IJS delovno poročilo
IJS DP-14746

2024

Marko Bohanec

Introduction to Decision Modeling Method DEX (Decision EXpert)

Document version 1.2

Jožef Stefan Institute
Ljubljana, Slovenija

DEPARTMENT OF KNOWLEDGE TECHNOLOGIES
Jožef Stefan Institute

CONTENTS

CHAPTER

ONE

INTRODUCTION

DEX (Decision EXpert) is a multi-criteria decision modeling (MCDM) (Greco, et al., 2016) method, aimed at supporting decision makers in complex *decision-making tasks and processes*. DEX was conceived in the 1980s as a fusion of multi-criteria decision analysis and artificial intelligence. From MCDM, it adopted the ideas of modeling decision situations using multiple criteria, structuring and decomposing complex decision problems in smaller and less complex sub-problems, and solving problems through evaluation and analysis of decision alternatives. From artificial intelligence, it primarily adopted concepts used in expert systems: using qualitative (symbolic) variables, representing decision knowledge in terms of "if–then" rules, handling imprecision and uncertainty, emphasizing the transparency of decision models, and facilitating the explanation of results. DEX also includes some elements of rule-based machine learning, e.g., for constructing *compact decision rules* from decision tables.

According to the classification of Ishizaka and Nemery (2013), DEX belongs to the category of *full aggregation* or "American school" methods. This approach is characterized by using an explicit multi-criteria model, which is developed first, more or less independently from individual decision alternatives. These alternatives are then evaluated by the model, first by scoring them for each criterion and then aggregating these evaluations into a global score.

The key characteristics of DEX are (Bohanec, 2022):

- DEX is *hierarchical*: a DEX model consists of *hierarchically structured attributes* (in MCDM, also called criteria or performance variables). In this aspect, DEX is similar to other hierarchical MCDM methods, such as AHP (Saaty, 2012) and MCHP (Corrente, et al., 2012).
- DEX is *qualitative*: attributes in a DEX model are *symbolic*, taking values that are words rather than numbers, such as "bad", "medium", "excellent", "low", or "high". This relates DEX to verbal decision analysis (Moshkovich and Mechitov, 2013), linguistic MCDM (García-Lapresta and del Pozo, 2019), and MCDM methods that use words, such as MACBETH (Bana e Costa, et al., 2003).
- DEX is *rule-based*: hierarchical aggregation of values is defined with *decision rules*, acquired and represented in the form of decision tables. In this way, DEX is most similar to Dominance-Based Rough Set Analysis (DRSA, Greco et al., 2002), which also uses decision tables and constructs decision rules from them.

With extensions, implemented in DEXi Suite software, DEX also became *probabilistic* and *fuzzy*: it allows using probability distributions and fuzzy sets in place of attribute values and evaluates alternatives using probabilistic and fuzzy aggregation. Also, the so called QQ (Qualitative-Quantitative) evaluation was added to rank alternatives within qualitative classes.

Another DEXi Suite extension allows using *numeric attributes* as terminal nodes of a decision model, bringing DEX a little bit closer to *quantitative* MCDM methods.

The DEX method is defined from two aspects: static and dynamic. The static aspect gives a description of concepts and components that constitute a *DEX model*. The *dynamic aspect* addresses algorithms and tools necessary to develop and modify models and to use them for decision support.

1.1 Brief history

The development of DEX can be traced back to Efstathiou and Rajkovič (1979), who proposed using fuzzy sets and fuzzy inference rules to represent and evaluate decision alternatives. The authors also suggested representing decision knowledge in terms of a decision table together with fuzzy operators. The following development of DEX was mainly continued at the Jožef Stefan Institute, Ljubljana, Slovenia, where elements of expert systems and machine learning were gradually added to the basic concepts, leaving the fuzzy aspects somewhat aside. The method, presented by Rajkovič, et al. (1988) and Bohanec and Rajkovič (1988) under the name DECMAK, already had all the main ingredients: tree-structured qualitative attributes, decision tables and decision rules, and algorithms supporting knowledge acquisition and explanation, including a graphical representation of decision tables and a machine-learning algorithm for constructing aggregate rules.

The name DEX (Decision EXpert) was first used in Bohanec and Rajkovič (1990), to denote both the method and the supporting software that was developed at that time. In 2000, the DEX software was replaced by software called DEXi; at that point, the development team decided to keep the name DEX only for the method and use other names for its implementations.

1.2 DEX Software

DEX has always been closely tied with the supporting software. Due to the combinatorial nature of DEX's decision tables, the method is unsuitable for manual construction of models and becomes practical only when supported by appropriate user interfaces and algorithms for knowledge elicitation, representation, verification, and explanation. In many aspects, the definition of the DEX method followed the actual software implementations.

Four generations of DEX-related software have been developed so far:

1. DECMAK (Bohanec and Rajkovič, 1988) was released in 1981 for mini and personal computers under operating systems RT-11, VAX/VMS, and MS-DOS operating systems.
2. DEX was released in 1987 as an integrated interactive computer program for VAX/VMS and MS-DOS.
3. DEXi was released in 2000 for Microsoft Windows. Originally, DEXi was designed as educational software (the letter "i" in DEXi, pronounced "ee", actually comes from the Slovenian "izobraževanje", education). Since 2000, additional features were gradually added to DEXi, which eventually became a complete, stable, and *de-facto* standard implementation of DEX. Until 2021, some additional DEXi software was implemented, forming the DEXi Classic collection of tools.
4. DEXi Suite is a collection of software tools, released in 2023, aimed at gradually replacing DEXi Classic. DEXi Suite brings a new software architecture and some methodological extensions, while remaining backward compatible with DEXi.

CHAPTER

TWO

DECISION MAKING

Decision making is generally defined as a conscious and deliberate selection of one alternative (option, action) from a set of possible ones in order to satisfy the goals of one or more decision makers. Making a decision involves an irrevocable allocation of resources (time, money, effort, . . .), has consequences and is inherently subjective (subject to individual and/or societal values).

2.1 Decision Analysis

Method DEX belongs to the class of multiple-criteria decision analysis methods.

Decision Analysis is a discipline popularly known as *"Applied Decision Theory"*. It provides a framework for analyzing *decision problems* by:

- structuring and breaking them down into more manageable parts,
- explicitly considering the possible alternatives, available information, involved uncertainties, and relevant preferences of the decision maker(s),
- combining these in order to arrive at optimal or at least 'sufficiently good' decisions.

Decision Analysis is particularly interested in *complex decision problems*, that is, problems which are for some reason considered difficult by the decision maker and require careful elaboration and analysis. Complex decision problems are usually characterized by:

- Novelty: the decision maker is confronted with the problem for the first time and has insufficient knowledge or skills to address the problem.
- Uncertainty: existence of possible events that cannot be controlled by the decision maker, but can affect the decision or its consequences (for example: competition response, weather).
- Imprecision: unclear understanding of the problem and its goals, unknown or incompletely defined alternatives, missing data.
- Multiple and possibly conflicting goals.
- Group decision-making: involvement of different decision-makers or groups that have different and possibly conflicting goals.
- Important consequences of the decision (such as possible big financial losses or environmental impacts, lifetime choices).
- Limited resources to conduct the decision process (most often: available time and expertise).

Decision Analysis is aimed at *supporting* people in making decisions rather than *making* decisions themselves. For this purpose, decision-support tools, including DEXiWin, provide methods and tools for developing *decision models* and using them for the *evaluation* and *analysis* of alternatives.

2.1.1 Decision Problem

In *Decision Analysis*, a *decision problem* is understood primarily as a problem of choice or ranking, which is defined as follows:

- Given a set of alternatives, which typically represent some objects or actions, either
- *choose* an alternative that best satisfies the goals (objectives) of the decision maker, or
- *rank* the alternatives accordingly to fulfill these goals.

A somewhat special cases of ranking are *sorting* and *classification*: assigning each alternative to one category among a family of predefined categories. These categories can be preferentially ordered (sorting) or unordered (classification). While method DEX can be used for general choosing and ranking, it is most suitable for sorting and classification tasks.

Making a choice usually occurs as part of a *decision process*.

2.1.2 Decision Process

The ultimate goal of a *decision process* is to solve a *decision problem*, that is, to make a decision. In *Decision Analysis*, the decision process is understood as a process of careful and in-depth analysis of the decision problem. It involves a systematic acquisition and organization of knowledge about the decision problem, which is done by *participants of the decision process* and typically includes:

- assessing the problem,
- collecting and verifying information,
- identifying alternatives,
- anticipating consequences of decisions,
- making the choice using sound and logical judgment based on available information,
- justification and informing others of the decision and its rationale,
- evaluating decisions and their consequences.

In general, such a process should:

- provide all the information needed for a ‘sufficiently good’ decision,
- reduce the chance of overlooking important information and making other errors,
- improve the effectiveness and efficiency of the decision-making, and
- improve the quality of the decision itself.

Usually, the decision process involves at least the following steps:

1. *Problem identification*
2. Modeling: developing a *decision model*
3. *Evaluation* and *analysis* of *alternatives*
4. Choice: making the decision
5. Implementation of the decision

The DEXi Suite software is primarily used in the steps 2 and 3.

2.1.3 Decision Problem Identification

The identification of decision problem occurs at the beginning of a *decision process*. At this stage, the objective is to understand the decision problem and its components. Some typical questions asked in this stage are:

- What is the decision problem about? Is it difficult and important? Why?
- Is this a one-time or recurring decision?
- Who is the stakeholder (decision owner)? Who is responsible, and who will be affected by the decision? Who are other possible *participants* in the decision process?
- What in general are the alternatives in this case? Can we define some specific ones?
- Which goals (objectives) should be achieved by the decision? Which are the criteria to be met by the decision?
- What are the uncertainties involved?
- What are the goals of the decision process? Should we select a single alternative, or evaluate or rank more of them?
- What are the expected consequences of this decision process?
- Do we need to justify the decision? To whom and how?

To be suitable for *multi-criteria modeling*, a decision problem must have some specific properties. Primarily, it should deal with alternatives, which need to be evaluated, analyzed and compared with each other. It is important that the decision problem can be decomposed into smaller, less complex sub-problems, and that the alternatives can be described by their basic features that correspond to the problem decomposition. Thus, we should also ask questions such as:

- Can we think of decomposing the problem into sub-problems? Can we define the relationship between factors that affect the decision?
- Can we think about representing alternatives with their basic features? Which are these features?

2.1.4 Participants in the Decision Process

In general, a typical *decision process* involves up to four types of participants, either individuals or groups:

1. *Stakeholders* (also called *decision problem owners*): individuals or organizations that have a legitimate interest in the decision-making problem. Usually, these are the ones that need to make the final decision, and are also responsible for that decision. They may, but need not, be familiar with the requirements and consequences of the decision problem at hand, and with the possible ways to approach the problem.
2. *Experts*: People knowledgeable in the field so that they can provide information and advice relevant for the decision. They may contribute to the overall *decision problem identification*, to the definition of alternatives, goals and criteria, and to the *decision model* development.
3. *Decision analyst(s)*: Methodologists with experience in *Decision Analysis*, that is, the underlying methodology and tools. Often, they take the role of moderators or mediators of the decision-making team.
4. *Users*: People affected by the decision.

2.1.5 Decision Model

The *Decision Analysis* approach is characterized by the use of decision models. In general, a *decision model* encodes knowledge and information that is relevant for solving the *decision problem* at hand. Decision models are usually developed by *participants* of the *decision process* using tools such as DEXiWin. Typical models used in Decision Analysis are:

- decision trees,
- influence diagrams,
- *multi-criteria models*.

Among these, DEXi Suite employs *qualitative multi-criteria models*.

Once developed, the decision model is used to:

- *evaluate alternatives* and
- perform various *analyses*, such as what-if or sensitivity analysis.

The obtained evaluation and analysis results provide the basis for decision maker's assessment and ranking of alternatives, and possible choice of the best one.

2.1.6 Multi-Criteria Model

Multi-criteria models (also called *multi-attribute models*) represent a class of models used in *Decision Analysis* that evaluate alternatives according to several, possibly conflicting, goals or objectives. In principle, a multi-criteria model represents a decomposition of a decision problem into smaller and less complex sub-problems.

2.1.7 Alternatives

Alternatives are basic entities studied in a *decision problem*. Depending on the problem, they can represent different objects, solutions, courses of action, etc., which are *evaluated* and *analyzed* using a *multi-criteria model*.

2.1.8 Evaluation of Alternatives

With *multi-criteria models*, *alternatives* are *evaluated* in the following way:

1. Each alternative is represented by a vector of basic *attribute values*.
2. The values of each alternative are aggregated in a bottom-up way according to the defined *structure* of the model and corresponding *functions*.
3. The overall evaluation of an alternative is finally obtained as the value of one or more root *attributes* of the model.

On this basis, the decision maker can compare and rank the alternatives, and possibly identify and select the best one.

2.1.9 Analysis of Alternatives

Analysis is one of the key concepts in *Decision Analysis*. In contrast with *evaluation*, which is merely a calculation directed from inputs (*alternative's* descriptions) to outputs (evaluation results), analysis is understood as an active involvement of *participants* who are trying to find answers to questions such as:

- Are evaluations of alternatives in accordance with expectations? Are they 'correct'? If not, why?
- How do the alternatives compare with each other? Which one is the best and why?
- Can we explain and justify the evaluations? What are the most important strong and weak points of individual alternatives?
- What if something changes: What if we try a new alternative? What if an alternative becomes unavailable? What if some characteristics of alternatives change?
- How sensitive is the evaluation to small changes of the model (such as addition or deletion of an attribute, modification of some decision rules)?
- Which properties of an alternative should change so that the overall evaluation becomes better? Which changes of an alternative could substantially worsen is evaluation?

In other words, analysis is a creative and possibly repetitive application of *decision models* aimed at better understanding of the decision problem, better understanding of alternatives, their characteristics and consequences, and better justification of the decision. In general, this involves techniques such as: *what-if* analysis, *sensitivity* analysis, and *stability analysis*.

2.2 DEX Model

Models used in DEXi Suite are prevalently *qualitative*. They are characterized by:

- using qualitative (symbolic) *tree-structured attributes*, whose *scales* are discrete and typically consist of words rather than numbers,
- employing *aggregation functions* that are represented by (tables of) decision rules rather that numerical formulae.

Here, the word "qualitative" is used for contrast with more traditional "quantitative" *decision models*, which are characterized by:

- using continuous numerical attributes, which typically represent the decision-maker's preferences, and
- using numerical aggregation functions, such as the weighted sum.

In DEXi Suite, numerical attributes are supported to a limited extent and typically constitute just a fraction of DEX models. *Numerical attributes* can appear only as input attributes and are immediately mapped to some quantitative attribute using the corresponding *discretization function*.

Generally, a DEX model consists of:

- *attributes*, structured in a *tree*,
- qualitative and continuous *scales* associated with each attribute,
- aggregation and discretization *functions* associated with aggregate attributes,
- *alternatives*.

2.2.1 Attributes

Attributes are variables that occur in *multi-criteria models*. They are organized into a hierarchical structure called *tree of attributes*. According to their position in the tree, the attributes are either:

- *basic attributes*: terminal nodes ("leaves") of the tree, or
- *aggregate attributes*: internal nodes in the tree.

Basic attributes represent inputs of the DEX model. *Alternatives* are described by the values of basic attributes.

Some basic attributes can be *"linked"* in order to structure the model as a full hierarchy rather than a tree.

Aggregate attributes represent *evaluations* of alternatives. They include one or more *roots* of the tree, which represent the overall evaluation of alternatives.

In DEXi Suite, each attribute is defined by its:

- *Name*: main identification of the attribute, which is typically a short string used in printouts, table headings, etc.
- *Description*: usually a longer string providing further documentation about the attribute
- *Scale*

Aggregate attributes also have associated an *aggregation or discretization function*.

Linked Attributes

In principle, DEX models have a strict *tree* structure: *attributes* are structured so that there is exactly one path from each attribute to the root of the tree. This means that each attribute, other than the root, influences exactly one parent in the tree. Sometimes, this is not enough and we wish to introduce attributes that influence more than one parent. In other words, we wish to create attribute *hierarchies* (directed acyclic graphs) rather than ordinary trees.

For this purpose, DEXi Suite uses the concept of *linked attributes*. The idea is that whenever there are two attributes in the tree that have equal names and equal *scales*, and at least one of them is basic, they are declared 'linked' and they - in a logical sense - represent a single attribute. Attribute linking is done automatically by DEXiWin, but only when enabled explicitly in preference settings. By default, linking is disabled and equally named attributes are considered different.

This concept allows that *DEX models* still retain their basic tree structure. In tree displays, linked attributes appear separately, so the tree structure is preserved. However, when defining *alternatives*, linked attributes appear only once and require only a single data entry.

Terminological Remarks

In MCDM, the definitions of terms *goal*, *objective*, *criterion*, *attribute*, *performance variable*, *indicator* vary to some extent depending on the context, author and method. These terms all refer to "something" that needs to be achieved by a proper decision and has to be in some way considered in the corresponding *decision model*. Specifically, the terms *criterion* and *attribute* are sometimes considered equivalent, while some other authors understand *criterion* only as a preferentially ordered attribute.

In DEXi Suite, the term *attribute* is used as a general term for all variables occurring in a *DEX model*. This is because the preference order of an attribute is known only *after* it has been associated with a *scale*; this can occur or change much later than defining the attribute itself. According to some strict definitions, the attribute can be understood as a *criterion* only when assigned an ordered scale. Thus, in general terms, DEX models consist of attributes, and are thus often referred to as *multi-attribute models*.

2.2.2 Tree of Attributes

In a *multi-criteria* model, *attributes* are organized hierarchically into a *tree of attributes*. A model can have one or more *root attributes*. Each attribute can be 'decomposed' into one or more descendant attributes that appear one level below that attribute in the tree. 'Decomposed' attributes are called *aggregate attributes*. Attributes that do not have descendants and appear as leaves of the tree, are called *basic attributes*.

