Ranked Enumeration for MSO on Trees via Knowledge Compilation

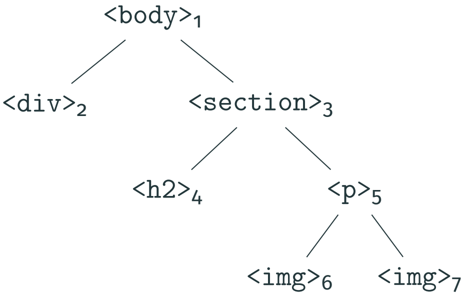
Antoine Amarilli [1], Pierre Bourhis [234], Florent Capelli[25], Mikael Monet[234],

[1]Telecom Paris, [2]CNRS, [3]University of Lille, [4]INRIA Lille, [5]University of Artois

Querying Trees

Trees as Representation of Data

Trees are a classical data structure to represent data into different contexts.



MSO is the classical language to express Boolean queries over trees.
The other classical formalism to express Boolean queries is tree automata.

More Complex Queries over Trees

General MSO queries: MSO with first order free variables returning tuples of nodes

Extension of MSO queries and trees:

- Counting number of solutions
- Queries over probabilistic tree representations [Cohen et al., 2009]
- Enumeration of solutions for an MSO formula with first order variables [Bagan, 2006, Kazana and Segoufin, 2013, Amarilli et al., 2017]

Ordering Results of Queries

In SQL, it is classical to order the results using `ORDER` and to return a limited number of elements with `LIMIT`.

How can we efficiently compute queries with such operators?

This problem was formalized as computing top-k results for ranked conjunctive queries and more generally as enumerating results in a specific order [Deep and Koutris, 2019, Tziavelis et al., 2020, Deep et al., 2022, Tziavelis et al., 2022].

What about queries over trees?

Framework to Evaluate Complex Queries on Trees

Reducing the problem over trees to a problem over **circuits** [Amarilli et al., 2017]:

1. Build a **circuit** representing the solutions $\varphi(T)$
2. Evaluate the problem **through the circuit representation**
3. These circuits fall in **restricted circuit classes** that allow for efficient complex operations

Given a **circuit**, how to efficiently perform ranked enumeration?

Preliminaries

Assignments

Let T be a binary tree. Let $N(T)$ be the nodes of the tree. Let φ be a MSO query and X be the set of first order free variables of φ.

An assignment is a function from X to $N(T)$. The set of assignments is denoted by $\overline{N(T)^X}$

A partial assignment is a partial function from X to $N(T)$. A partial assignment can also be defined as a function from a subset of X to $N(T)$.

Ranking Functions

A ranking function gives a score in S to each partial assignment.

Example of subset monotone ranking function:

- A function ν assigning to each node an integer
- For a partial assignment τ in $\overline{N(T)^Y}$

$$w(\tau) = \sum_{y \in Y} \nu(\tau(y))$$

Subset Monotonicity

Definition (Subset Monotonicity [Tziavelis et al., 2022])

A $(N(T), X)$-ranking function w is subset-monotone if for every $Y \subseteq X$ and partial assignments $\tau_1, \tau_2 \in \overline{N(T)^Y}$ such that $w(\tau_1) \leq w(\tau_2)$, for every partial assignment $\sigma \in \overline{N(T)^{X \setminus Y}}$ (so disjoint with τ_1 and τ_2), we have $w(\sigma \times \tau_1) \leq w(\sigma \times \tau_2)$.

Problem Statement

Let φ be an MSO query with X its first-order variables. Let T be a binary tree and $N(T)$ its nodes. Let w be a $(N(T), X)$ subset monotone ranking function.

The problem $\texttt{RankEnum}(\varphi, T)$ is to enumerate the solutions in $\varphi(T)$ in nonincreasing order given by w.

Enumeration Algorithms

Enumeration algorithms are split in two phases

1. Preprocessing phase: Compute a data structure from φ and T and w.
2. Enumeration phase: Compute the next solution following the nonincreasing order induced by w

We measure the data complexity of an enumeration algorithm:

1. Preprocessing time: complexity of the preprocessing phase
2. Delay: worst case complexity of computing the next solution

We use the RAM model

RAM Model

Representation of data using **logarithmic-sized words**

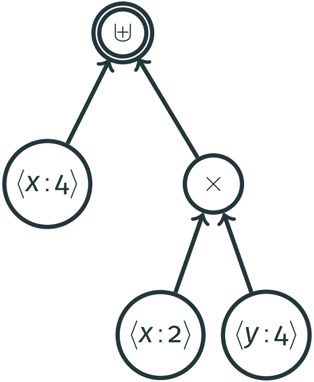
Arithmetic operations take **constant time**

