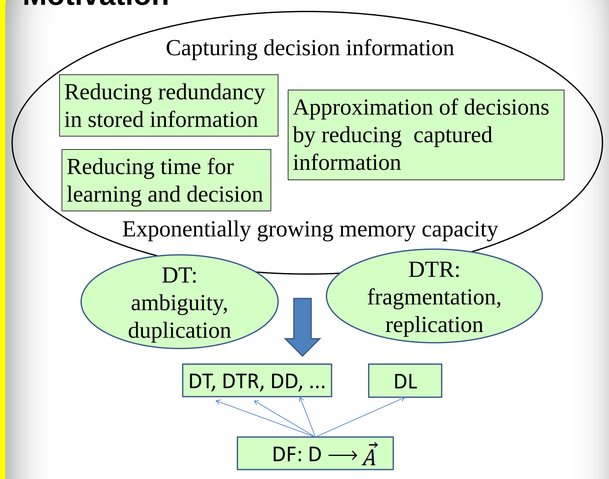
Decision algebras. Capturing and manipulating decision information.

PhD student: Antonina Khairova, advisors: Welf Löwe, Jonas Lundberg **(Linnaeus University, Sweden)**

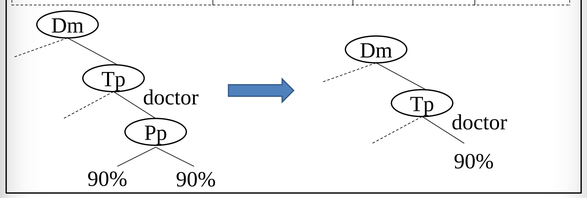
Motivation



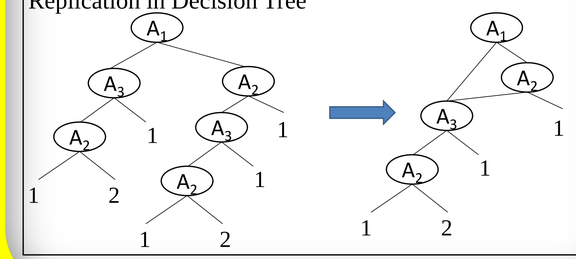
Where DT – decision table, DTR – decision tree, DD – decision diagram, and DL – decision lattice (our data structure for capturing decision information). DF – one common decision function, decision representation D based on unique attribute values vector $\vec{A}$.

Duplication in Decision table

Conditions	1	2	...
Deductible met?	No	No	...
Type of visit	Doctor	Doctor	...
Participation Physician?	Yes	No	
Actions			
...			
Reimburse 90%	X	X	...
...	...	...	...



Replication in Decision Tree



Goals

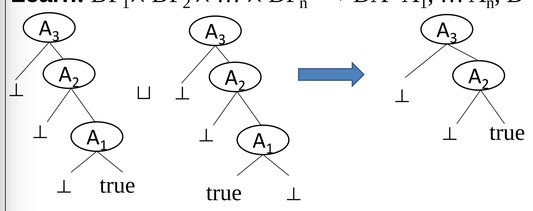
1. Comparing and benchmarking the data structures for capturing decision information.
   In order to provide the right benchmark there is a need for defining the common interface, which we call decision algebra DA. It provides us with the number of standard operations on these data structures.
   DA<A_1, ..., A_n, D >, where
   A_1, ..., A_n – the set of decision attributes;
   D – the set of decisions.
2. Contributing to a classification of an existing technology in a field.

Decision function representation:

Let DF be a set of all decision functions *df* over the decision values D. Each *df* has a unique decision tree representation: $G_t = \{N, E, r\}$, where $N \in$ DF is a set of nodes, E is a set of edges, and $r \in$ DT is the root of the tree. Each *df* is a subgraph of a larger graph which represents the whole decision domain.

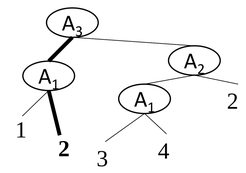
Algebraic laws:

Learn: $DF_1 \times DF_2 \times \ldots \times DF_n \longrightarrow DA<A_1, \ldots A_n, D>$

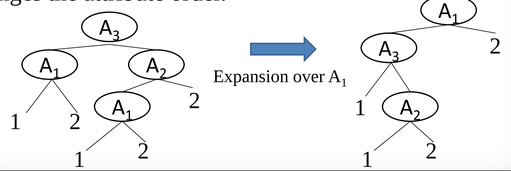


Where × is **Merge** ⊔

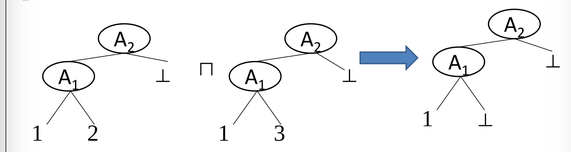
Decide: $DA<A_1, \ldots A_n, D> \times \underline{A}_1 \times \ldots \times \underline{A}_n \longrightarrow D$