Interpretation

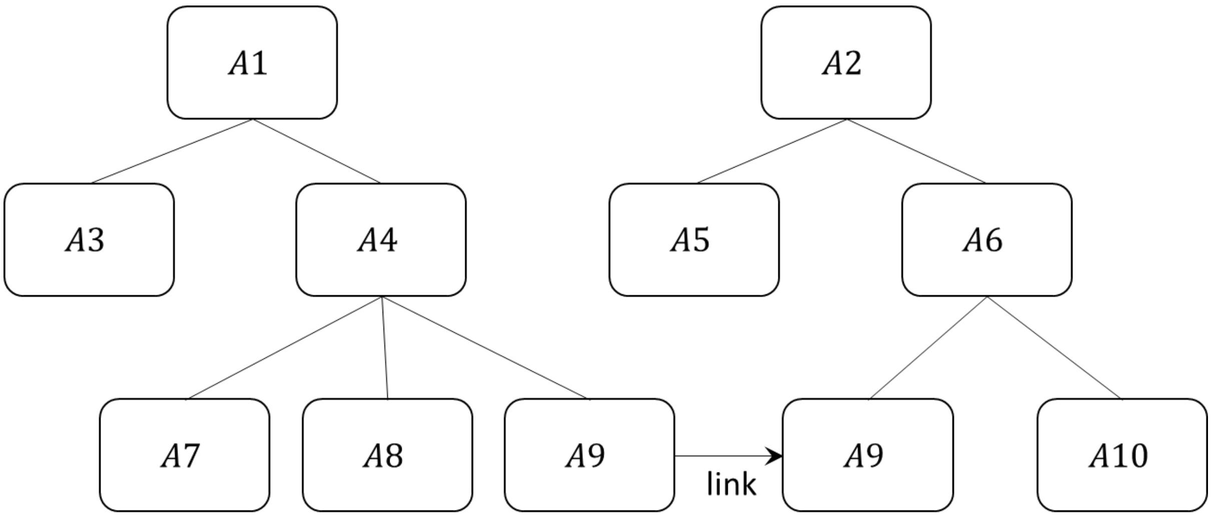
A tree of attributes can be interpreted in three ways:

1. *Decomposition*: It represents a decomposition of a decision problem into sub-problems. To solve 'a problem', which is represented by a a higher-level attribute, one has to solve sub-problems represented by its lower-level descendants.
2. *Dependency*: A higher-level attribute depends on its immediate descendants in the tree. This dependency is modeled by a *function* that is associated with the higher-level attribute.
3. *Aggregation*: Tree structure defines the bottom-up aggregation of alternative values. The value of a higher-level attribute is calculated as an aggregation or discretization of the values of its immediate descendants in the tree. Again, this aggregation is defined by the corresponding *function*.

Attributes can have the following roles:

- *basic attributes* represent inputs of the model,
- *root attributes* represent its main outputs, and
- other *aggregate attributes* represent intermediate results of *alternative evaluation*.

Example



This is an example of a tree of attributes. It consists of 7 basic attributes (A3, A5, A7, A8, A9 (twice), A10) and four aggregate ones (A1, A2, A4, A6). Among the latter, A1 and A2 are two roots. The two basic attributes named A9 illustrate the concept of *linking*: although they appear as two separate attributes in the tree, they have the same name and have been linked together, so that they, in a logical sense, represent a single basic attribute that affects two parent attributes, A4 and A6.

Recommendations

1. Before making a real tree, create an unstructured list of attributes. Brainstorm! At this stage, try not to overlook important attributes, but do not bother about their structure or redundancy. Sketch tree structure on paper.
2. When making a structure of attributes, create meaningful subtrees that contain related attributes. Try structuring your unstructured list in two directions:
   - Bottom-up: Group similar attributes together into a single higher-level attribute. It is usually a good indication if you can find a meaningful name for it.
   - Top-down: Decompose complex attributes into simpler ones.
3. Avoid meaningless, redundant, duplicate, inessential and unoperational attributes. In other words, check each basic attribute that:
   - it has a well defined meaning,
   - it does not duplicate or overlap with some other attributes,
   - it does affect the decision (and you know how, at least approximately),
   - it can be measured or assessed with sufficient accuracy.
4. Avoid aggregate attributes that have too many descendants (usually more than three, but this depends on the size of associated scales, too). Too many descendants cause a *combinatorial explosion* on the size of corresponding *functions*, making them difficult to handle. In this case, try to *restructure* the tree below that attribute.

2.2.3 Scales

Scale defines the set of values that can be assigned to an *attribute*.

In DEXi Suite, there are two types of scales: *qualitative* (or discrete) and *continuous*.

Scales can be *ordered* or *unordered*, and ordered scales can be either *ascending* or *descending*. An unordered scale is just a set of values, whose relation with each other is unknown or undefined. In contrast, the values of an ordered scale are ordered *preferentially*, that is, according to their contribution to the quality of alternatives. The values of ascending scales are ordered from 'bad' to 'good' values, and the value of a descending scales are ordered from 'good' to 'bad' values. In both cases, 'bad' means something that is disadvantageous for the alternative and is not preferred by the decision maker. Analogously, 'good' represents an advantageous and preferred value. The ordering of scales plays an important role in the definition of *aggregation functions*, where it simplifies the definition of decision rules and allows checking of their consistency.

For emphasis and better visualization, 'bad' and 'good' values of ordered scales are printed in different fonts and colors. By default, 'bad' values appear in bold-red and 'good' values appear in italic-green.

Qualitative Scale

A DEX qualitative scale consists of a list of *values* or *categories*. Generally, they are just words, such as 'excellent', 'acceptable', 'inappropriate', etc. Optionally, each may be associated with a textual description.

With preferentially ordered qualitative scales, each category can also be associated with a *class*, which is either 'bad', 'neutral' or 'good', For ascending scales, the default is that the lowest value is considered 'bad' and the highest 'good'. The opposite order holds for descending scales. Value class can be assigned for each category individually, allowing to declare whole subsequences of categories as 'bad' or 'good'.

Example qualitative scales

no, yes
low, medium, high (e.g., for "Quality")
high, medium, low (e.g., for "Price")
unacceptable, acceptable, good, excellent

Continuous Scale

A DEX continuous scale represents all real (floating-point) numbers. Additionally, ordered continuous scales allow a definition of two thresholds, called bad and good threshold, that define value intervals that are considered 'bad' and 'good', respectively, on that scale. Optionally for continuous scales, it is possible to define:

- measuremen unit, (a string of characters), and
- number of decimals for displaying values on that scale (the default is `-1`, meaning as many decimals as needed).

In DEXiWin, continuous scales are represented using angular brackets. The notation `<>` represents a general continuous scale and `<-5; 5>` a scale with defined 'bad' (-5) and 'good' (+5) thresholds.

Continuous scales can only be assigned to basic attributes that are single descendants of some aggregate attribute. This configuration allows defining a *discretization function* and associating it with the aggregate attribute.

Recommendations

On size (number of values) of discrete scales:

- For basic attributes: Use the least number of values that is still sufficient to distinguish between importantly different characteristics of *alternatives*. Usually, this means two to four values.
- For aggregate attributes: The number of values should gradually increase from basic attributes towards the root(s). For example, three four-valued attributes might be aggregated into a five-valued attribute. Five-valued root attributes usually work well.

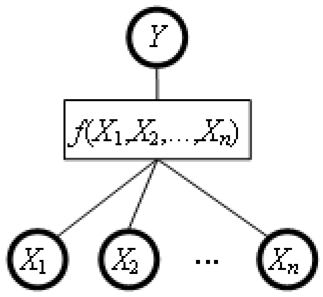
On scale ordering:

- Use ordered scales whenever possible, they really help while defining *functions*.
- Avoid descending scales. They are much less comprehensible than ascending scales. It is particularly confusing when both types are mixed together in a single function.

2.2.4 Functions

Functions are components of *DEX models* that define the aggregation aspect of alternatives' *evaluation*. For each aggregate *attribute* Y, whose descendants in the *tree of attributes* are X_1, X_2,..., X_n, the corresponding function f defines the mapping:

f: $X_1 \times X_2 \times \ldots \times X_n$ —> Y



There are two types of functions used in *DEX models*: *aggregation* and *discretization* functions.

Aggregation Functions

Aggregation functions are the most common function type used in *DEX models*. They are used in cases where both Y (aggregate attribute) and all its descendants X_i are qualitative, i.e., associated with *qualitative scales*.

An aggregation function maps all the *combinations* of the lower-level attribute values into the values of Y. The mapping is represented in a *decision table*, where each row gives the value of f for one combination of the lower-level attribute values. Rows are also called *decision rules*, because each row can be interpreted as an *if-then* rule of the form:

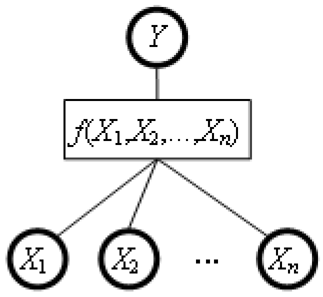
if $X_1 = \text{value}_1$ **and** $X_2 = \text{value}_2$ **and** … **and** $X_n = \text{value}_n$ **then** Y = value (or value *interval*)

The size of decision tables (number of elementary rules) raises quickly with the number of descendants X_i and the number of qualitative values they can take. This is known as a *combinatorial explosion* and should be considered carefully when designing the *tree of attributes*.

Particularly when decision tables are large, it is important to *represent* them in more compact and comprehensible forms. This includes representations using *weights*, which are also important in the *acquisition* of decision tables.

Combinatorial Explosion

Consider an *aggregation function* f that maps the values of the attributes X_1, X_2,…, X_n to values of aggregate attribute Y.



In DEX, an aggregation function maps all the *combinations* of the lower-level attribute values into the values of Y. Suppose that each X_i has a *qualitative scale* consisting of s_i values. Then, the number of combinations and thus the size of the corresponding decision table is equal to

$S = s_1 \times s_2 \times \ldots \times s_n$

Example

Let all the n lower-level attributes have s-valued scales. In this case, the size of f equals to

S $=s^n$

The following table shows how fast S grows with the increasing n and s.

$S = s^n$	Number of lower-level attributes n			
Scale size s	2	3	4	5
2	4	8	16	32
3	9	27	81	243
4	16	64	256	1024
5	25	125	625	3125

Recommendations

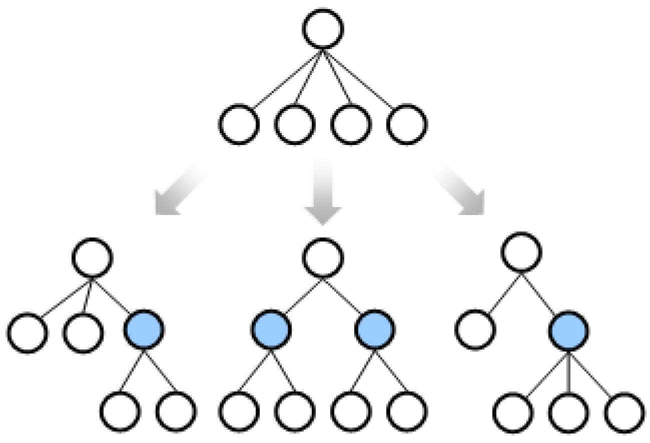
Experience indicates that aggregation functions of size up to 25 are small and usually quite easy to define. The difficulty grows towards the size of about 100, which is already quite difficult. Everything above 100 is very difficult, and everything above 500 is extremely hard if not impossible to define.

Also, it is not only the size that matters. The more the attributes, the more difficult the function to define, even if the size of the functions is comparable. Combining four attributes usually appears harder than combining three.

For these reasons, the DEXi method strongly advises to limit the number of aggregate attributes' descendants to three, and to *restructure* the *tree of attributes* whenever this condition has not been met. Or alternatively, the number of decision rules in a single decision table should be kept below 100.

Restructuring Tree of Attributes

In order to avoid *combinatorial explosion*, it is strongly advised to structure the *tree of attributes* so that each aggregate *attribute* has only two or three immediate descendants. Whenever you encounter an aggregate attribute with four descendants, you may want to consider restructuring the tree below that attribute. Usually, there are several ways to do that:



In all cases, you should regroup the lower-level attributes and introduce one or two new aggregate attributes. Usually, the 'right' structure is the one that appears the most 'logical', so that:

- it groups together similar or related attributes, and
- it is easy to give names to the newly created attributes.

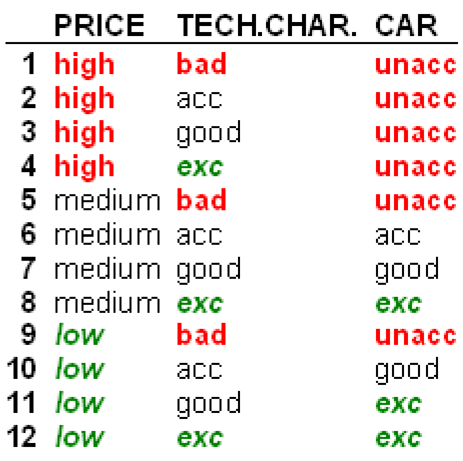
Representing Aggregation Functions

Aggregation functions are sometimes defined with large decision tables that contain 20, 30 or more decision rules. In such cases, it is important to represent such functions in a more compact and comprehensible way. DEX software uses three such representations: using *complex rules*, *decision trees* and *three-dimensional (3D) graphics*.

Complex Rules

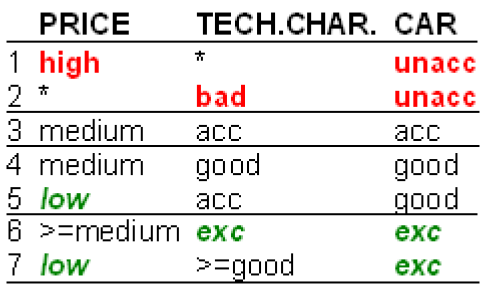
A more compact representation of a decision table is obtained by joining several elementary rules that have the same function value. In other words, a *complex rule* represents one or more elementary rules. A complex rule is characterized by using *intervals* in the conditional part of the rule. The underlying rule-creation algorithm belongs to the class of rule learning algorithms.

The following is the CAR aggregation function from the *Car Evaluation* model, represented with *elementary rules*:



	PRICE	TECH.CHAR.	CAR
1	high	bad	unacc
2	high	acc	unacc
3	high	good	unacc
4	high	exc	unacc
5	medium	bad	unacc
6	medium	acc	acc
7	medium	good	good
8	medium	exc	exc
9	low	bad	unacc
10	low	acc	good
11	low	good	exc
12	low	exc	exc

This is the same function represented with *complex rules*:



	PRICE	TECH.CHAR.	CAR
1	high	*	unacc
2	*	bad	unacc
3	medium	acc	acc
4	medium	good	good
5	low	acc	good
6	>=medium	exc	exc
7	low	>=good	exc

Notice that the number of rows has decreased from 12 to 7 and that symbols '*' and '>=' are used in the conditional part of the second table. For example, the complex rule 1 says that if PRICE is 'high', and regardless on the value of TECH.CHAR. (denoted '*'), the value of CAR is 'unacc'. This complex rule is a compact representation of the first four elementary rules.

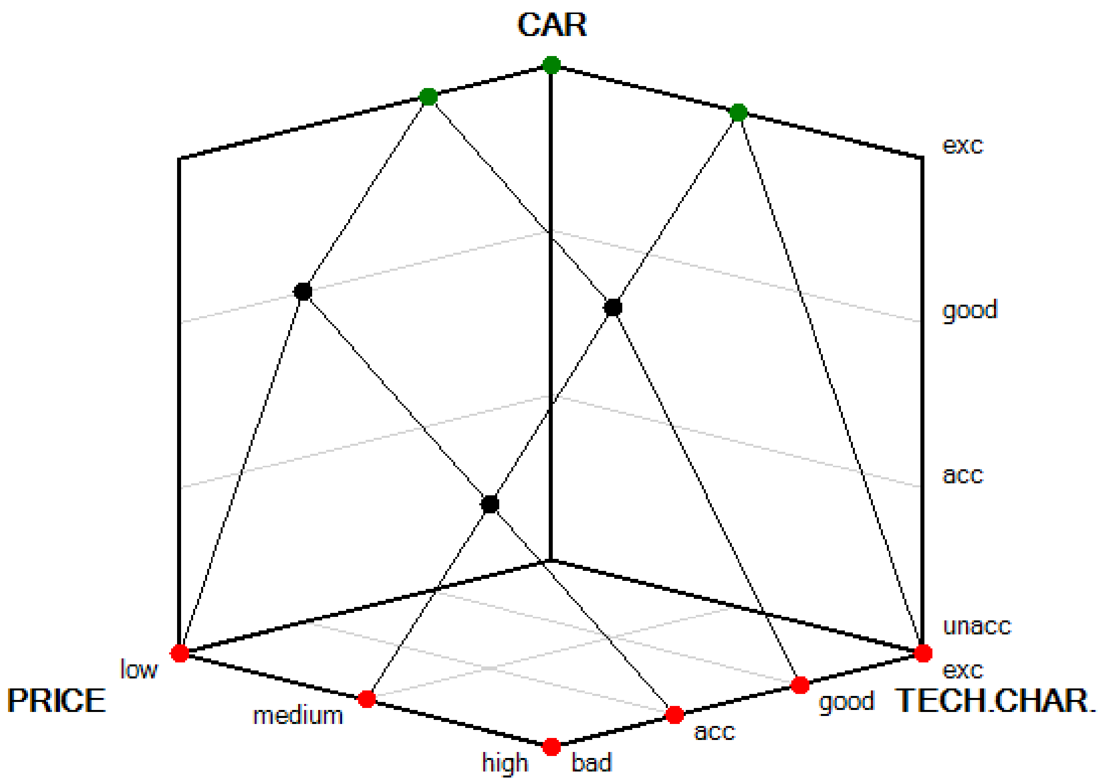
Decision Trees

Alternatively, a decision table can be represented by a decision tree. Following the conditions, starting at the root of the tree (such as 'TECH.CHAR: bad'), decision tree branches eventually lead to terminal nodes that represent the corresponding function value ('CAR: unacc').

```
TECH.CHAR.:
  bad: CAR: unacc
  acc:
   PRICE:
     high: CAR: unacc
     medium: CAR: acc
     low: CAR: good
  good:
   PRICE:
     high: CAR: unacc
     medium: CAR: good
     low: CAR: exc
  exc:
   PRICE:
     high: CAR: unacc
     >=medium: CAR: exc
```

3D Graphics

Another way of representing an aggregation function is using 3D graphics. The same CAR function as above is displayed as follows.



Functions that have more than two input arguments (such as PRICE and TECH.CHAR. above) are displayed as a series of 3D intersections.