Allocation of arrays in **constant time**

More details in [Grandjean and Jachiet, 2022]

Multi-Valued Circuits

Multi-Valued Circuits



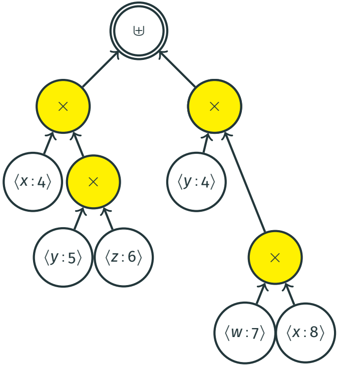
- Directed acyclic graph of **gates**
- **Output** gate: 
- **Value** gates: $\langle x:4 \rangle$,
- **Internal** gates: $\uplus$ $\times$

Circuit Restrictions

DNNF:

- $\times$ are all **decomposable**:

The inputs are **independent**
(= no variable x has a path to two different inputs)



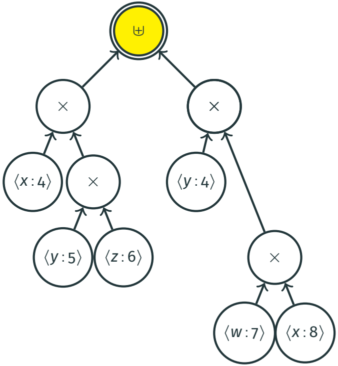
d-DNNF:

- $\times$ are all **decomposable**:

The inputs are **independent**
(= no variable x has a path to two different inputs)

- $\uplus$ are all **deterministic**:

The inputs are **mutually exclusive**
(= no assignments are accepted by both gates)



Smoothed

Smooth Circuit

- $\uplus$ are all **smoothed**:

the valuations defined by the subcircuits are defined on the same set of variables. To smooth, we need to consider all valuations over the missing variables.

Main Results for Querying circuits

Ranked Enumeration for DNNF

Theorem

For any constant $n \in \mathbb{N}$, we can solve the $\mathtt{RankEnum}$ *problem on an input smooth multivalued **DNNF** circuit C on domain D and variables X with $|X| = n$ and a subset-monotone (D, X)-ranking function with **no preprocessing** and with **delay** $O(|D| \times |C| + \log(K+1))$, where K is the number of assignments produced so far.*

Ranked Enumeration for d-DNNF

Theorem

We can solve the $\mathtt{RankEnum}$ problem on an input smooth multivalued d-DNNF circuit C on domain D and variables X with $|X| = n$ and a subset-monotone (D, X)-ranking function with preprocessing linear in $|C|$
and with delay $O(\log(K+1))$, where K is the number of assignments produced so far.

Key Points of the Proof

We do ranked enumeration of partial assignments in C.

We want to skip the possible long paths of $\uplus$ gates.

We associate to each gate g the following structures:

- an integer i_g which counts the number of partial assignments already enumerated
- a priority queue Q_g containing some partial assignments not yet enumerated
- a table T_g containing the assignments already enumerated in nonincreasing order
- a table R_g containing a Boolean indicating if the assignment has already been seen and added to Q_g or not

Preprocessing Phase

1. **Clean** the circuit so that $\times$ gates have exactly two children
2. Compute the **number of partial assignments** defined at each gate
3. **Initialize** the priority queue B_g used for the initialization
4. **Initialize** the priority queue Q_q storing elements of different forms following the type of the gate
5. **Allocate** the tables T_g, R_g and initialize i_g at 0

At the End of the Preprocessing Phase

- $i_g = 0$,
- T_g and R_g are empty
- in Q_g appears
  - for a value gate: $(p : w(\tau),\ d : (g, 1, \tau))$
  - for a $\uplus$-gate: $(p : w(\tau),\ d : (g', 1, \tau))$ where τ is a partial assignment and g' is the descendant gate accepting τ and 1 is the rank of τ when enumerated in g'.
  - for a $\times$-gate: $(p : w(\tau_1 \times \tau_2),\ d : (1, 1, \tau_1, \tau_2))$, where τ_1 is an assignment of the left child g_1 and τ_2 is an assignment of the right child g_2. 1 is the rank of τ_i when enumerated from g_i.

