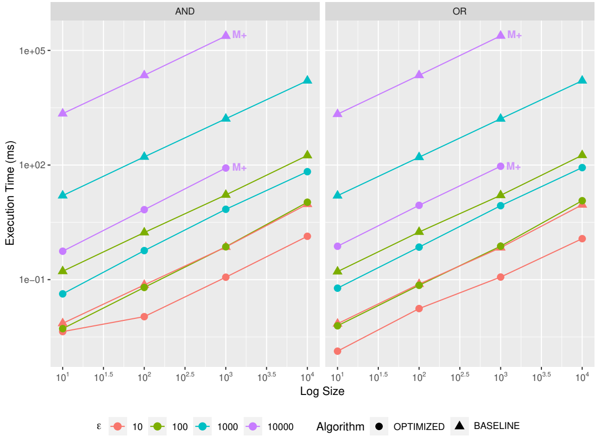
$$\textsc{ConjunctiveQuery}(\rho_1, \dots, \rho_n) = \mathsf{And}_{\mathbf{True}}(\rho_1, \dots, \mathsf{And}_{\mathbf{True}}(\rho_{n-1}, \rho_n))$$

$$\text{Max-SAT}(\rho_1, \dots, \rho_n) = \left(\frac{|\{l | \exists j, L.\, \langle i, j, L \rangle \in \rho_l\}|}{|\mathcal{M}|} \right)_{\sigma^i \in \mathcal{L}}$$

$$\textsc{Confidence}(\rho_1, \dots, \rho_n) = \left(\frac{|\{i | \exists j, L.\, \langle i, j, L \rangle \in \rho_l\}|}{|\mathsf{ActLeaves}(\rho_l)|} \right)_{c_l \in \mathcal{M}}$$

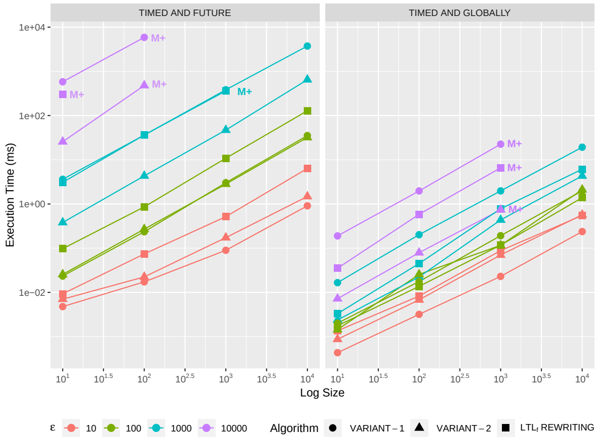
$$\textsc{Support}(\rho_1, \dots, \rho_n) = \left(\frac{|\{i | \exists j, L.\, \langle i, j, L \rangle \in \rho_l\}|}{|\mathcal{L}|} \right)_{c_l \in \mathcal{M}}$$

IV. Query Plan Optimisation (1/4)



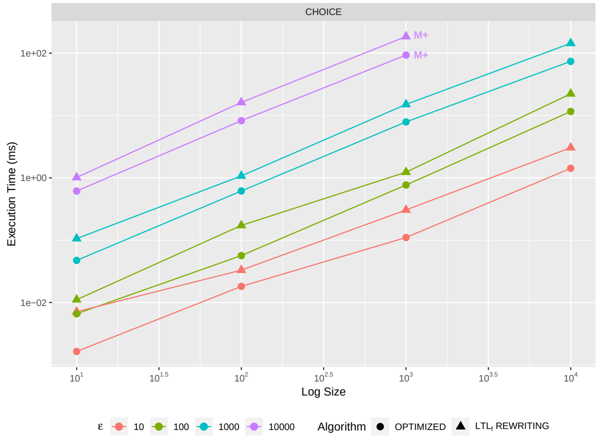
1. For untimed And and Or, we can provide different implementations, one being more memory efficient (Out of Primary Memory) and faster than the other.