Weights

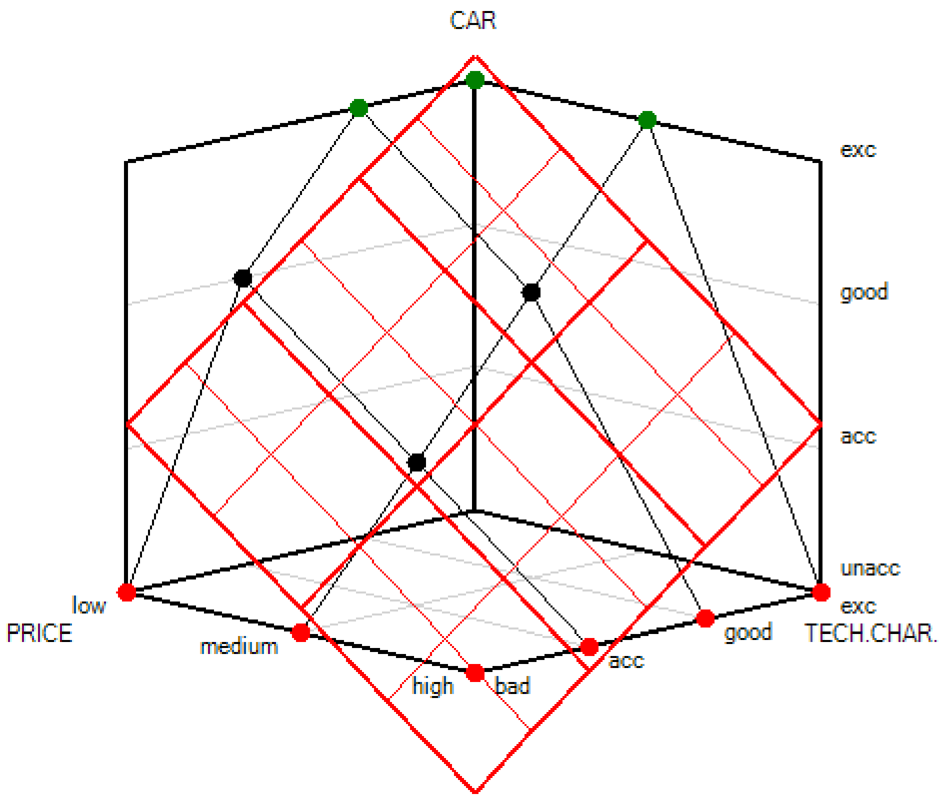
Weights are commonly used in *Decision Analysis* to model the importance of *attributes*. Weights are numbers, usually normalized to the sum or maximum of 1 or 100, which define the relative contribution of the corresponding attribute to the final *evaluation*. In Decision Analysis, *aggregation functions* are commonly defined using some form of the weighted sum, for example:

$f(X_1, X_2, \ldots, X_n) = w_0 + w_1 \times X_1 + w_2 \times X_2 + \ldots + w_n \times X_n$

Here, w_i denote weights and X_i denote attributes.

In *qualitative DEX models*, there is natively no room for weights: attributes are symbolic and aggregation functions are defined by decision rules. However, to bridge the gap between qualitative and quantitative models, it is possible to introduce weights - in a somewhat approximate and imprecise way - also in qualitative models.

Principle



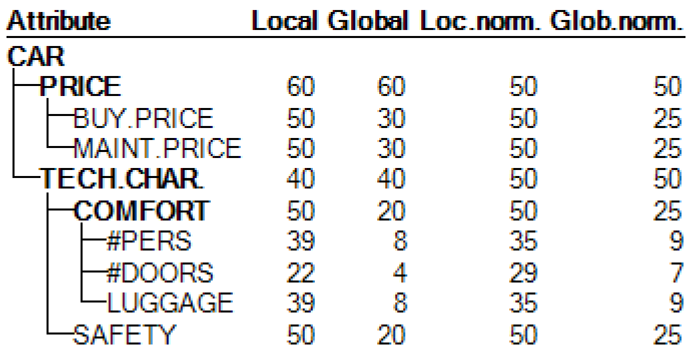
The figure above illustrates the basic approach. It shows the CAR aggregation function as defined in the *Car Evaluation* model, represented by points (dots) in a three-dimensional space. Each point represents one defined decision rule. To find out the weights, we place a (hyper)plane (shown in red) into this space so that it matches the points as closely as possible (using the least squares measure). Once done, weights can be determined directly from the slopes of the hyperplane: the higher the slope in the direction of an attribute, the higher the corresponding relative weight. In the above figure, the weights of PRICE and TECH.CHAR. are both 50. These are *local normalized* weights (see the definition below).

In DEX, weights are used for two purposes:

- as an approximate representation of aggregation functions, used primarily for verification and overview, and
- for defining aggregation functions or their parts (see *using weights*).

Weight Types

Actually, DEX uses four types of weights, as illustrated with the following weights from the *Car Evaluation* model:



Attribute	Local	Global	Loc.norm.	Glob.norm.
CAR				
├PRICE	60	60	50	50
│├BUY.PRICE	50	30	50	25
│└MAINT.PRICE	50	30	50	25
└TECH.CHAR.	40	40	50	50
├COMFORT	50	20	50	25
│├#PERS	39	8	35	9
│├#DOORS	22	4	29	7
│└LUGGAGE	39	8	35	9
└SAFETY	50	20	50	25

The difference between *local* and *global* weights is due to the *tree of attributes*:

- *Local* weights always refer to a single aggregate attribute and a single corresponding aggregation function, so that the sum of weights of the attribute's immediate descendants (function arguments) is 100%.
- *Global* weights, on the other hand, take into account the structure of the tree and relative importance of its sub-trees. A global weight of an attribute is calculated as a product of the local weight and the global weight of the parent attribute. A global weight of the root attribute is 100%. For example: the global normalized weight of BUY.PRICE is 50% (its local normalized weight) × 50% (global normalized weight of PRICE), which gives 25%.

Weights can also be *normalized* or not. This is because some *scales* can have more values than others. Geometrically, larger scales appear longer, they have lower slopes and, consequently, smaller weights. *Normalization* refers to the procedure in which all scales are adjusted to the same length (unit interval) before determining the weights. Usually, this is the better method of weight assessment and comparison of attributes.

Discretization Functions

Discretization functions map *numeric* to *qualitative attribute values*. They can be associated with Y only when Y is an aggregate attribute that has exactly one numeric descendant X, so that f: X —> Y and Y = f(X).

A discretization function is defined as a collection (table) or *discretization rules* of the general form:

interval of X values —> single value or interval of Y

Intervals on the left hand side are defined by their corresponding lower and upper bounds. Considering the whole discretization table, the admissible bounds are constrained so that intervals cover the whole numeric range without overlapping.

2.2.5 Alternatives

Alternatives are basic entities studied in a *decision problem*. Depending on the problem, they can represent different objects, solutions, courses of action, etc., which are *evaluated* and *analyzed* by a *DEX model*.

In DEXi Suite, each alternative is represented by its:

- *name* (not necessarily unique),
- *description*: optional, possibly longer description of the alternative,
- a set of *values*, so that one value is assigned to each *attribute* in the *tree of attributes*.

The set of values is further partitioned into:

- *alternative description*: vector of values assigned to basic attributes;
- *intermediate evaluation results*: values assigned to aggregate attributes other than the roots of the tree;
- *final* (or *overall*) *evaluation results*: values assigned to the root(s) of the tree.

Values assigned to attributes are of different types, collectively referred to as *DEX values*.

2.2.6 DEX Values

DEX values are used throughout *DEX models* to provide input values and carry results of *evaluations* of decision *alternatives*. A value that can be assigned to an *attribute* is always bound to the context provided by that attribute's *scale*.

A special DEX value *undefined* can be assigned to any attribute, including those whose scales are undefined. In principle, any operation (aggregation, discretization) involving *undefined* gives an *undefined* result. However, for qualitative attributes, DEXi Suite allows an *evaluation* mode in which all *undefined* values are expanded to the full *set* of the corresponding attribute's values prior to evaluation. In this way, *undefined* is interpreted as *unknown*.

DEX values assigned to *continuous attributes* can only be real (floating-point) numbers. No intervals or other representations of uncertainty are possible.

DEX value types assigned to *qualitative attributes* are richer and allow expressing uncertainty and imprecision. There are four main qualitative value types: *single value*, *interval*, *set* and *value distribution*. An additional *offset value type* is used in *qualitative-quantitative evaluation*.

For illustrations below, let us take an ordered qualitative scale, consisting of four values: *bad, acc, good, excellent*.

Single Values

A *single* DEX value consists of a single element of the corresponding scale.

Example: `acc`

Interval Values

An *interval value* is defined by two bounds, lower and upper, and consists of all scale elements between these bounds, including the bounds. An interval is usually denoted `lower:upper`.

Example: `acc:excellent`: includes acc, good and excellent

Notational shortcuts are used for preferentially ordered scales:

- `<=acc`: worse than or equal to acc, equivalent to `bad:acc`
- `>=acc`: better than or equal to acc, at least acc, equivalent to `acc:excellent`

Sets

A *value set* consists of a set of elements of the corresponding scale. Sets are usually denoted as lists of ;-separated values. All intervals can be interpreted as sets, while the converse is not always true.

Example: `bad; acc; excellent`

Special notation `*` is used to denote the full set of values.

Example: `*` = `bad:excellent` = `bad; acc; good; excellent`

Value Distributions

A *value distribution* associates each qualitative scale value with some number, generally denoted p and normally expected to be in the [0,1] interval. Depending on the context and used *evaluation method*, p can be interpreted as probability or fuzzy set membership. Distributions are usually displayed in terms of a ;-separated list of pairs `value/p`. $p = 0$ is assumed for elements not appearing in the list.

Example: `bad/0.5; good/0.2; excellent/0.3` (omitted: `acc/0`)

Offset Values

Offset values are used in *qualitative-quantitative evaluation* to facilitate ordering of alternatives within qualitative classes. An *offset value* is composed of a *single qualitative value* and a *numeric offset* on the interval [-0.5, +0.5]. For instance, `acc-0.25` denotes the qualitative value `acc` having the offset `-0.25`. Similarly, `acc+0.11` is the same qualitative value with the offset `+0.11`.

Offset values are meaningful only with ordered value scales. Numeric offsets represent evaluations *within* the associated qualitative values, providing means to preferentially compare and order these values. In this way, `acc-0.25` is considered worse than `acc+0.11`. Decreasing negative offsets denote increasingly worse `acc` values, and increasing positive offsets denote increasingly better `acc` values. The extreme cases are `acc-0.5` and `acc+0.5`.

2.3 Dynamic Aspects of DEX

Dynamic aspects of DEX modeling refer to procedures, algorithms, and tools that are primarily used in two distinct decision analysis stages:

1. *Creation*: Create, edit and maintain a *DEX model*
2. *Use*: use the DEX model to solve the decision problem, i.e., *evaluate* and *analyse* decision alternatives

2.3.1 Creating and Editing DEX Models

In the *DEX-model* creation stage, the main task is to develop an operational model. The main challenges addressed in this stage are how to

- define the model and its components,
- modify, edit, and maintain the model,
- verify the model and its components, e.g., for completeness and consistency,
- deal with uncertainty of knowledge and modeled phenomena, and
- ensure transparency and comprehensibility of the model.

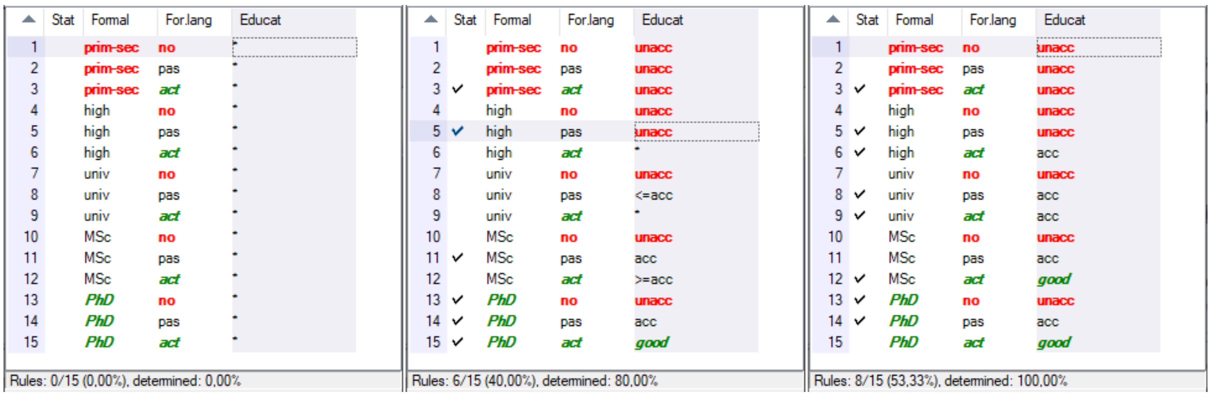
DEX models are typically developed by individual *decision makers or groups*. In most cases, DEX models are developed through *expert modeling*, i.e., "handcrafting" model components and structure, following the approach of expert systems. In this process, DEX models do not only "grow" from the scratch, but are often changed in other ways: attributes are added or deleted, their scales and aggregation functions are defined or changed, attribute hierarchies are restructured, etc. For all such operations, *DEX Software* provides interactive editing software, such as DEXi or DEXiWin.

With regard to making the *model structure*, a DEX editor implements all operations, mentioned above, including model restructuring through drag-drop, duplicate, and copy-paste operations. All these operations are entirely up to the user, who is responsible for the appropriate definition and selection of attributes and their connection in a feasible structure.

However, there are two model creation stages that largely benefit from an active, dynamic support of the software:

- *acquiring decision tables and decision rules*
- *restructuring the model*

Acquiring Decision Tables and Decision Rules



	Stat	Formal	For.lang	Educat
1		prim-sec	no	*
2		prim-sec	pas	*
3		prim-sec	act	*
4		high	no	*
5		high	pas	*
6		high	act	*
7		univ	no	*
8		univ	pas	*
9		univ	act	*
10		MSc	no	*
11		MSc	pas	*
12		MSc	act	*
13		PhD	no	*
14		PhD	pas	*
15		PhD	act	*

Rules: 0/15 (0,00%), determined: 0,00%

	Stat	Formal	For.lang	Educat
1		prim-sec	no	unacc
2		prim-sec	pas	unacc
3	✓	prim-sec	act	unacc
4		high	no	unacc
5	✓	high	pas	unacc
6		high	act	*
7		univ	no	unacc
8		univ	pas	<=acc
9		univ	act	*
10		MSc	no	unacc
11	✓	MSc	pas	acc
12		MSc	act	>=acc
13	✓	PhD	no	unacc
14	✓	PhD	pas	acc
15	✓	PhD	act	good

Rules: 6/15 (40,00%), determined: 80,00%

	Stat	Formal	For.lang	Educat
1		prim-sec	no	unacc
2		prim-sec	pas	unacc
3	✓	prim-sec	act	unacc
4		high	no	unacc
5	✓	high	pas	unacc
6	✓	high	act	acc
7		univ	no	unacc
8	✓	univ	pas	acc
9	✓	univ	act	acc
10		MSc	no	unacc
11		MSc	pas	acc
12	✓	MSc	act	good
13	✓	PhD	no	unacc
14	✓	PhD	pas	acc
15		PhD	act	good

Rules: 8/15 (53,33%), determined: 100,00%

The above figure illustrates typical stages of creating a DEX aggregation function - in this case, function *Educat* from the *Employee Selection* model.

At the beginning, DEX software generates all possible combinations of values of *Formal* and *For. lang.* There are 15 such combinations, referred to as *decision rules*, for which the value of *Educat* should be determined. All the right-hand values are denoted '*', representing the whole value set of *Educat* and effectively indicating that nothing is known about the corresponding input value combinations. The *status* bar below confirms than 0 of 15 rules have been defined and the function has been determined 0%.

While it is in principle possible to proceed sequentially through the 15 rules and define the value of *Educat* for each of them, it is usually better to follow some strategies. The example above illustrates the *use scale orders* strategy: defining just a few key decision rules and leaving the rest to DEX to determine using the principle of dominance. In the middle table, the user has defined 6 rules (marked with '✓'), clarifying the boundary requirements for 'unacc', 'acc' and 'good' *Educat*. With 6 defined rules, the function became 80% determined, leaving some incompletely defined values at rules 6 ('*'), 9 ('*') and 12 ('>=acc'). This incompleteness was resolved by defining two more rules, as shown on the right.

Handling Non-Entered Function Values

So, how does DEX method handle incompletely defined functions and determine function values that have not been entered by the user?

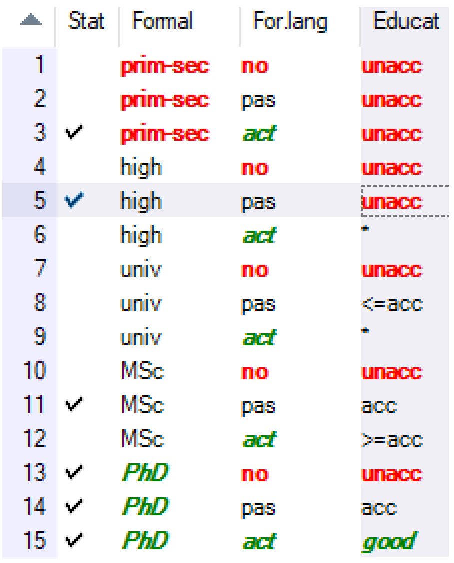
In DEX, function values are either *entered* (marked '✓') by the user or *non-entered*. Entered values are never changed by DEX algorithms. Non-entered values, on the other hand, are handled by DEX with the purpose to aid and simplify the function editing process, and maintain the consistency of function definitions. Non-entered values are recalculated whenever the table changes. The calculation is based on already entered values and other available information (particularly *scale* order, and *weights*).

DEXi uses two strategies for calculating non-entered values, which can be individually activated or deactivated: *Use scale orders* and *Use weights*. Both are available only when function arguments are *preferentially ordered* attributes.

Scale orders

This strategy takes into account the ordering of *scales*, considering the *principle of dominance*. In short, when comparing two decision rules, of which the second has all the left-hand side values equal or better that the first's, the value of the second rule should be equal to or better than the first.

Consider the function y=f(x), where both y and x have ascending scales. Then, whenever x increases, it is clear that f(x) should also increase or remain constant. This function property is known as *monotonicity*. A function is called *consistent* when all transitions between comparable rule pairs are monotone.