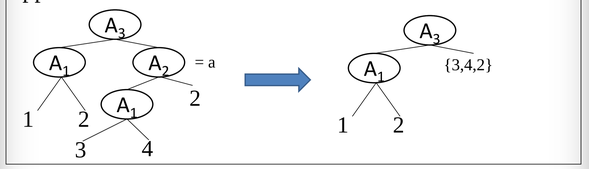
Evert: $DA<A_1, \ldots A_n, D> \times \Pi_n \longrightarrow DA<A'_1, \ldots A'_n, D>$, changes the attribute order.



Apply: $DF_1 \times DF_2 \times .. DF_n \times T \longrightarrow DF$, applies an operation $\tau \in T$ to the terms of the decision functions.



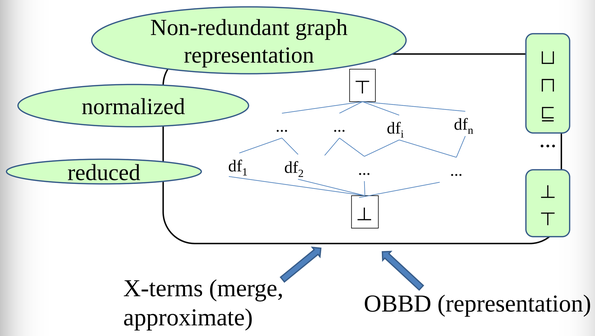
Approximate:
$DA<A_1, \ldots A_n, D> \times \{1...n\} \longrightarrow DA<A'_1, \ldots A'_{n-1}, D>$, approximates the classification information.



The operations merge, learn and approximate can be expressed by a merge operation ⊔ on the decision function DF and equivalence classifications $\equiv_i$ on the attribute algebras A_i. If(D, ⊥, ⊔) induces a semi-lattice, it is guaranteed that approximate defines conservative approximation.

Decision lattice implementation

Since all the operations are defined, we can even provide axioms of decision algebra OP: {⊏, ⊐, ⊥, ⊤, ⊔, ⊓}. The set of *df* form the decision lattice DL = {DF, OP}, where *df*s are partially ordered, ⊤ is the set of all possible decision values, and ⊥ is the empty set ("don't know" situation in classification).

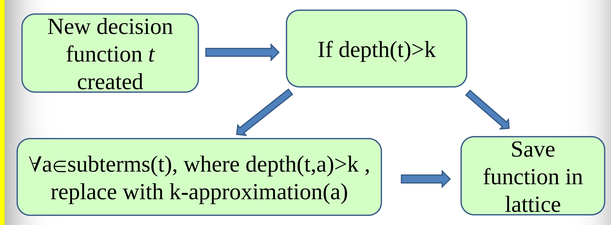


•All decision functions are kept normalized:
Normalized decision function df_b is defined recursively as :
$\forall x^i \in$ descendance(df_b)| i<b,
obeying predefined but changeable normal order of attributes
•All decision functions are kept reduced:
$\forall df_b, df_i \in$ children(t)| $df_b \neq df_i$

•The decision function *t* is not redundant,
$\forall a_i, a_k \in$ subterms(t), if $a_i = a_k$, then $t = a_i$.
• Lattice keeps unique element for each decision functions. The main contribution of using a lattice is memory reduction for keeping decision-sensitive information.`

K-Approximated decision lattice

The size of the decision function grows larger as a learning process proceeds and more and more attributes start to influence decision value D. There is a need to introducing the operation, which can approximate the decision function up-to the defined level *k*.
K-approximation is the operation which keeps track of the subgterms of decision function *df*, where the depth of the subterms should not be larger than *k*, up to this level parts of the tree are collapsed (approximated) into leaf values.
If to include this operation to the decision function creation process, the set of all created functions will form a k-approximated lattice.



Conclusions:

•Replication, duplication : non redundant DL.
•Ambiguity, fragmentation :
  •Reordering ⟶Evert;
  •Partitioning ⟶ Approximation;

Future work:

•More thorough analysis of the technologies to be classified.
•Introduce current learning algorithms and data structures implementation as alternative implementations of DA.

Selected references

1) M. Trapp. *Optimierung Objektorientierter Programme*, Ph.D thesis, Universität Karlsruhe, December 1999.
2) R. E. Bryant, "Symbolic boolean manipulation with ordered binary decision diagrams", *ACM Computing Surveys*, 24(3), September 1992.

Contact information:

PhD student Antonina Khairova
Linnaeus University, Sweden www.lnu.se
e-mail: antonina.khairova@lnu.se