IV. Query Plan Optimisation (2/4)



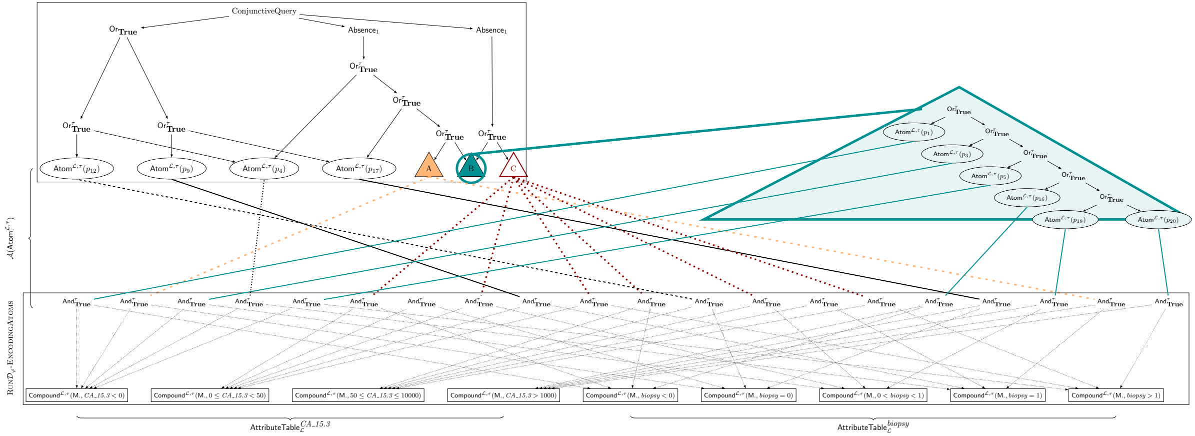
2a. As customary in the database world, by defining *ad hoc* operators subsuming the evaluation of recurrent sub-expressions, we can design better algorithms!

IV. Query Plan Optimisation (3/4)



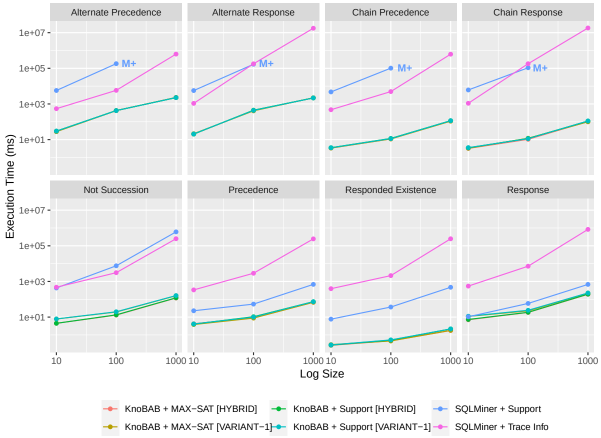
2b. Given that our operators admit $\mathsf{Or}_\Theta(\mathsf{Future}(\rho_1), \mathsf{Future})(\rho_2)) = \mathsf{Or}_\Theta(\rho_1, \rho_2)$ without breaking the "correctness", we can reduce the amount of unnecessary operations as well as memory allocations.

IV. Query Plan Optimisation (4/4)



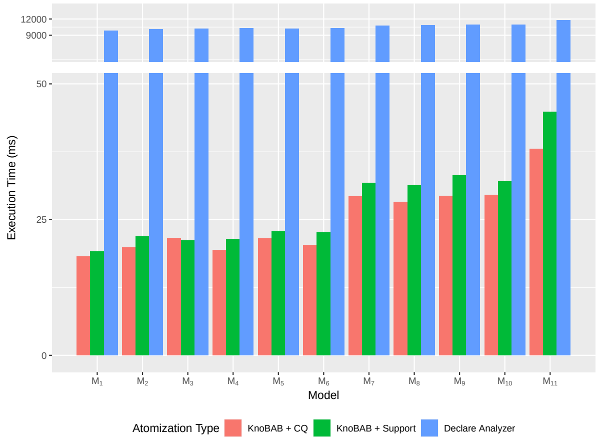
ConjunctiveQuery
Or$^{\mathcal{T}}_{\mathbf{True}}$
Absence$_1$
Absence$_1$
Atom$^{\mathcal{L},\tau}(p_{12})$
Atom$^{\mathcal{L},\tau}(p_9)$
Atom$^{\mathcal{L},\tau}(p_4)$
Atom$^{\mathcal{L},\tau}(p_{17})$
A
B
C
Atom$^{\mathcal{L},\tau}(p_1)$
Atom$^{\mathcal{L},\tau}(p_3)$
Atom$^{\mathcal{L},\tau}(p_5)$
Atom$^{\mathcal{L},\tau}(p_{16})$
Atom$^{\mathcal{L},\tau}(p_{18})$
Atom$^{\mathcal{L},\tau}(p_{20})$
$\mathcal{A}(\mathrm{Atom}^{\mathcal{L},\tau})$
RUN$\mathcal{D}_\varphi$-ENCODINGATOMS
And$^{\mathcal{T}}_{\mathbf{True}}$
Compound$^{\mathcal{L},\tau}$(M., $CA\_15.3 < 0$)
Compound$^{\mathcal{L},\tau}$(M., $0 \leq CA\_15.3 < 50$)
Compound$^{\mathcal{L},\tau}$(M., $50 \leq CA\_15.3 \leq 10000$)
Compound$^{\mathcal{L},\tau}$(M., $CA\_15.3 > 1000$)
Compound$^{\mathcal{L},\tau}$(M., $biopsy < 0$)
Compound$^{\mathcal{L},\tau}$(M., $biopsy = 0$)
Compound$^{\mathcal{L},\tau}$(M., $0 < biopsy < 1$)
Compound$^{\mathcal{L},\tau}$(M., $biopsy = 1$)
Compound$^{\mathcal{L},\tau}$(M., $biopsy > 1$)
AttributeTable$^{CA\_15.3}_{\mathcal{L}}$
AttributeTable$^{biopsy}_{\mathcal{L}}$