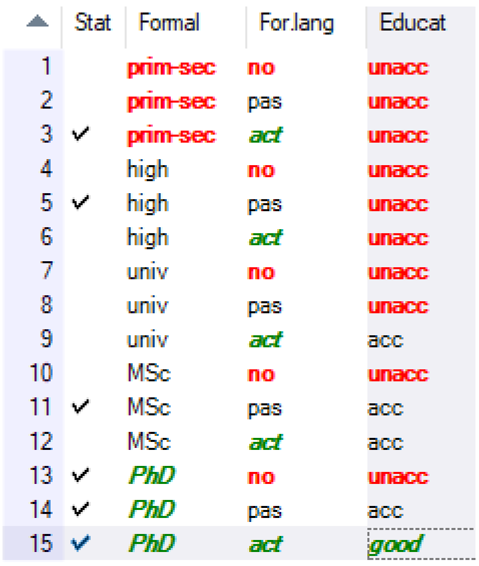
	Stat	Formal	For.lang	Educat
1		prim-sec	no	unacc
2		prim-sec	pas	unacc
3	✓	prim-sec	act	unacc
4		high	no	unacc
5	✓	high	pas	unacc
6		high	act	*
7		univ	no	unacc
8		univ	pas	<=acc
9		univ	act	*
10		MSc	no	unacc
11	✓	MSc	pas	acc
12		MSc	act	>=acc
13	✓	PhD	no	unacc
14	✓	PhD	pas	acc
15	✓	PhD	act	good

It is easy to see that, in general, monotonicity narrows the *intervals* of values that can be assigned to non-entered decision rules. Consider the rule 12 above. Its conditional part is *Formal* = 'MSc' and *For. lang* = 'act'. The lower bound, 'acc', of rule 12 is constrained by rule 11, which has the same input value of *Formal*, but a worse value of *For. lang* = 'pas'. Rule 15, with *Formal* = 'PhD', defines the upper bound 'good'. Consequently, the interval of rule 12 is narrowed to 'acc:good` = '>=acc', as shown in the table.

This strategy fails whenever the user enters values that violate monotonicity and therefore the function becomes *inconsistent*. When the user attempts to enter a violating ("inconsistent") value into a monotone rule set, a DEX editor issues a warning and requests confirmation. In the case that the user confirms such entry, the scale order strategy is deactivated.

Weights

This strategy calculates the values of non-entered rules using a *hyperplane* (linear function), which is constructed using *weights*, defined by the user, and other already entered rules. The hyperplane is constructed so that its slopes correspond to weights required by the user, and that its surface lies as close as possible (in the least squares sense) to the already entered values.



	Stat	Formal	For.lang	Educat
1		prim-sec	no	unacc
2		prim-sec	pas	unacc
3	✓	prim-sec	act	unacc
4		high	no	unacc
5	✓	high	pas	unacc
6		high	act	unacc
7		univ	no	unacc
8		univ	pas	unacc
9		univ	act	acc
10		MSc	no	unacc
11	✓	MSc	pas	acc
12		MSc	act	acc
13	✓	PhD	no	unacc
14	✓	PhD	pas	acc
15	✓	PhD	act	good

Above, the requested weights were 70% for *Formal* and 30% for *For. lang.* Using these weights and the already defined rules 3, 5, 11, 13, 14, and 15, whose values are never changed in this process, the software determined all values of the remaining decision rules.

When using this strategy, at least a few rules have to be entered by the user before this method could construct a hyperplane. The exact number of needed rules depends on function dimensions and geometric positions of entered rules, but until this condition has not been met, the strategy *use weights* is disabled.

Function Status

While editing a function, it is recommended to observe its status. A function is completely *undefined* at first: all function values contain the undefined value '*'. Then, values are assigned to more and more rules and the function becomes more and more defined. Usually, the goal is to completely define the function, that is, to precisely specify values for all decision rules in the table.

Two measures of function definition are used, which are both displayed as status/progress indicators.

The first measure is called *entered rules ratio*. This is a ratio between the number of entered rules (that is, values defined by the user) and the total number of rules.

The second measure, *determination*, is somewhat more complex, but usually more informative. It takes into the account that, in general, rule values are *intervals* rather than single values. A rule is 100% determined if it is assigned a single value, and is 0% determined if it is completely undefined, that is, it contains the complete interval of values (denoted '*'). For smaller intervals, intermediate values are calculated proportionally.

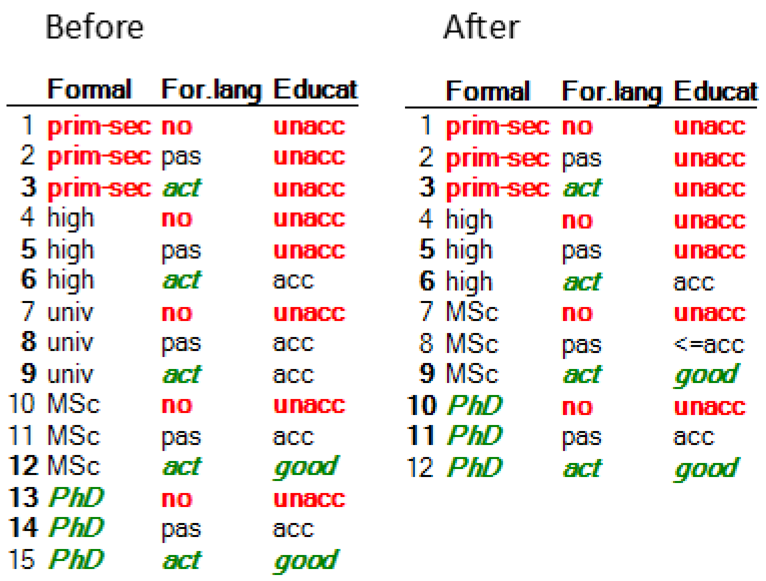
A function is fully determined when all its rules - both entered and non-entered - are 100% determined. This is in general achievable with less than 100% entered rules due to DEX's *handling of non-entered values*. Therefore, when editing a function, the primary aim is to make it 100% determined, regardless on the proportion of entered rules.

Model Restructuring

Model restructuring is an essential dynamic property of DEX that automatically adapts a DEX model to changes and simplifies its maintenance.

At the beginning, when developing model *structure*, there is not much to adapt to; attributes are created, grouped in subtrees, moved around, deleted, etc. However, other model components are added later in the process: *scales*, *functions*, and *alternatives*. After that, any change of model structure or scale definitions may substantially affect associated functions and alternatives.

Here, DEX tries to, whenever possible, *preserve* as much information as possible. This is not always possible, for instance when adding or deleting function arguments, but some scale-editing operations do allow preserving function definition to some extent. Let us show an example of adapting function *Educat* from the *Employee Selection* model to deleting the value ‘univ’ of attribute *Formal*.



Before

	Formal	For.lang	Educat
1	prim-sec	no	unacc
2	prim-sec	pas	unacc
3	prim-sec	act	unacc
4	high	no	unacc
5	high	pas	unacc
6	high	act	acc
7	univ	no	unacc
8	univ	pas	acc
9	univ	act	acc
10	MSc	no	unacc
11	MSc	pas	acc
12	MSc	act	good
13	PhD	no	unacc
14	PhD	pas	acc
15	PhD	act	good

After

	Formal	For.lang	Educat
1	prim-sec	no	unacc
2	prim-sec	pas	unacc
3	prim-sec	act	unacc
4	high	no	unacc
5	high	pas	unacc
6	high	act	acc
7	MSc	no	unacc
8	MSc	pas	<=acc
9	MSc	act	good
10	PhD	no	unacc
11	PhD	pas	acc
12	PhD	act	good

Notice that the majority of decision rules have been preserved, and only rule 8 became somewhat less defined given its new lower (rule 5) and upper (rule 11) bounds. Also note that this change affected the description of alternatives (employee candidates) behind the scene, removing all occurrences of the value ‘univ’.

2.3.2 Using DEX Models

In this stage, one or more *DEX models* are already available and we want to use them to effectively solve the decision problem. This is associated with questions of how to

- obtain and represent data about alternatives,
- handle incomplete or uncertain data about alternatives,
- *evaluate* alternatives, and
- *analyze* alternatives in order to explain, justify, and validate results.

Evaluation of Alternatives

With *DEX models*, *alternatives* are *evaluated* in the following way:

1. Each alternative is represented by a vector of basic *attribute* values.
2. The values of each alternatives are aggregated in a bottom-up way according to the defined *structure* of the model and corresponding *functions*.
3. The overall evaluation of an alternative is finally obtained as the value of one or more root *attributes* of the model.

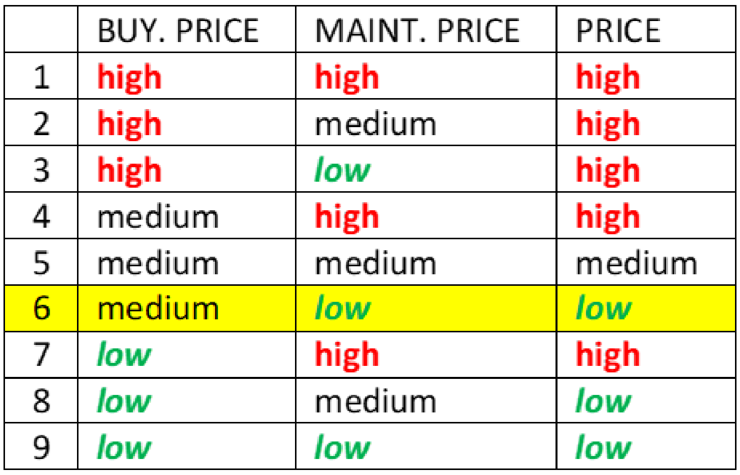
On this basis, the decision maker can compare and rank the alternatives, and possibly identify and select the best one.

A more detailed description of the process depends on the type of *DEX values* used: single values, intervals, sets, or value distributions. Actually, the latter is the most general and covers all cases, but is also the most complex. So let us begin with simple cases. For illustration, we use the PRICE aggregation function from the *Car Evaluation* model.

Before that, it should be noted that *DEX values* include the value *undefined*. In principle, evaluation involving an *undefined* value yields an *undefined* results. In DEX, as an exception, is is possible to declare that any *undefined* value is interpreted as a full *set* of values of the corresponding qualitative attribute. In this case, the *set-based evaluation* takes place.

Single-Value Evaluation

The simplest case occurs when all input values of an alternative are defined and represented by *single values*. In this case, only a simple table lookup is required to find the resulting value.



	BUY. PRICE	MAINT. PRICE	PRICE
1	high	high	high
2	high	medium	high
3	high	*low*	high
4	medium	high	high
5	medium	medium	medium
6	medium	*low*	*low*
7	*low*	high	high
8	*low*	medium	*low*
9	*low*	*low*	*low*

This table shows the evaluation with two single-value inputs: *BUY. PRICE* = 'medium' and *MAINT. PRICE* = 'low'. The table lookup finds the corresponding decision rule 6, which yields the evaluation *PRICE* = 'low'. This single value is used for further evaluation in the tree above *PRICE*.

Interval and Set-Based Evaluation

In this case, input values consist of *intervals* or *sets*. Several lookups in the table are generally required, one for each possible combination of input values.

	BUY. PRICE	MAINT. PRICE	PRICE
1	high	high	high
2	high	medium	high
3	high	*low*	high
4	medium	high	high
5	medium	medium	medium
6	medium	*low*	*low*
7	*low*	high	high
8	*low*	medium	*low*
9	*low*	*low*	*low*

This table shows the chase when *MAINT. PRICE* is represented as an interval consisting of two values, 'medium' and 'low'. Two rules correspond to this situation, 5 and 6. Rule 5 yields 'medium' and rule 6 yields 'low', giving the resulting interval/set *PRICE* = 'medium:low'.

Distribution-Based Evaluation

The most general qualitative evaluation type occurs when input values are represented with *value distributions*. The evaluation considers all possible combinations of discrete input values together with probability/membership numbers p assigned to each input value.

Using Probability Distributions

	BUY. PRICE	MAINT. PRICE	PRICE	Rule prob.
1	high	high	high	
2	high	medium	high	
3	high	*low*	high	
4	medium	high	high	
5	medium	medium	medium	$0.2 \times 0.7 = 0.14$
6	medium	*low*	*low*	$0.2 \times 0.3 = 0.06$
7	*low*	high	high	
8	*low*	medium	*low*	$0.8 \times 0.7 = 0.56$
9	*low*	*low*	*low*	$0.8 \times 0.3 = 0.24$

Consider input values represented with probability distributions: *BUY. PRICE* = 'medium/0.2; low/0.8' and *MAINT. PRICE* = 'medium/0.7, low/0.3'. There are four possible combinations of input values, highlighted in the table. For each combination, its probability is determined as a product of the corresponding p values. Three combinations (rules 6, 8, and 9) yield value 'low', and one combination (rule 5) yields 'medium'. Summing up the corresponding yield probabilities gives the final evaluation: *PRICE* = 'medium/0.14; low/0.86'.

Using Fuzzy Distributions

	BUY. PRICE	MAINT. PRICE	PRICE	Rule memb.
1	high	high	high	
2	high	medium	high	
3	high	*low*	high	
4	medium	high	high	
5	medium	medium	medium	min(0.2,0.7) = 0.2
6	medium	*low*	*low*	min(0.2,0.3) = 0.2
7	*low*	high	high	
8	*low*	medium	*low*	min(0.8,0.7) = 0.7
9	*low*	*low*	*low*	min(0.8,0.3) = 0.8

In fuzzy evaluation, input values are interpreted as fuzzy distributions and the corresponding p values as set memberships. The evaluation proceeds similarly as with probability distributions, except that product and summation operators are replaced with the minimum and maximum, respectively. Cumulative result, determined as a maximum of corresponding value yields, is $PRICE$ = 'medium/0.2; low/0.8'.

Qualitative-Quantitative (QQ) Evaluation

All evaluation methods described above are qualitative and assign decision alternatives to some, usually small, number of qualitative evaluations. In principle, alternatives assigned to the same "bucket" are indistinguishable between each other and cannot be ranked further. This is often undesirable in practice, particularly when the number of alternatives is large and many are in qualitative terms evaluated the same.

	BUY. PRICE	MAINT. PRICE	PRICE
1	high	high	high
2	high	medium	high
3	high	*low*	high
4	medium	high	high
5	medium	medium	medium
6	medium	*low*	*low*
7	*low*	high	high
8	*low*	medium	*low*
9	*low*	*low*	*low*

Looking at the above decision table, one can notice that there are as many as five rules yielding the value $PRICE = high$ (rules 1, 2, 3, 4, 7). Comparing these rules, we can see that some rules correspond to better situations that other rules, despite the same outcome. For instance, an alternative corresponding to rule 3 (where $MAINT.PRICE = low$) is better or at least as good as those corresponding to rule 2 ($MAINT.PRICE = medium$) or rule 1 ($MAINT.PRICE = high$). When comparing those alternatives, the former alternative may be considered better than the latter two and thus ranked higher.

Qualitative-Quantitative (QQ) evaluation method supports such ranking by comparing decision rules in a given decision table and assigning *offsets* to them. Offsets are numbers on the interval [-0.5, +0.5] and reflect the internal ordering of rules (considering dominance) within each output value. The larger the offset, the better the input value combination represented by the rule.

	BUY. PRICE	MAINT. PRICE	PRICE	Ord. value	Offset	Num. value
1	**high**	**high**	**high**	0	–0.25	-0.25
2	**high**	medium	**high**	0	0	0
3	**high**	***low***	**high**	0	+0.25	0.25
4	medium	**high**	**high**	0	0	0
5	medium	medium	medium	1	0	1.00
6	medium	***low***	***low***	2	–0.17	1.83
7	***low***	**high**	**high**	0	+0.25	0.25
8	***low***	medium	***low***	2	–0.17	1.83
9	***low***	***low***	***low***	2	+0.17	2.17

The above table illustrates the principle. The column *Ord.value* shows ordinal values 0, 1, 2 of the corresponding values of *PRICE*: *high*, *medium* and *low*. Offsets, calculated by the method QQ2, are shown in the column *Offset*. The rules 1, 2 and 3, discussed above, are indeed ranked so that 1 is the worst (-0.25) and 3 is the best (+0.25) of them. Rule 4 has the same offset as rule 2 (0), and rule 7 the same as rule 3 (+0.25). Also, rule 9 (with offset +0.17) is better than rule 8 (-0.17).

The column *Num.value* shows the sums of ordinal values and corresponding offsets, giving numerical values that are actually used to rank alternatives.

Analysis

Analysis is one of the key concepts in *Decision Analysis*. In contrast with *evaluation*, which is merely a calculation directed from inputs (data describing alternatives) to outputs (evaluation results), analysis is understood as an active involvement of *participants* who are trying to find answers to questions such as:

- Are alternative evaluations in accordance with expectations? Are they 'correct'? If not, why?
- How do the alternatives compare with each other? Which one is the best and why?
- Can we explain and justify the evaluations? What are the most important strong and weak points of individual alternatives?
- What if something changes: What if we try a new alternative? What if an alternative becomes unavailable? What if some alternative's characteristics change?
- How sensitive is the evaluation to small changes of the model (such as addition or deletion of an attribute, modification of some decision rules)?

In other words, analysis is a creative and possibly repetitive application of *DEX models* aimed at better understanding of the decision problem, better understanding of alternatives, their characteristics and consequences, and better justification of the decision. In general, this involves techniques such as: *what-if* analysis, *sensitivity* analysis, *stability analysis*, etc.

In DEX software, analyses are mostly carried out on the evaluation page, where the user can:

- Review intermediate and overall results of evaluation. In order to clarify and justify the results, they may focus on particularly bad or good evaluations.
- Change individual alternative values and immediately see the effects on evaluation results.
- Perform other analyses, described below.

Typical analysis operations, implemented in DEX software, include:

- *Selective explanation*: Identifying particular advantages and disadvantages of an alternative.
- *Compare Alternatives*: Comparing an alternative with some other alternatives.
- *Plus-Minus Analysis*: Investigating the effects of changing one basic alternative value by one or more steps up and down.

- *Target Analysis*: Investigating the changes of multiple basic alternative values that may improve or degrade evaluation at some selected attribute.

For examples of analyses, see *Car Evaluation* and *Employee Selection*.

For more detailed description of analyses, refer to DEX Software: DEXi and DEXiWin.