How to Get the Preprocessing in $O(|T|)$

Key points:

- Arithmetic operations in $O(1)$ (RAM Model)
- Initialisation of tables in constant time (RAM Model)
- Persistent priority queue Q with the following properties
  - adding a pair (element, value) in $O(1)$
  - giving an maximum pair (element,value) respecting the nonincreasing order over the values in $O(1)$
  - union of two priority queues in $O(1)$
  - deleting a maximum pair in $O(\log |Q|)$

$\rightarrow$ Brodal Queue [Brodal, 1996]

Enumeration

Implementation of the operator $\texttt{Get}(i, g)$ which returns the i-th partial assignment at the gate g.

- If $i \leq i_g$
  - then use the structure T_g to find it.
  - Otherwise case distinction depending on whether the gate is a $\times$-gate or a $\uplus$-gate

$\texttt{Get}(i_g + 1, g)$ for $\uplus$-gates

- Pop the max element (g', j, τ) of Q_g
- Add τ to $T_g(i_g + 1)$
- $\tau' = \texttt{Get}(g', j + 1)$; Add $(p : w(\tau'), d : (g', j + 1, \tau'))$ in Q_g.
- Increment i_g
- Output τ

$\texttt{Get}(i_g + 1, g)$ for $\times$-gates

Let g_1 and g_2 be the two children of g.

- Pop the max element (j, m, τ_1, τ_2) of Q_g
- Add $\tau_1 \times \tau_2$ to $T_g(i_g + 1)$
- $\tau_1' = \texttt{Get}(g_1, j + 1)$, $\tau_2' = \texttt{Get}(g_2, m + 1)$
    - Check if $\tau_1' \times \tau_2$ or $\tau_1 \times \tau_2'$ were already seen by using R_g
    - If not add them to Q_g and update R_g
- Increment i_g
- Output τ

Complexity Analysis

Number of gates got through the recursive call is linear in the number of variables

At each call of `Get`, the complexity comes from popping the max element which is in $O(\log(|Q_g|))$

Construction of the Circuit Representing the Answers of a MSO Query

Construction of the Circuit Representing $Q(T)$

Theorem

*For any MSO formula φ with capture variables $\alpha_1, \ldots, \alpha_k$, given a **tree** T, we can build in $O(|T| \times |A|)$ a **smoothed multi-valued d-DNNF** capturing exactly the set of tuples $\{\langle \alpha_1 : n_1, \ldots, \alpha_k : n_k \rangle$ in the output of A on T*

This result works via a tree automaton translation of the MSO formula

Proof idea for trees: circuit construction (details)

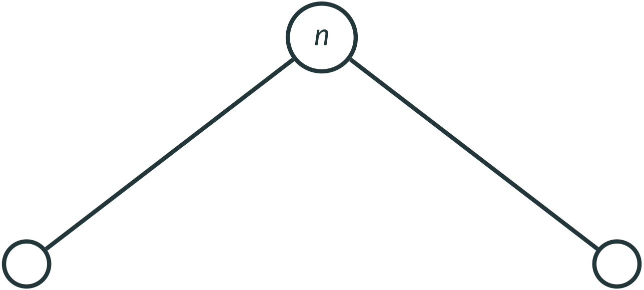
- Automaton: *"Select all node pairs* (α, β)*"*

- States: $\{\emptyset, \alpha, \beta, \alpha\beta\}$
- Rules: $\{\beta, \emptyset \longrightarrow \beta,$
  $\beta, \emptyset, \alpha : n \longrightarrow \alpha\beta$
  $\ldots\}$

Proof idea for trees: circuit construction (details)

- States: $\{\emptyset, \alpha, \beta, \alpha\beta\}$
- Rules: $\{\beta, \emptyset \longrightarrow \beta,$ $\beta, \emptyset, \alpha : \textit{n} \longrightarrow \alpha\beta$ $\dots\}$

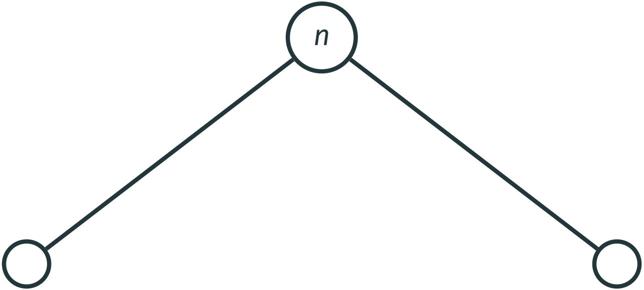
- Automaton: *"Select all node pairs* (α, β)*"*

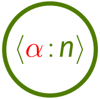


Proof idea for trees: circuit construction (details)