V. Temporal Model Mining. . .



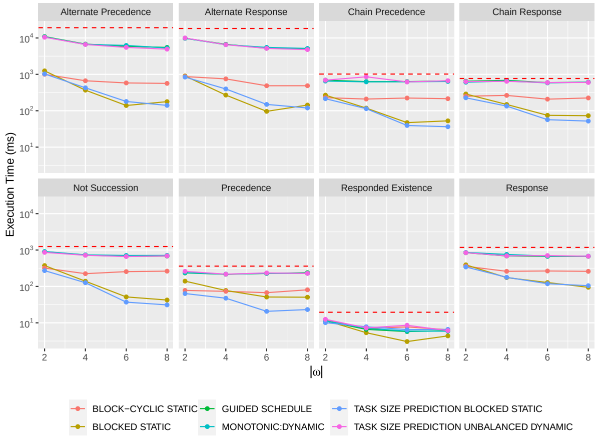
SQLMiner. If compared to other SQL-driven mining algorithms (where a given set of declarative clauses to be tested is previously generated through *propositionalisation*), our results show that our solution outperforms similar querying tasks on PostgreSQL.

V. . . . and Conformance Checking



Declare Analyzer. Our solution also outperforms ad-hoc Java implementations for computing data-aware conformance checking solutions.

VI. Query Plan Parallelisation



The DAG query plan allows running a *topological sort*. By "*layering*" the query plan, we can schedule which operators might be parallelised. Different scheduling policies give different speed-ups exploited by reducing page faults!

VII.-VIII. Customisability/Interoperability

```
load TAB "/home/giacomo/test.tab" as "tab";                    /* Loading Log */

display ACTTABLE for "tab";                                    /* Dump the internal representation as CSV */

auto-timed queryplan "aaai23" {                                /* Setting the semantics for declarative op. */
 template "Example" args 2 := (EXISTS  1 t #2 target) U
                                 (NEXT EXISTS  1 t #1 activation)
};

model-check                                                    /* Conformance Checking */
       declare "Example" ("b", true, "a", true)                /* Manually providing the model */
using "TraceIntersection" over "tab"                           /* Conjunctive query + Log name */
plan "aaai23" with operators "Hybrid"                          /* Semantics to adopt, type of algorithms */
display query-plan;                                            /* Provide the DAG query plan */
```

- By supporting xtLTL$_f$ instead of declarative clauses, our system allows the definition of customary templates (`queryplan`s) that can be loaded at runtime.
- The running example could be run in KnoBAB (`knobab_server`) with the commands given above on a console (`knobab_client`).

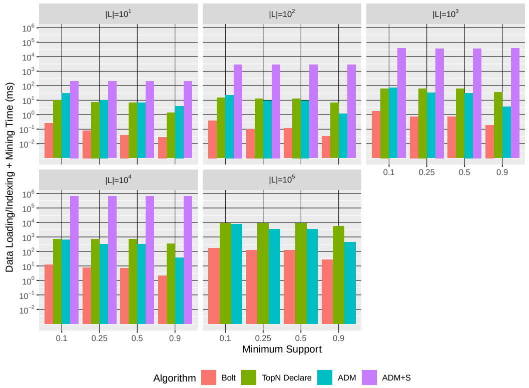
VII.-VIII. Customisability/Interoperability

```
import pandas as pd
from io import StringIO

# Setting up the client
cli = uk_knobab_kinj_KINJ("localhost", 8795)
# Generating a loading data query
q = uk_knobab_kinj_QueryCompiler.LoadDataQuery(uk_knobab_kinj_LogFormat.TAB,
                                               "/home/giacomo/test.tab", "tab",
                                               False,False,False)
# Sending the data loading request to the server
cli.PerformQuery(q)
# Asking to dump the ActivityTable as a CSV string
q = uk_knobab_kinj_QueryCompiler.DisplayData(uk_knobab_kinj_Displays.Display(uk_knobab_kinj_DisplayTable.ACT_TABLE,"tab"))
result = cli.PerformQuery(q)
# Loading the CSV as a Pandas Dataframe
pd.read_csv(StringIO(result.message))
```

- KnoBAB can be queried as a restful HTTP server (`knobab_server`) and has API (**KINJ**) supporting major programming language by transpiling code in Haxe!
- The above example shows how to perform a query and getting answers in Python by connecting to the KnoBAB server.