CHAPTER

THREE

EXAMPLES

In this documentation, method DEX and supporting software (both DEXi and DEXi Suite) are illustrated and explained using two examples:

- *Car Evaluation*
- *Employee Selection*

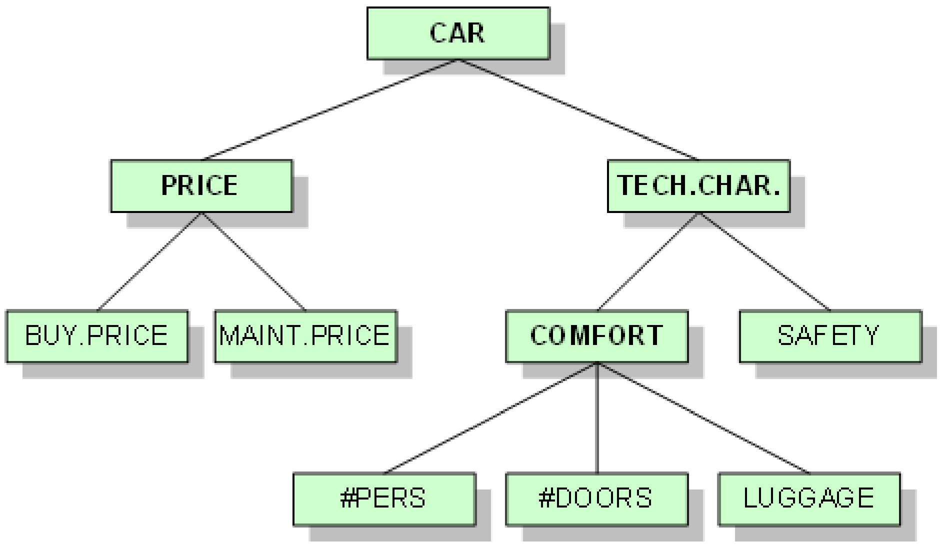
Download example models from: https://dex.ijs.si/dexisuite/download/DEXiExamples.zip.

3.1 Example: Car Evaluation

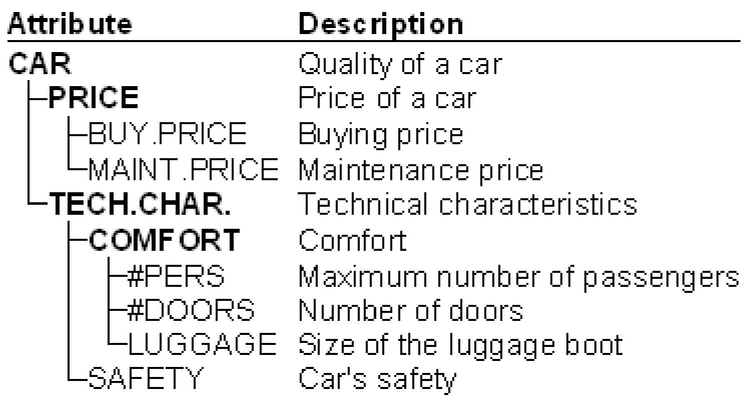
This is an example of a *DEX model* for the evaluation of cars. This is a small and simple model used to illustrate the main concepts of DEX modeling, and is not meant to address the problem of car evaluation at any realistic level. This model has been traditionally handed out together with all versions of DEX software.

3.1.1 Tree of Attributes

The Car Evaluation model has the following *tree structure* of attributes:



The same structure, displayed in a typical DEX way with descriptions of attributes:



Attribute	Description
CAR	Quality of a car
├**PRICE**	Price of a car
│ ├BUY.PRICE	Buying price
│ └MAINT.PRICE	Maintenance price
└**TECH.CHAR.**	Technical characteristics
├**COMFORT**	Comfort
│ ├#PERS	Maximum number of passengers
│ ├#DOORS	Number of doors
│ └LUGGAGE	Size of the luggage boot
└SAFETY	Car's safety

This tree of attributes can be interpreted as follows:

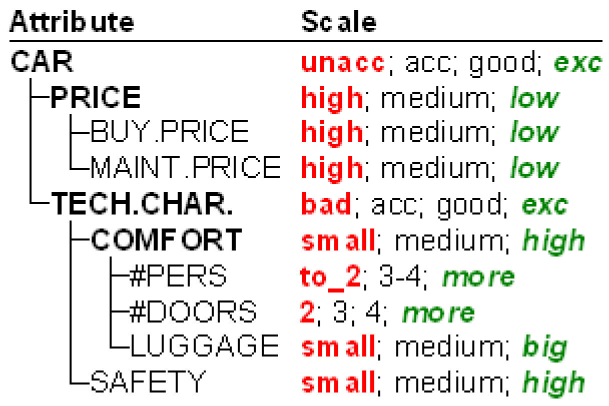
1. *Decomposition*: In order to evaluate a CAR, we consider its PRICE and TECHnical CHARacteristics. PRICE is further decomposed into BUYing PRICE and MAINTenance PRICE. Similarly, TECH.CHAR. are decomposed into COMFORT and SAFETY, and COMFORT is further decomposed into the number of PERSons (passengers), number of DOORS and size of the LUGGAGE boot.
2. *Dependency*: The attribute CAR depends on PRICE and TECH.CHAR. Similarly, COMFORT depends of #PERS, #DOORS and LUGGAGE. Etc.
3. *Aggregation*: The values of #PERS, #DOORS and LUGGAGE are aggregated into a value of COMFORT. Then, in the following order, BUY.PRICE and MAINT.PRICE are aggregated into PRICE, COMFORT and SAFETY are aggregated into TECH.CHAR., and PRICE and TECH.CHAR. are aggregated into CAR.

The attributes in this tree are of the following types:

- Basic attributes are: BUY.PRICE, MAINT.PRICE, #PERS, #DOORS, LUGGAGE and SAFETY
- Aggregate attributes are: CAR, PRICE, TECH.CHAR. and COMFORT.
- The root attribute is CAR.

3.1.2 Scales

The *scales* of attributes are defined as follows:



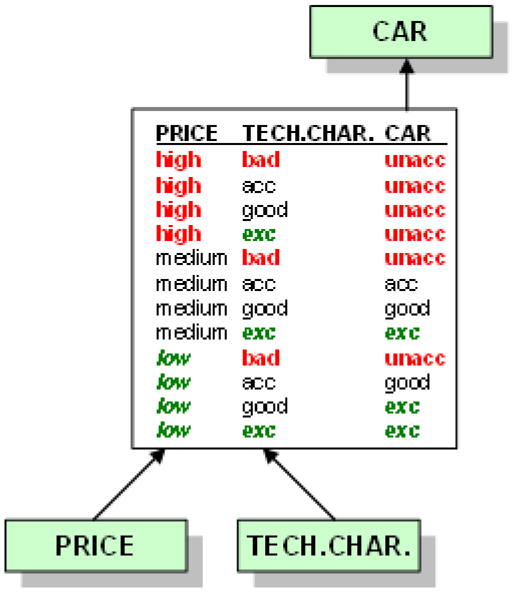
Attribute	Scale
CAR	**unacc**; acc; good; ***exc***
├**PRICE**	**high**; medium; ***low***
│ ├BUY.PRICE	**high**; medium; ***low***
│ └MAINT.PRICE	**high**; medium; ***low***
└**TECH.CHAR.**	**bad**; acc; good; ***exc***
├**COMFORT**	**small**; medium; ***high***
│ ├#PERS	**to_2**; 3-4; ***more***
│ ├#DOORS	**2**; 3; 4; ***more***
│ └LUGGAGE	**small**; medium; ***big***
└SAFETY	**small**; medium; ***high***

Note that all scales are qualitative and preferentially ordered ascendingly, i.e., from bad (red) to good (green) values.

3.1.3 Aggregation Functions

The Car Evaluation model contains only qualitative attributes. Therefore, all functions in the model are *aggregation functions*, represented in the form of decision tables.

Consider the root attribute CAR. According to the tree of attributes, CAR depends on lower-level attributes: PRICE and TECH.CHAR. Thus, the corresponding aggregation function maps all the combinations of values of PRICE and TECH.CHAR. into the values of CAR. The function is defined by the following decision table:

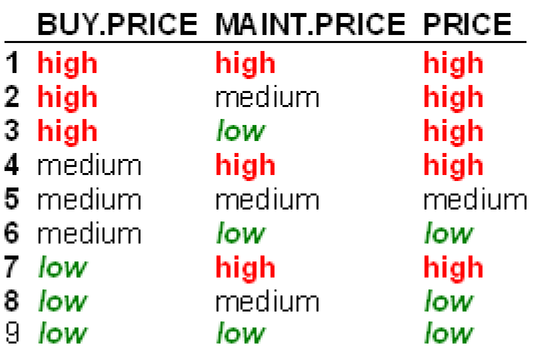


PRICE	TECH.CHAR.	CAR
high	bad	unacc
high	acc	unacc
high	good	unacc
high	exc	unacc
medium	bad	unacc
medium	acc	acc
medium	good	good
medium	exc	exc
low	bad	unacc
low	acc	good
low	good	exc
low	exc	exc

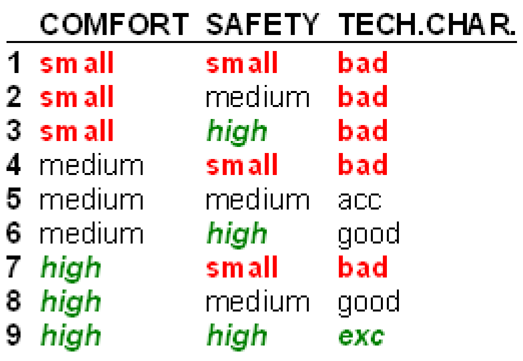
The attributes PRICE and TECH.CHAR. have three and four values, respectively, so the number of rows in the table is 3×4=12. Each row provides a value of CAR for one combination of the values of PRICE and TECH.CHAR. Interpreted as an *elementary decision rule*, the fourth row, for example, means the following:

if PRICE=medium **and** TECH.CHAR.=bad **then** CAR=unacc.

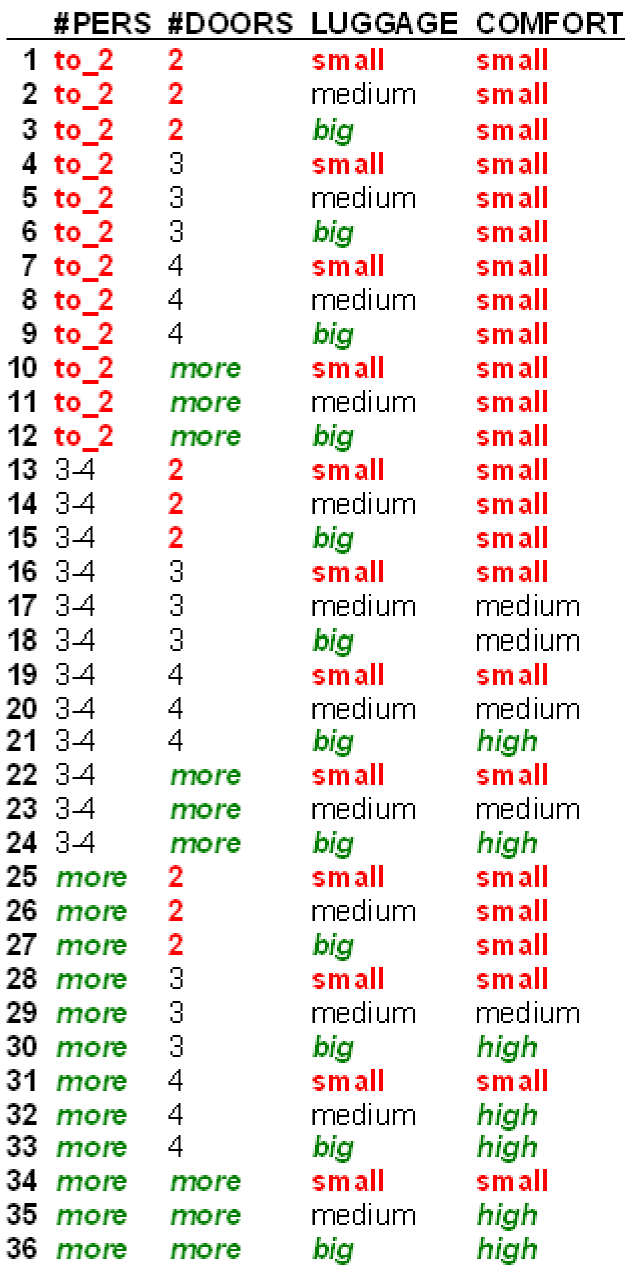
The Car Evaluation model has four aggregate attributes and, consequently, four aggregation functions. The remaining three are:



	BUY.PRICE	MAINT.PRICE	PRICE
1	high	high	high
2	high	medium	high
3	high	low	high
4	medium	high	high
5	medium	medium	medium
6	medium	low	low
7	low	high	high
8	low	medium	low
9	low	low	low



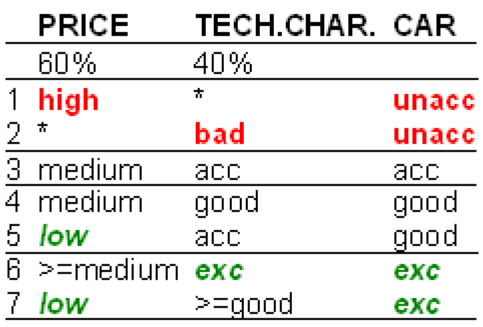
	COMFORT	SAFETY	TECH.CHAR.
1	small	small	bad
2	small	medium	bad
3	small	high	bad
4	medium	small	bad
5	medium	medium	acc
6	medium	high	good
7	high	small	bad
8	high	medium	good
9	high	high	exc



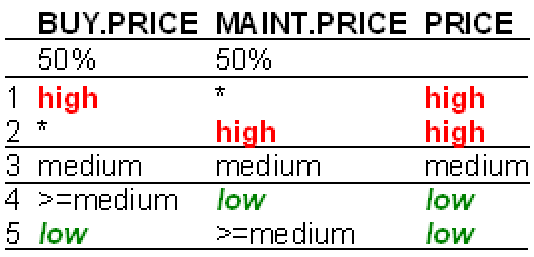
	#PERS	#DOORS	LUGGAGE	COMFORT
1	to_2	2	small	small
2	to_2	2	medium	small
3	to_2	2	big	small
4	to_2	3	small	small
5	to_2	3	medium	small
6	to_2	3	big	small
7	to_2	4	small	small
8	to_2	4	medium	small
9	to_2	4	big	small
10	to_2	more	small	small
11	to_2	more	medium	small
12	to_2	more	big	small
13	3-4	2	small	small
14	3-4	2	medium	small
15	3-4	2	big	small
16	3-4	3	small	small
17	3-4	3	medium	medium
18	3-4	3	big	medium
19	3-4	4	small	small
20	3-4	4	medium	medium
21	3-4	4	big	high
22	3-4	more	small	small
23	3-4	more	medium	medium
24	3-4	more	big	high
25	more	2	small	small
26	more	2	medium	small
27	more	2	big	small
28	more	3	small	small
29	more	3	medium	medium
30	more	3	big	high
31	more	4	small	small
32	more	4	medium	high
33	more	4	big	high
34	more	more	small	small
35	more	more	medium	high
36	more	more	big	high

3.1.4 Complex Rules

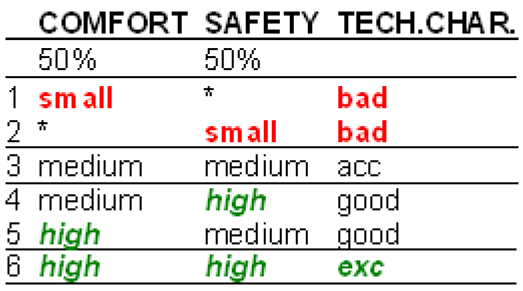
The same functions, represented in terms of local *weights* and *complex rules*, look as follows:



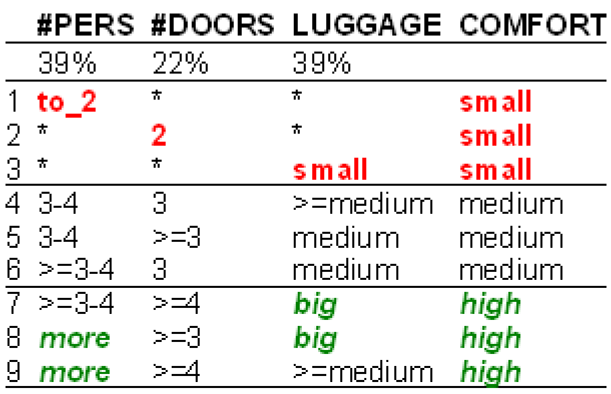
	PRICE	TECH.CHAR.	CAR
	60%	40%	
1	high	*	unacc
2	*	bad	unacc
3	medium	acc	acc
4	medium	good	good
5	low	acc	good
6	>=medium	exc	exc
7	low	>=good	exc



	BUY.PRICE	MAINT.PRICE	PRICE
	50%	50%	
1	high	*	high
2	*	high	high
3	medium	medium	medium
4	>=medium	low	low
5	low	>=medium	low



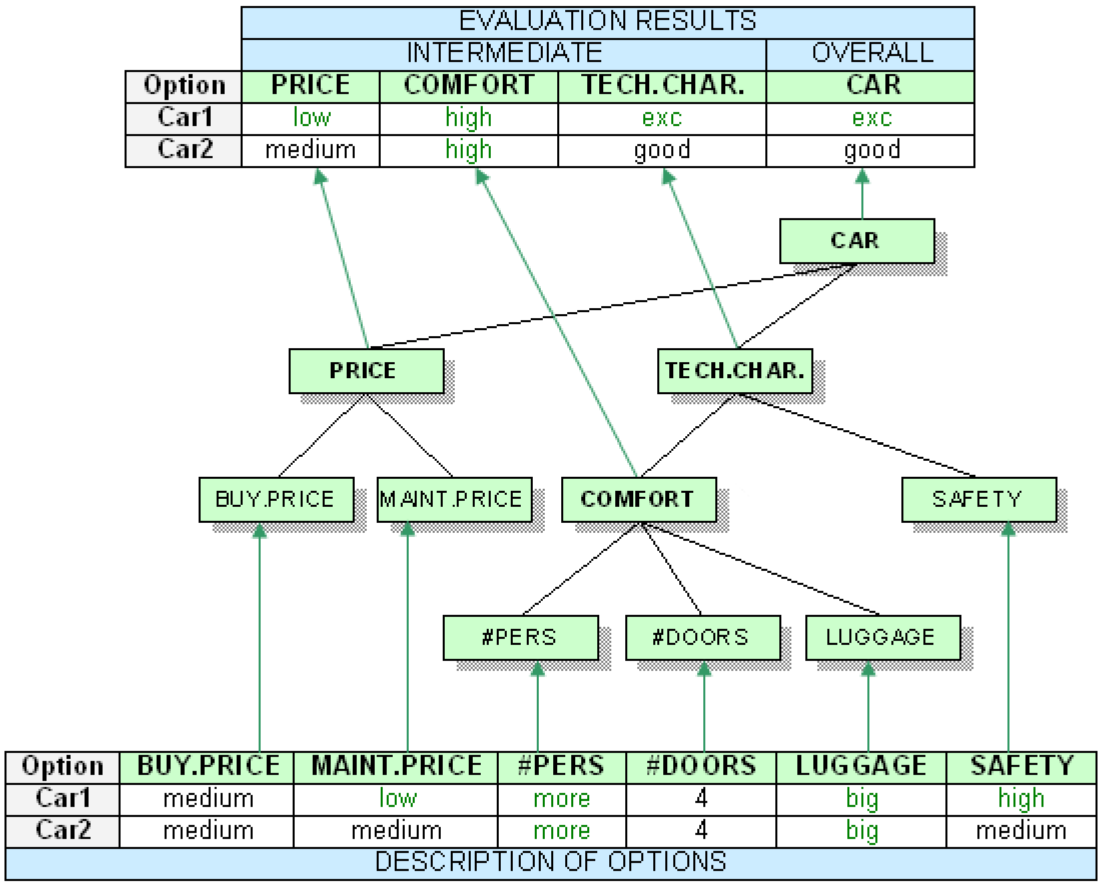
	COMFORT	SAFETY	TECH.CHAR.
	50%	50%	
1	small	*	bad
2	*	small	bad
3	medium	medium	acc
4	medium	high	good
5	high	medium	good
6	high	high	exc



	#PERS	#DOORS	LUGGAGE	COMFORT
	39%	22%	39%	
1	to_2	*	*	small
2	*	2	*	small
3	*	*	small	small
4	3-4	3	>=medium	medium
5	3-4	>=3	medium	medium
6	>=3-4	3	medium	medium
7	>=3-4	>=4	big	high
8	more	>=3	big	high
9	more	>=4	>=medium	high

3.1.5 Description and Evaluation of Cars

Alternatives evaluated by the Car Evaluation model are, obviously, cars. This example illustrates two basic concepts: *description of alternatives* and *evaluation of alternatives*.