- **Automaton:** *"Select all node pairs* (α, β)*"*

- **States:** $\{\emptyset, \alpha, \beta, \alpha\beta\}$
- **Rules:** $\{\beta, \emptyset \longrightarrow \beta,$
  $\beta, \emptyset, \alpha : n \longrightarrow \alpha\beta$
  $\cdots\}$

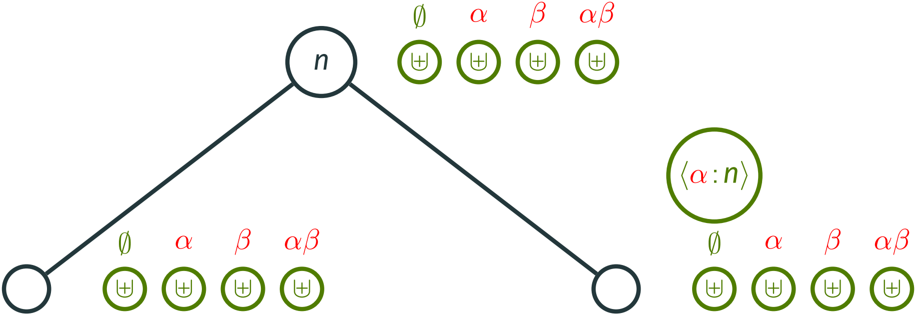




Proof idea for trees: circuit construction (details)

- **Automaton:** *“Select all node pairs* (α, β)*”*

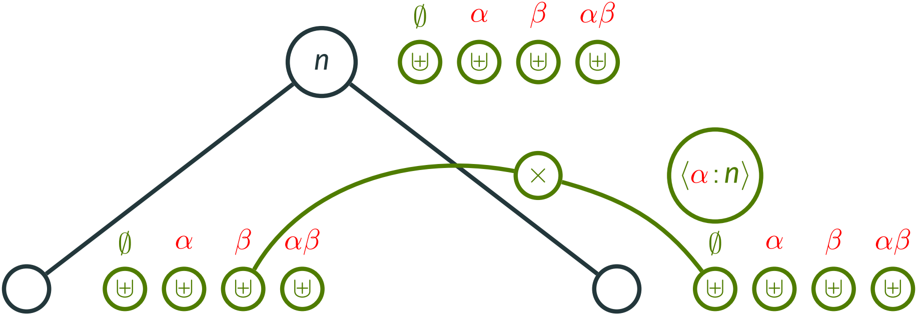
- **States:** $\{\emptyset, \alpha, \beta, \alpha\beta\}$
- **Rules:** $\{\beta, \emptyset \longrightarrow \beta,$
  $\beta, \emptyset, \alpha : n \longrightarrow \alpha\beta$
  $\cdots\}$



Proof idea for trees: circuit construction (details)

- **Automaton:** *"Select all node pairs* (α, β)*"*

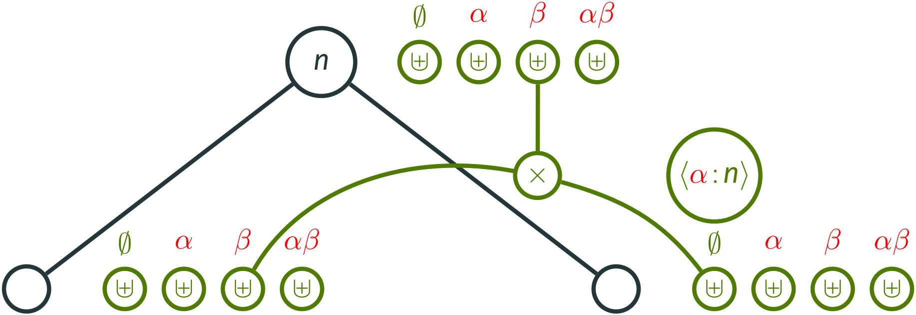
- **States:** $\{\emptyset, \alpha, \beta, \alpha\beta\}$
- **Rules:** $\{\beta, \emptyset \longrightarrow \beta,$
  $\beta, \emptyset, \alpha : n \longrightarrow \alpha\beta$
  $\cdots\}$



Proof idea for trees: circuit construction (details)