IX. Minimal Model Description Length

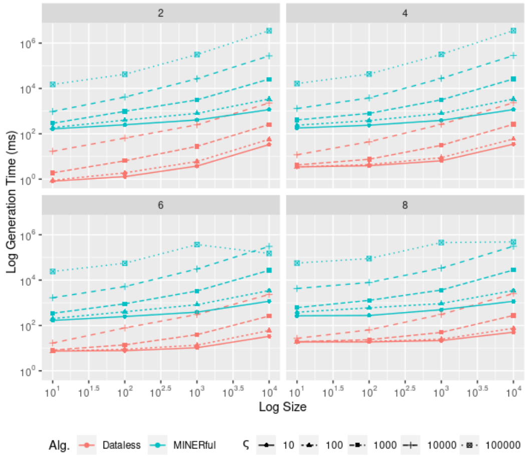


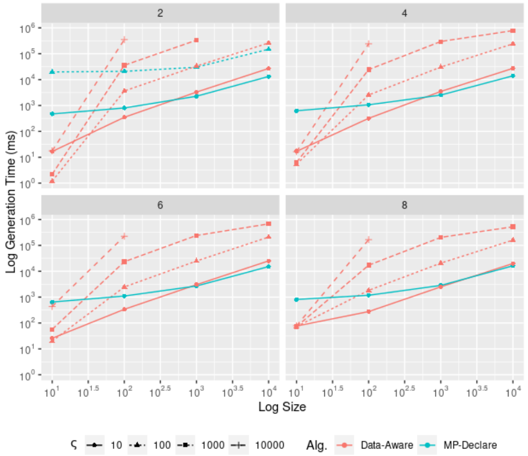
Log Gen. Model, $\mathcal{M}^*$	**Bolt**	**TopN**	**ADM**	**ADM+S**
ChainPrecedence(b, c)	✓	✗	✓(+Choice,CoExist.)	✓(+Choice,CoExist.)
ChainResponse(d, e)	✓	✗	✓(+Choice,CoExist.)	✓(+Choice,CoExist.)
Choice(f, g)	RespExistence(g, f)	✓(+Choice(g, f))	✓(+Choice(g, f))	✓(+Choice(g, f))
ExclChoice(h, a)	✓	✗	✗	✓(+ ExclChoice(a, h))
Exists(b, 1)	✓	✗	✓	✓
Init(e)	✓	✓	✓	✓
Precedence(c, b)	✓	✗	✓(+Choice,RespEx.,CoEx.)	✓(+Choice,RespEx.)
RespExistence(e, f)	Response(e, f)	✓	✓	✓
RespExistence(h, i)	CoExistence(h, i)	✗	✓(+ CoExistence(h, i))	✓
RespExistence(i, h)		✗	✓(+ CoExistence(i, h))	✓
Response(e, f)	✓	✓(+ RespExistence(e, f))	✓(+ RespExistence(e, f))	✓(+ RespExistence(e, f))
$\lvert\mathcal{M}_\theta\rvert$	197	112	548	684
$\lvert\mathcal{M}_{\geq 0.999}\rvert$	46	35	119	127
Mining time (μ)	$1.73\cdot 10^0$ms	$5.17\cdot 10^1$ms	$1.40\cdot 10^2$ms	$3.29\cdot 10^3$ms

- Our latest work shows that it is possible to obtain compact temporal models without additional learning tasks or "pruning scores" by only enforcing non-zero confidence and traversing the lattice of the declarative patterns.
- Our solution also outperforms previous solutions and renditions of previous algorithms on KnoBAB.

- It is possible to use KnoBAB as a library for re-implementing existing algorithms (**ADM**).

Extra: Synthetic Logs via Graph Algorithms [Under Review]





X. Transparency and Result Replicability

- The codebase is associated to test sets run with googletest.
- The papers' datasets are released on the Open Science Framework.
- We exploit GitHub Workflows for guaranteeing that the released code compiles and that all the tests are run satisfactorily.

`https://github.com/datagram-db/knobab`