	EVALUATION RESULTS			
	INTERMEDIATE			OVERALL
Option	PRICE	COMFORT	TECH.CHAR.	CAR
Car1	low	high	exc	exc
Car2	medium	high	good	good

Option	BUY.PRICE	MAINT.PRICE	#PERS	#DOORS	LUGGAGE	SAFETY
Car1	medium	low	more	4	big	high
Car2	medium	medium	more	4	big	medium
DESCRIPTION OF OPTIONS						

The table at the bottom shows two alternatives, *Car1* and *Car2*, *described* by the qualitative values assigned to the six basic attributes.

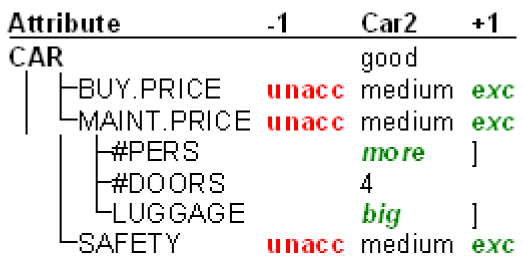
These values are then aggregated from bottom to the top of the *tree of attributes* according to the structure of the tree and defined *aggregation functions*. In this way, *intermediate evaluation results* are first obtained and assigned to the attributes PRICE, COMFORT and TECH.CHAR. (see the table at the top). Finally, the values of PRICE and TECH.CHAR. are aggregated into CAR, giving the *overall evaluation* of both cars.

3.1.6 Some Analyses of Cars

This example shows some analysis reports that can be obtained in DEXi software.

Plus-minus-1 analysis

This example shows results of "Plus-minus-1 analysis" for the alternative *Car2* and the aggregate attribute CAR. The column **Car2** displays the current values of *Car2*. The column **-1** displays the values of the attribute **CAR** when each corresponding lower-level attribute's value changes by one step down (independently of other attributes). Similarly, the column **+1** shows the effects of increasing the value by one step up. Empty fields denote no effect, and the brackets '[' and ']' indicate that the attribute value cannot be decreased or increased, respectively.



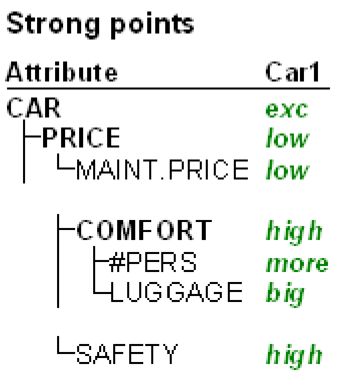
```
Attribute            -1     Car2    +1
CAR                         good
 │ ├BUY.PRICE       unacc  medium  exc
 │ └MAINT.PRICE     unacc  medium  exc
   │ ├#PERS                more     ]
   │ ├#DOORS               4
   │ └LUGGAGE              big      ]
   └SAFETY          unacc  medium  exc
```

The above display shows, for example, that BUY. PRICE considerably affects the evaluation of *Car2*. When BUY. PRICE decreases by one step (from 'medium' to 'high'; the latter value is not shown), the overall value of CAR becomes 'unacc'. In the other direction (from 'medium' to 'low'), the overall evaluation improves to 'exc'.

The two brackets ']' indicate that the values of corresponding attributes, #PERS and LUGGAGE, cannot be increased any more, preventing the **+1** part of the analysis.

Selective explanation

Selective explanation highlights particular advantages and disadvantages of an alternative. The method finds and displays only those connected sub-trees of attributes for which the alternative has been evaluated as particularly good or bad.



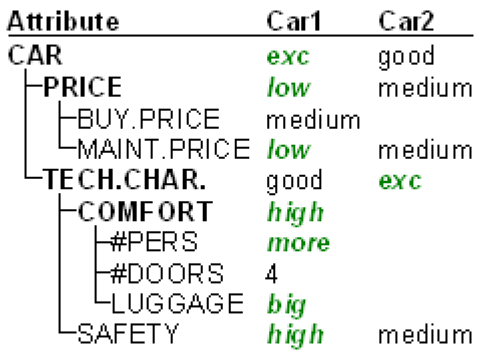
```
Strong points
Attribute          Car1
CAR                exc
 ├PRICE            low
 │ └MAINT.PRICE    low

   ├COMFORT        high
   │ ├#PERS        more
   │ └LUGGAGE      big

   └SAFETY         high
```

This example shows that *Car2* has three particularly strong parts (two sub-trees and one single attribute):

1. overall evaluation, which is strongly influenced by low MAINT. PRICE,
2. COMFORT due to very good #PERS and LUGGAGE, and
3. high SAFETY.

Compare alternatives

This analysis compares one (primary) alternative with one or more other selected (secondary) alternatives, displaying all values of the primary alternative and only those values of the secondary alternatives that differ from the primary's ones.

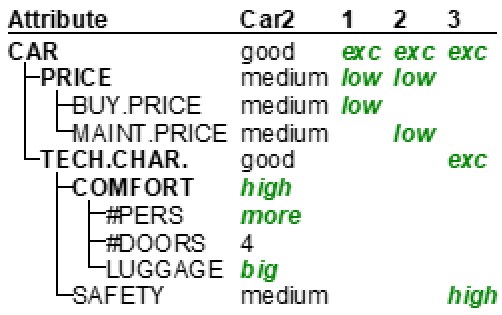


```
Attribute          Car1    Car2
CAR                exc     good
 ├PRICE            low     medium
 │ ├BUY.PRICE      medium
 │ └MAINT.PRICE    low     medium
 └TECH.CHAR.       good    exc
   ├COMFORT        high
   │ ├#PERS        more
   │ ├#DOORS       4
   │ └LUGGAGE      big
   └SAFETY         high    medium
```

This example compares *Car1* (primary alternative) with *Car2* (secondary alternative). *Car2* differs from *Car1* in the values of basic attributes MAINT. PRICE and SAFETY, which cause different evaluations of TECH. CHAR, PRICE and CAR.

Target Analysis

Target analysis (called "Option Generator" in DEXi) tries to find the smallest changes of input values that improve or degrade the value of some selected aggregate attribute. The following are the results of target analysis finding the ways to improve *Car2*'s overall evaluation from good to exc:



Attribute	Car2	1	2	3
CAR	good	*exc*	*exc*	*exc*
├PRICE	medium	*low*	*low*	
│├BUY.PRICE	medium	*low*		
│└MAINT.PRICE	medium		*low*	
└TECH.CHAR.	good			*exc*
├COMFORT	*high*			
│├#PERS	*more*			
│├#DOORS	4			
│└LUGGAGE	*big*			
└SAFETY	medium			*high*

There are three possible ways of improvement:

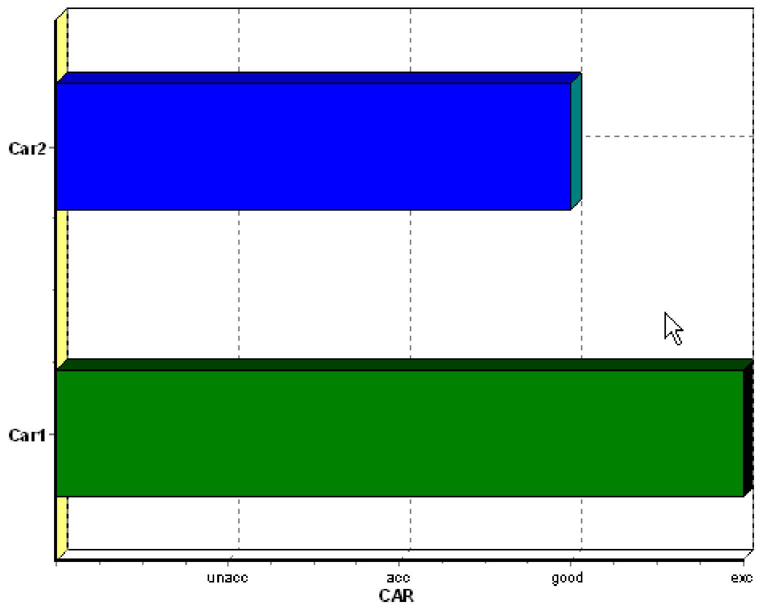
1. lowering BUY.PRICE from medium to low
2. lowering MAINT.PRICE from medium to low
3. improving SAFETY from medium to high

3.1.7 Charts

This example shows some charts that can be obtained in DEXi. The charts differ in the number of evaluation dimensions.

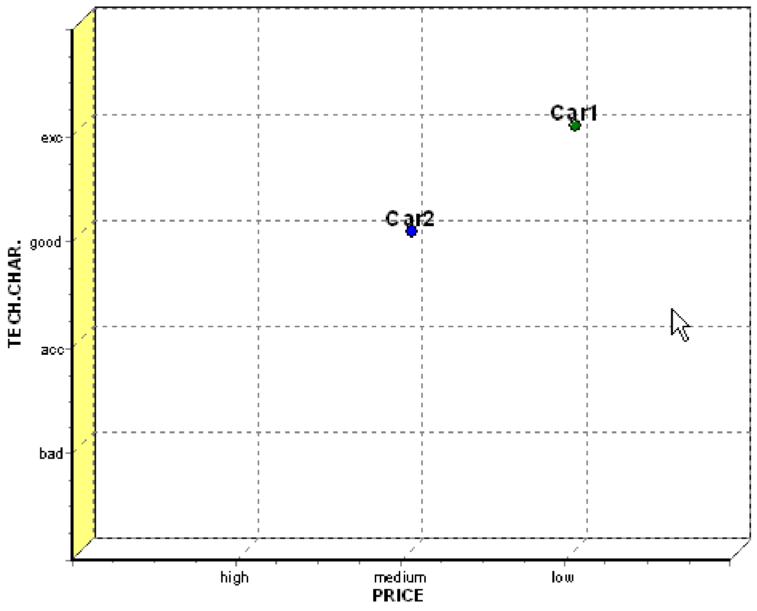
Bar Chart

This chart displays evaluation results according to one evaluation dimension. In this case, this is the root attribute CAR, so the chart shows the overall evaluation of two cars.



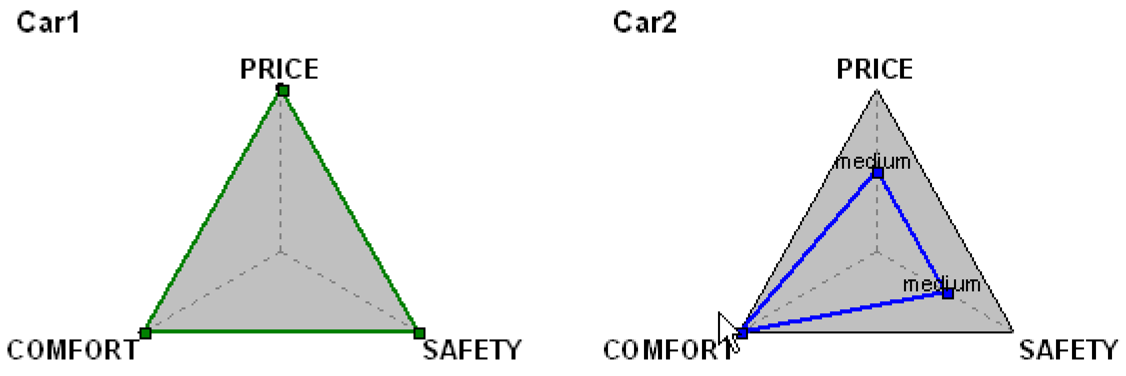
Scatter Chart

A scatter chart displays evaluation results according to two selected evaluation dimensions. In this case, the selected dimensions are PRICE and TECH.CHAR., that is, the two attributes that occur just below the root attribute CAR.



Radar Chart

Radar chart displays evaluation results according to three or more dimensions. The next chart shows the evaluation of cars using the three attributes PRICE, COMFORT and SAFETY.

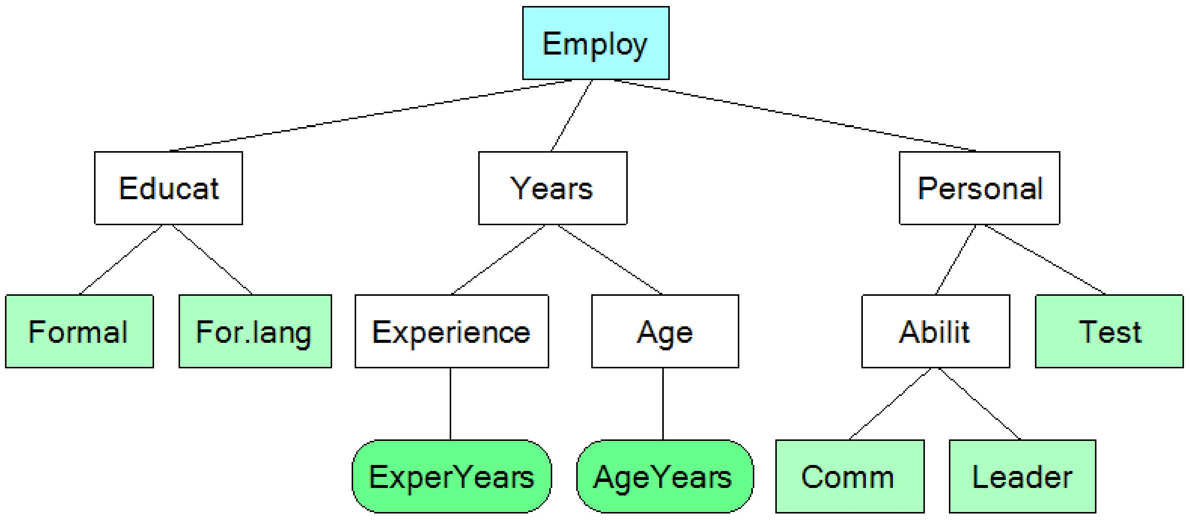


3.2 Example: Employee Selection

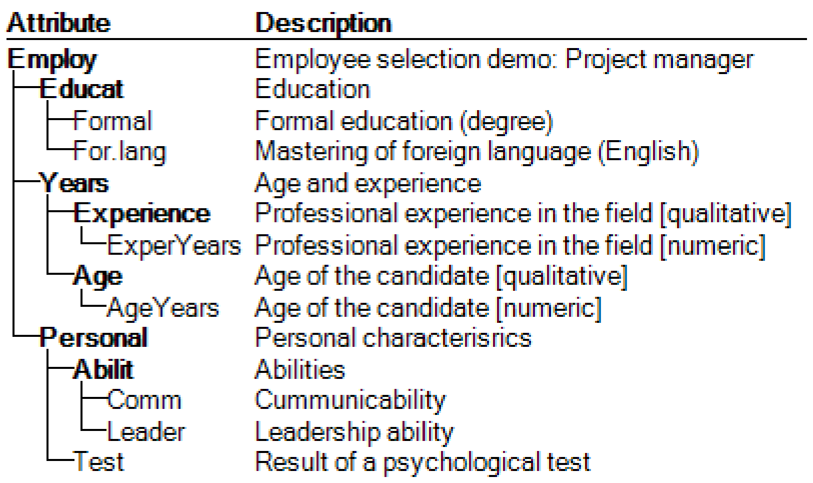
This is an example of a simple didactic *DEX model* aimed at the assessment of applicants for a Project Manager position in a small company. In contrast with the *Car Evaluation example*, this particular model contains two continuous attributes and two discretization functions, and can be thus fully utilized only in DEXi Suite. All figures and charts on this page have been generated using DEXiWin.