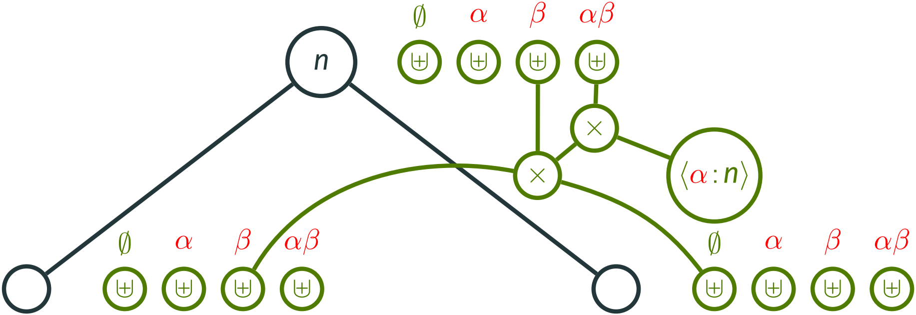
- **Automaton:** *"Select all node pairs* (α, β)*"*

- **States:** $\{\emptyset, \alpha, \beta, \alpha\beta\}$
- **Rules:** $\{\beta, \emptyset \longrightarrow \beta,$
  $\beta, \emptyset, \alpha : n \longrightarrow \alpha\beta$
  $\cdots\}$



Proof idea for trees: circuit construction (details)

- **Automaton:** *"Select all node pairs* (α, β)*"*
- **States:** $\{\emptyset, \alpha, \beta, \alpha\beta\}$
- **Rules:** $\{\beta, \emptyset \longrightarrow \beta,$
  $\beta, \emptyset, \alpha : n \longrightarrow \alpha\beta$
  $\cdots\}$



Theorem

For any fixed MSO query $Q(x_1, \ldots, x_n)$ *with free first-order variables, given as input a tree* T *and a subset-monotone ranking function* w *on the partial assignments of* $x_1, \ldots, x_n$ *to nodes of* T*, we can enumerate the answers to* Q *on* T *in nonincreasing order of scores according to* w *with a preprocessing time of* $O(|T|)$ *and a delay of* $O(\log(K+1))$*, where* K *is the number of answers produced so far enumerated.*

Summary and Future Work

Summary

Established an algorithm for Ranked Enumeration of MSO queries over trees

Approach: uses a circuit representation of the solutions as multivalued smooth d-DNNFs

Ranked Enumeration on d-DNNF circuits can be done with preprocessing in linear time in the size of the circuit and with delay $O(\log(k+1))$ where k is the number of assignments already enumerated.

Future Work

New types of queries to consider from databases:

- Direct Access
- Uniform Sampling
- Generalizing the enumeration of weighted MSO queries on words [Bourhis et al., 2021] to trees
- ...

Incremental Maintenance of the preprocessing part when the tree is updated

Better understanding of the impact of the RAM Model [Grandjean and Jachiet, 2022]

Future Work

New types of queries to consider from databases:

- Direct Access
- Uniform Sampling
- Generalizing the enumeration of weighted MSO queries on words [Bourhis et al., 2021] to trees
- ...

Incremental Maintenance of the preprocessing part when the tree is updated

Better understanding of the impact of the RAM Model [Grandjean and Jachiet, 2022]

Thanks for your attention!

Amarilli, A., Bourhis, P., Jachiet, L., and Mengel, S. (2017).
A circuit-based approach to efficient enumeration.
In *ICALP*.

Bagan, G. (2006).
MSO queries on tree decomposable structures are computable with linear delay.
In *CSL*.

Bourhis, P., Grez, A., Jachiet, L., and Riveros, C. (2021).
Ranked enumeration of MSO logic on words.
In *ICDT*.

Brodal, G. S. (1996).
Worst-case efficient priority queues.
In *SODA*.

Cohen, S., Kimelfeld, B., and Sagiv, Y. (2009).
Running tree automata on probabilistic XML.
In *PODS*.

Deep, S., Hu, X., and Koutris, P. (2022).
Ranked enumeration of join queries with projections.
arXiv preprint arXiv:2201.05566.

Deep, S. and Koutris, P. (2019).
Ranked enumeration of conjunctive query results.
arXiv preprint arXiv:1902.02698.

Grandjean, E. and Jachiet, L. (2022).
Which arithmetic operations can be performed in constant time in the RAM model with addition?
CoRR, abs/2206.13851.

Kazana, W. and Segoufin, L. (2013).
Enumeration of monadic second-order queries on trees.
TOCL.

Tziavelis, N., Ajwani, D., Gatterbauer, W., Riedewald, M., and Yang, X. (2020).
Optimal algorithms for ranked enumeration of answers to full conjunctive queries.
PVLDB, 13(9).

References iv

Tziavelis, N., Gatterbauer, W., and Riedewald, M. (2022).
Any-k algorithms for enumerating ranked answers to conjunctive queries.
arXiv preprint arXiv:2205.05649.