3.2.1 Tree of Attributes

The Employee Selection model has the following *tree structure* of attributes:



The same structure, displayed with descriptions of attributes:

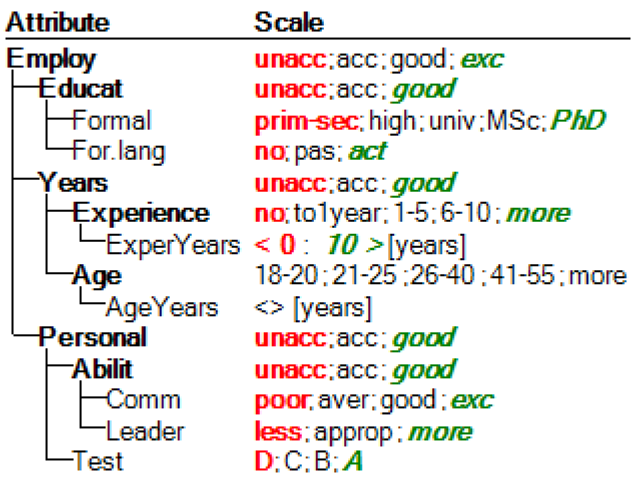


Attribute	Description
Employ	Employee selection demo: Project manager
Educat	Education
Formal	Formal education (degree)
For.lang	Mastering of foreign language (English)
Years	Age and experience
Experience	Professional experience in the field [qualitative]
ExperYears	Professional experience in the field [numeric]
Age	Age of the candidate [qualitative]
AgeYears	Age of the candidate [numeric]
Personal	Personal characterisrics
Abilit	Abilities
Comm	Cummunicability
Leader	Leadership ability
Test	Result of a psychological test

Notice that attributes *ExperYears* and *AgeYears* are *continuous*. In the model, they are *discretized* and mapped to their respective parents, *Experience* and *Age*.

3.2.2 Scales

The *scales* of attributes are defined as follows:



Attribute	Scale
Employ	unacc; acc; good; *exc*
├Educat	unacc; acc; *good*
│├Formal	prim-sec; high; univ; MSc; *PhD*
│└For.lang	no; pas; *act*
├Years	unacc; acc; *good*
│├Experience	no; to1year; 1-5; 6-10; *more*
││└ExperYears	< 0 ; *10* >[years]
│└Age	18-20; 21-25; 26-40; 41-55; more
│ └AgeYears	<> [years]
└Personal	unacc; acc; *good*
├Abilit	unacc; acc; *good*
│├Comm	poor; aver; good; *exc*
│└Leader	less; approp; *more*
└Test	D; C; B; *A*

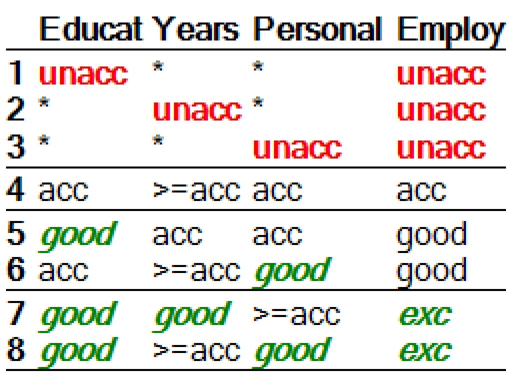
Apart from the two continous scales, *ExperYears* and *AgeYears*, all the remaining scales are *qualitative*. Among these, *Age* is unordered, and all the others are preferentially ordered in the ascending order.

Most of the qualitative values are represented by words: 'unacc', 'high', etc. Even though some values, for instance '1–5' and '21–25', are formulated as numeric intervals, they still represent single qualitative symbols.

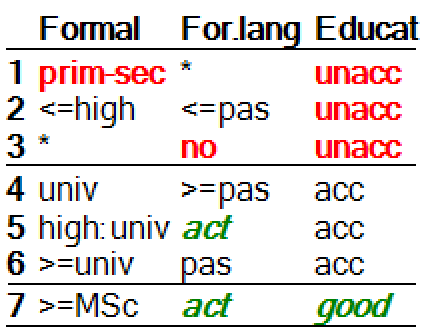
The symbol '<>' denotes an unordered continuous scale of *AgeYears*. The continuous scale of *ExperYears*, denoted '<0; 10>', is preferentially ordered and has two bounds: 0 and 10. All values below and including 0 are considered preferentially 'bad', and all values above and including 10 are considered 'good'.

3.2.3 Aggregation Functions

The Employee Selection model contains seven qualitative attributes: *Employ*, *Educat*, *Years*, *Personal*, *Abilit*, *Experience*, and *Age*. The former five of them are associated with *aggregation functions*. For brevity, only two of those are shown below in the form of *complex* decision rules.



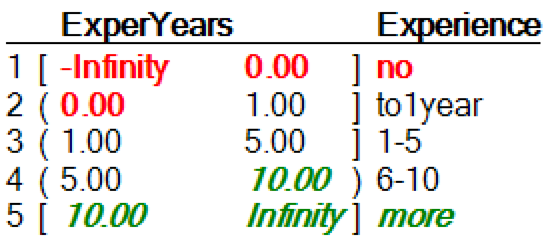
	Educat	Years	Personal	Employ
1	unacc	*	*	unacc
2	*	unacc	*	unacc
3	*	*	unacc	unacc
4	acc	>=acc	acc	acc
5	*good*	acc	acc	good
6	acc	>=acc	*good*	good
7	*good*	*good*	>=acc	*exc*
8	*good*	>=acc	*good*	*exc*



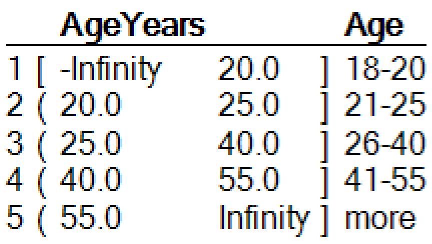
	Formal	For.lang	Educat
1	prim-sec	*	unacc
2	<=high	<=pas	unacc
3	*	no	unacc
4	univ	>=pas	acc
5	high:univ	*act*	acc
6	>=univ	pas	acc
7	>=MSc	*act*	*good*

3.2.4 Discretization Functions

Discretization functions map numeric values of continuous input attributes to the qualitative values of the corresponding parent attributes. The two discretization function in the Employee Selection model are associated with *Experience* and *Age*, and are defined as follows:



	ExperYears			Experience
1	[-Infinity	0.00	]	no
2	(0.00	1.00	]	to1year
3	(1.00	5.00	]	1-5
4	(5.00	10.00	)	6-10
5	[10.00	Infinity	]	more



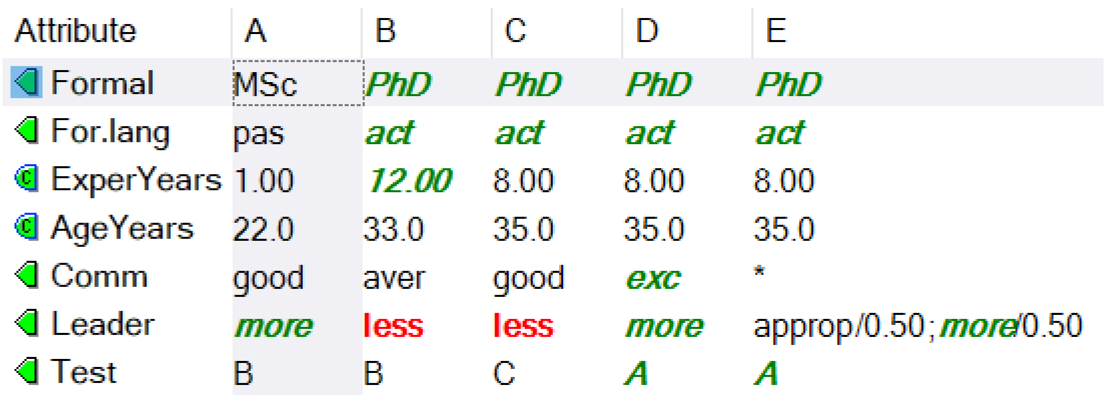
	AgeYears			Age
1	[-Infinity	20.0	]	18-20
2	(20.0	25.0	]	21-25
3	(25.0	40.0	]	26-40
4	(40.0	55.0	]	41-55
5	(55.0	Infinity	]	more

Notice the symbols '[', ']', '(' and ')', associated with intervals:

- '[' and ']' denote the closed interval, so that the corresponding value belongs to the interval
- '(' and ')' denote the open interval, so that the associated value does not belong to that interval, but to the neighboring one

3.2.5 Description of Employees

Alternatives - employee candidates - are defined by values of input (basic) attributes as follows:



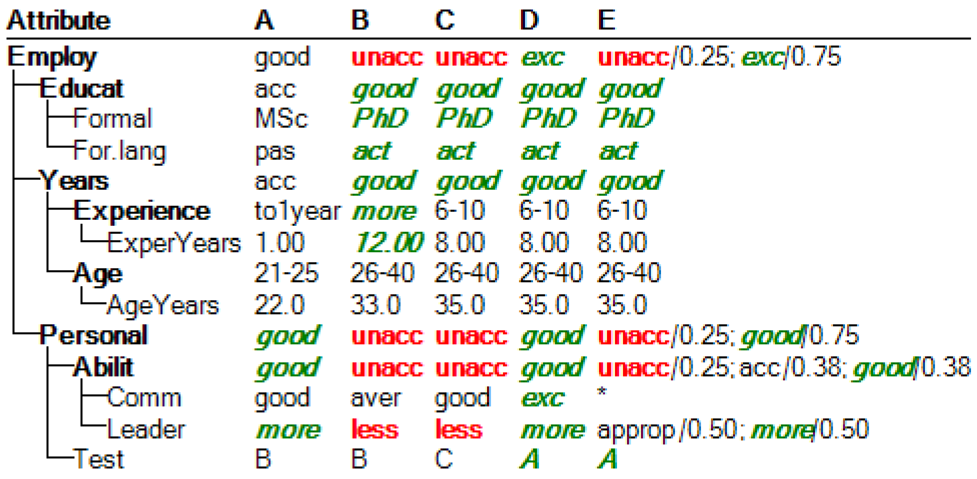
Attribute	A	B	C	D	E
Formal	MSc	PhD	PhD	PhD	PhD
For.lang	pas	act	act	act	act
ExperYears	1.00	12.00	8.00	8.00	8.00
AgeYears	22.0	33.0	35.0	35.0	35.0
Comm	good	aver	good	exc	*
Leader	more	less	less	more	approp/0.50;more/0.50
Test	B	B	C	A	A

There are five candidates, named A, B, C, D, and E. The former four are represented by single values, assigned to the five qualitative and two continuous input attributes.

The candidate E is an exception, aimed at illustrating the use of other types of *DEX values*. The asterisk '*' denotes the whole set of values of the corresponding attribute *Comm*, indicating that nothing is known about E's communicative abilities. The value of *Leader* is also somewhat uncertain, described by the value distribution 'approp/0.50; more/0.50'. This distribution is interpreted later in the *evaluation* either as a probability or fuzzy distribution, but the basic interpretation is that E's leadership abilities are assessed as 'appropriate' or 'more' (with equal strength), but not 'less'.

3.2.6 Evaluation of Employees

The above five employee candidates are evaluated, using the *probabilistic* value interpretation and evaluation method, as follows:



Attribute	A	B	C	D	E
Employ	good	unacc	unacc	*exc*	unacc/0.25; *exc*/0.75
├**Educat**	acc	*good*	*good*	*good*	*good*
│├Formal	MSc	*PhD*	*PhD*	*PhD*	*PhD*
│└For.lang	pas	*act*	*act*	*act*	*act*
├**Years**	acc	*good*	*good*	*good*	*good*
│├**Experience**	to1year	*more*	6-10	6-10	6-10
││└ExperYears	1.00	*12.00*	8.00	8.00	8.00
│└**Age**	21-25	26-40	26-40	26-40	26-40
│ └AgeYears	22.0	33.0	35.0	35.0	35.0
└**Personal**	*good*	unacc	unacc	*good*	unacc/0.25; *good*/0.75
├**Abilit**	*good*	unacc	unacc	*good*	unacc/0.25; acc/0.38; *good*/0.38
│├Comm	good	aver	good	*exc*	*
│└Leader	*more*	less	less	*more*	approp/0.50; *more*/0.50
└Test	B	B	C	*A*	*A*

The final evaluations are: candidate A is 'good', candidates B and C are 'unacc', candidate D is 'exc', and the evaluation of candidate E is a probability distribution of 'unacc' (with probability 0.25) and 'exc' (0.75).

In order to explain why the evaluations are such, one can look at the respective columns and inspect lower-level values that affected the evaluation.

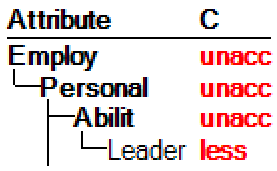
3.2.7 Analysis of Employees

Selective explanation

Selective explanation highlights particular advantages and disadvantages of an alternative. The method finds and displays only those connected sub-trees of attributes for which the alternative has been evaluated as particularly good or bad.

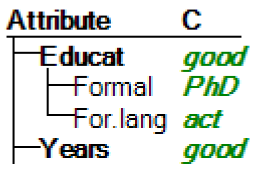
Alternative: **C**

Weak points



Attribute	C
Employ	unacc
└**Personal**	unacc
├**Abilit**	unacc
│└Leader	less

Strong points

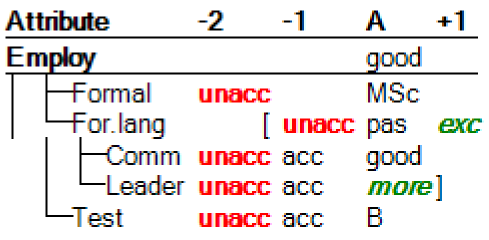


Attribute	C
├**Educat**	*good*
│├Formal	*PhD*
│└For.lang	*act*
├**Years**	*good*

This example shows that candidate C has both weak and strong points. Her particular weakness is *Leader*, which is 'less' and largely affects the 'unacc' evaluation of *Employ*. On the other hand, the candidate is very strong regarding her *Educat* (both *Formal* and *For. lang*) and *Years* (age and experience).

Plus-minus analysis

This is an example of "Plus-minus analysis" of candidate A:



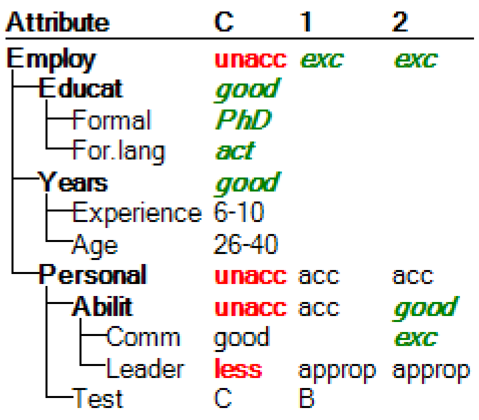
Attribute	-2	-1	A	+1
Employ			good	
├─Formal	unacc		MSc	
└─For.lang	[	unacc	pas	*exc*
├─Comm	unacc	acc	good	
└─Leader	unacc	acc	*more*	]
└─Test	unacc	acc	B	

Overall, candidate A has been evaluated as 'good', as shown in the first row. Column **+1** shows that evaluation can be improved by one step (i.e., to 'exc') only by changing the value of *For. lang* by one step (from 'pas' to 'act'); in this case, the overall evaluation would become 'exc'. In a similar way, the columns **-1** and **-2** show the overall evaluation results when the corresponding attributes (only one at a time) change by one or two steps, respectively.

The brackets '[' and ']' indicate that the values of the corresponding attributes (*For. lang* and *Leader*) cannot be changed by the requested number of steps.

Target Analysis

Target analysis tries to find the smallest changes of input values that improve or degrade the value of some selected aggregate attribute. In contrast with Plus-Minus analysis, possible changes of more than one attribute at a time are observed. The following example shows the results of Target Analysis for candidate C:



Attribute	C	1	2
Employ	unacc	*exc*	*exc*
├─Educat	*good*		
├─Formal	*PhD*		
└─For.lang	*act*		
├─Years	*good*		
├─Experience	6-10		
└─Age	26-40		
└─Personal	unacc	acc	acc
├─Abilit	unacc	acc	*good*
├─Comm	good		*exc*
└─Leader	less	approp	approp
└─Test	C	B	

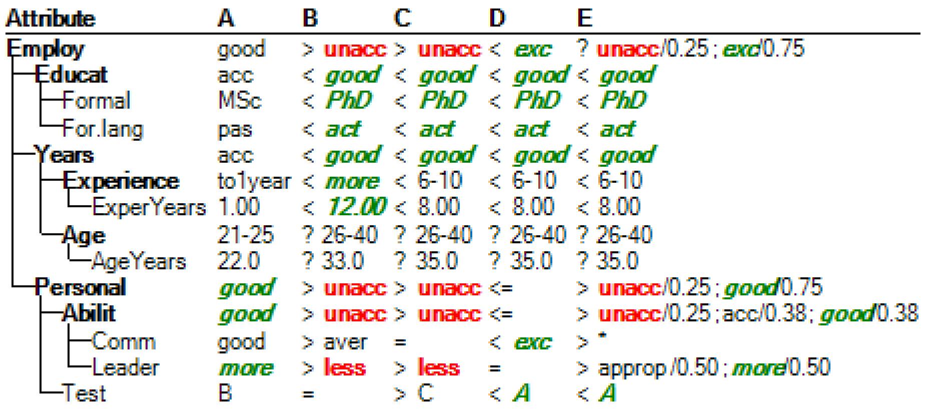
There are two possible ways of improving his evaluation:

1. improving *Leader* from 'less' to 'approp' and *Test* from C to B
2. improving *Comm* from 'good' to 'exc' and *Leader* from 'less' to 'approp'

In both cases, an improvement of two basic attributes at the same time is required. In any case, improving candidate's leadership abilities from 'less' to 'approp' is mandatory.

"Deep" Comparison of Alternatives

When comparing alternatives, it is possible to check the option 'Show comparison operators'. In this case we call the comparison "deep", as DEXiWin tries to establish preferential relations between evaluation values by first inspecting the corresponding decision rules and then, if necessary, by comparing evaluations lower in the model hierarchy.



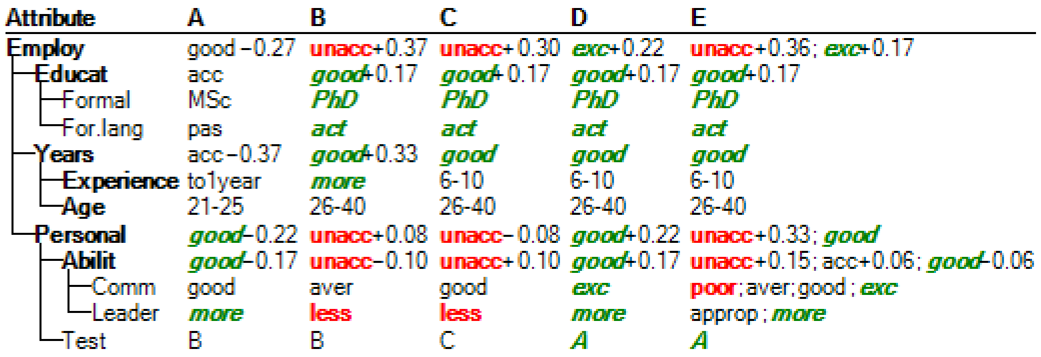
Attribute	**A**	**B**	**C**	**D**	**E**
Employ	good	> unacc	> unacc	< *exc*	? unacc/0.25 ; *exc*/0.75
Educat	acc	< *good*	< *good*	< *good*	< *good*
Formal	MSc	< *PhD*	< *PhD*	< *PhD*	< *PhD*
For.lang	pas	< *act*	< *act*	< *act*	< *act*
Years	acc	< *good*	< *good*	< *good*	< *good*
Experience	to1year	< *more*	< 6-10	< 6-10	< 6-10
ExperYears	1.00	< *12.00*	< 8.00	< 8.00	< 8.00
Age	21-25	? 26-40	? 26-40	? 26-40	? 26-40
AgeYears	22.0	? 33.0	? 35.0	? 35.0	? 35.0
Personal	*good*	> unacc	> unacc	<=	> unacc/0.25 ; *good*/0.75
Abilit	*good*	> unacc	> unacc	<=	> unacc/0.25 ; acc/0.38 ; *good*/0.38
Comm	good	> aver	=	< *exc*	> *
Leader	*more*	> less	> less	=	> approp /0.50 ; *more*/0.50
Test	B	=	> C	< *A*	< *A*

In the above display, the first alternative (candidate A) is compared in a pairwise way with the remaining ones. Operators denote the correponding preferential relations:

- =: indifference, the values are equal
- >: strong preference: the left-hand value is better than the right-hand one
- <: strong preference: the left-hand value is worse than the right-hand one
- <=: weak preference: the left-hand value is worse than or equal to the right-hand one
- >=: weak preference: the left-hand value is better than or equal to the right-hand one
- '?': unknown preference relation (due to unordered value scales or incomparable values)

Qualitative-Quantitative (QQ2) Evaluation

In addition to normal qualitative evaluation of alternatives, *Qualitative-Quantitative* evaluation attempts to rank altenatives within qualitative evaluations. To this end, a numeric offset is assigned to each qualitative evaluation. The maximum range of offsets is [-0.5, +0.5]: the higher the numeric offset, the better the evaluation relative to other evaluations of the same attribute. For instance, 'good-0.27' denotes a somewhat "bad" 'good' value, while 'good+0.17' is much better. Zero offsets are not displayed; 'good' actually means 'good+0.0'.



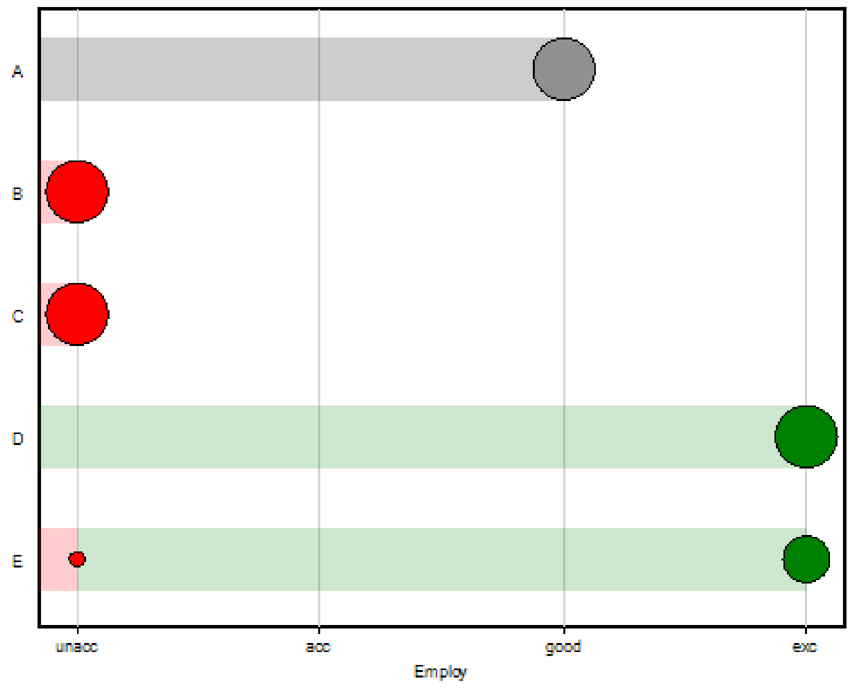
Attribute	**A**	**B**	**C**	**D**	**E**
Employ	good −0.27	unacc+0.37	unacc+ 0.30	*exc*+0.22	unacc+0.36; *exc*+0.17
Educat	acc	*good*+0.17	*good*+ 0.17	*good*+0.17	*good*+0.17
Formal	MSc	*PhD*	*PhD*	*PhD*	*PhD*
For.lang	pas	*act*	*act*	*act*	*act*
Years	acc −0.37	*good*+0.33	*good*	*good*	*good*
Experience	to1year	*more*	6-10	6-10	6-10
Age	21-25	26-40	26-40	26-40	26-40
Personal	*good*−0.22	unacc+0.08	unacc− 0.08	*good*+0.22	unacc+0.33; *good*
Abilit	*good*−0.17	unacc−0.10	unacc+ 0.10	*good*+0.17	unacc+0.15; acc+0.06; *good*−0.06
Comm	good	aver	good	*exc*	poor; aver; good ; *exc*
Leader	*more*	less	less	*more*	approp ; *more*
Test	B	B	C	*A*	*A*

3.2.8 Charts

This example shows some charts that can be obtained in DEXiWin. The charts differ in the number of evaluation dimensions and chart settings.

Bar Chart

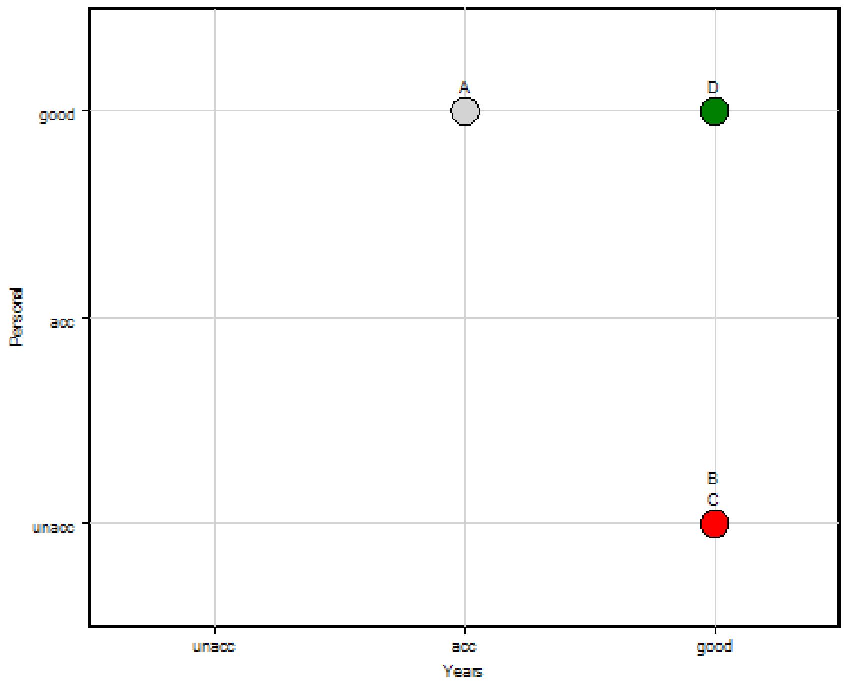
This chart displays evaluation results according to one evaluation dimension. In this case, this is the root attribute *Employ*, so the chart shows the overall evaluation of the five candidates.



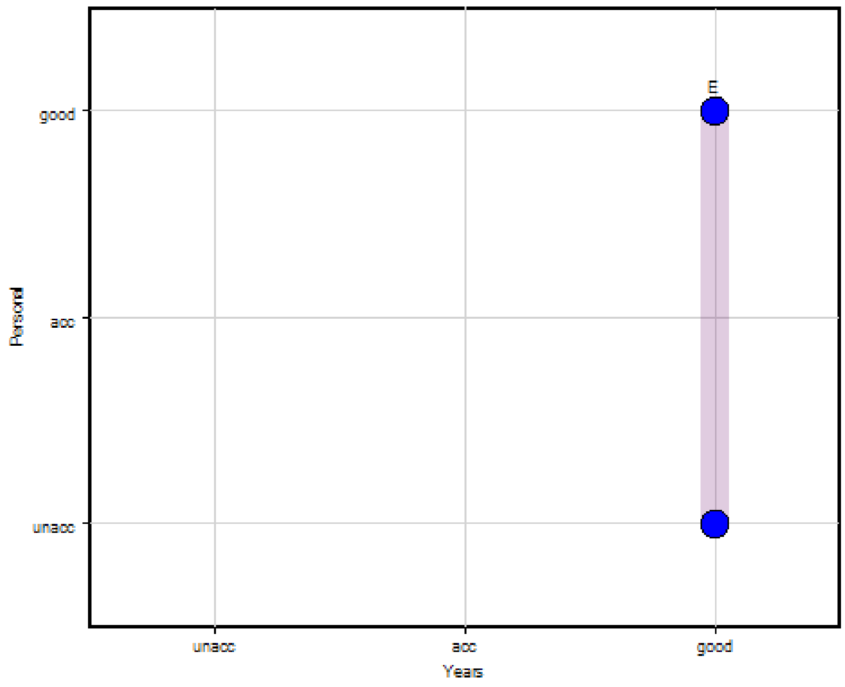
The evaluation of candidate E is distributed between ‘unacc’ and ‘exc’. Point diameters indicate the relative difference of the corresponding probabilities, 0.25 and 0.75.

Scatter Chart

A scatter chart displays evaluation results according to two selected evaluation dimensions. In this case, the selected dimensions are *Personal* and *Years*. The candidate E is excluded from this chart.



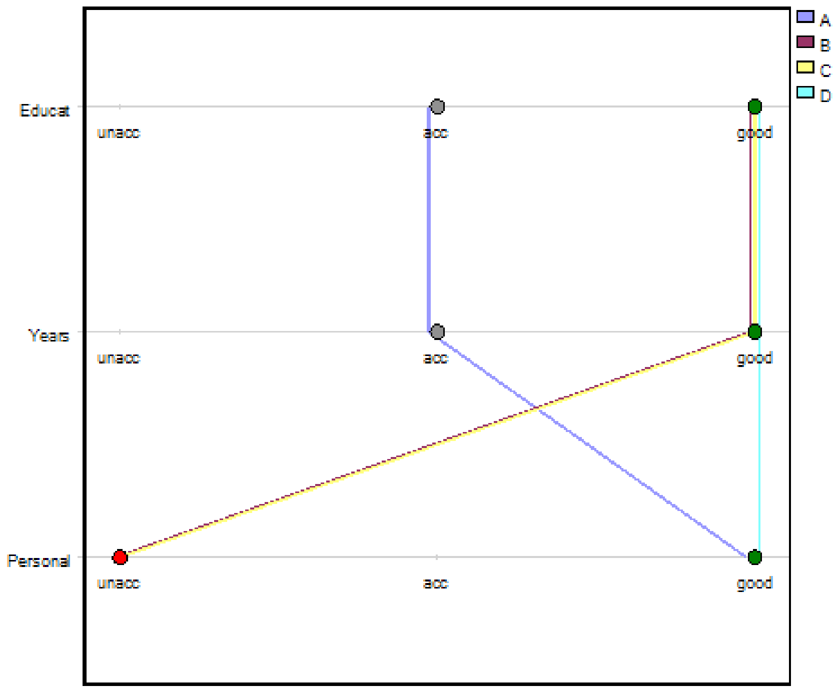
The following is the scatter chart of E, illustrating the display of value distributions:



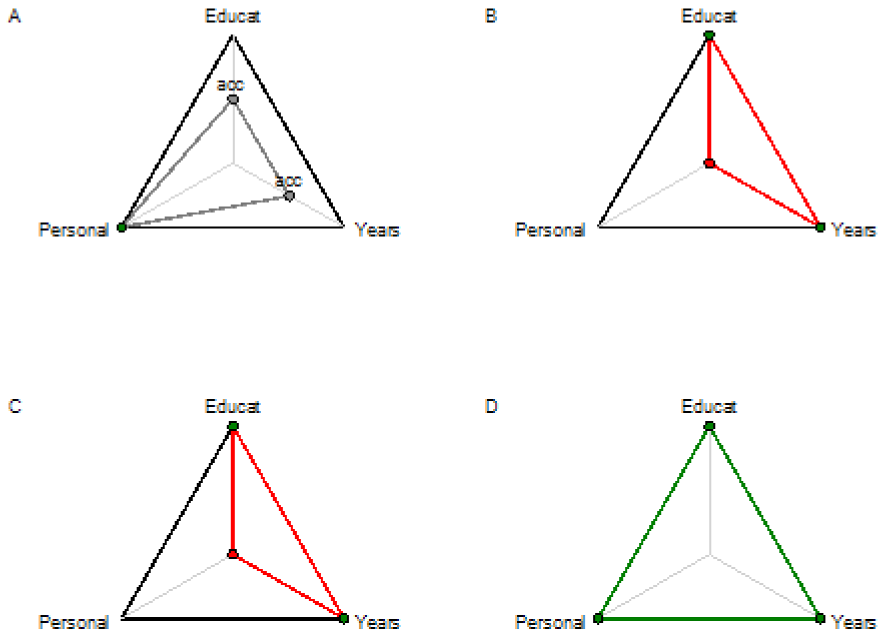
Charts Displaying Three or More Attributes

There are three types of charts that display evaluation results using three or more selected attributes: linear, radar and radar grid. Examples below show all of them. The second-level attributes *Educat*, *Years* and *Personal* have been selected as main chart dimensions. Candidate E has been excluded for clarity.

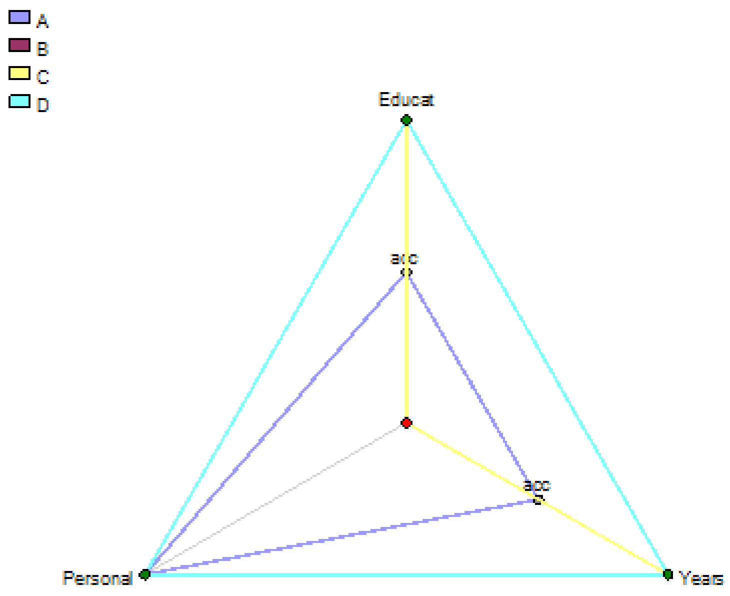
Linear Chart



Radar Grid Chart



Radar Chart



CHAPTER

FOUR

REFERENCES

Bohanec, M., Rajkovič, V. (1988): Knowledge acquisition and explanation for multi-attribute decision making. In: *Proceedings of the 8th International Workshop Expert Systems and Their Applications AVIGNON 88*, vol. 1, pp. 59–78, Avignon.

Bohanec, M., Rajkovič, V. (1990): DEX: an expert system shell for decision support. *Sistemica* 1(1), 145–157.

Bohanec, M. (2022): DEX (Decision EXpert): A qualitative hierarchical multi-criteria method. Multiple Criteria Decision Making (ed. Kulkarni, A.J.), Studies in Systems, Decision and Control 407, Singapore: Springer, doi: 10.1007/978-981-16-7414-3_3, 39-78.

Bana e Costa, C., De Corte, J.-M., Vansnick, J.-C. (2003): MACBETH (Overview of MACBETH multicriteria decision analysis approach). *International Journal of Information Technology & Decision Making* 11, 359–387.

Corrente, S., Greco, S., Słowiński, R. (2012): Multiple criteria hierarchy process in robust ordinal regression. *Decision Support Systems* 53(3), 660–674.

Efstathiou, J., Rajkovič, V. (1979): Multiattribute decisionmaking using a fuzzy heuristic approach. *IEEE Transactions on Systems, Man, and Cybernetics* SMC-9, 326–333.

García-Lapresta, J.L., del Pozo, R.G. (2019): An ordinal multi-criteria decision-making procedure under imprecise linguistic assessments. *European Journal of Operational Research* 279(1), 159–167.

Greco, S., Matarazzo, B., Słowiński, R. (2002): Rough sets methodology for sorting problems in presence of multiple attributes and criteria. *European Journal of Operational Research* 138(2), 247–259.

Greco, S., Ehrgott, M., Figueira, J. (2016): *Multi Criteria Decision Analysis: State of the art Surveys.* Springer, New York.

Ishizaka, A., Nemery, P. (2013): *Multi-criteria Decision Analysis: Methods and Software.* Wiley, Chichester.

Moshkovich, H.M., Mechitov, A.I. (2013): Verbal decision analysis: foundations and trends. *Advances in Decision Sciences* 2013, 697072.

Rajkovič, V., Bohanec, M., Batagelj, V. (1988): Knowledge engineering techniques for utility identification. *Acta Physiologica* 683(1–3), 271–286.

Saaty, T.L., Vargas, L.G.: *Models, Methods, Concepts and Applications of the Analytic Hierarchy Process.* Springer, US, New York.