# LISp-Miner installation

- https://lispminer.vse.cz/download/index.html
- Download LISp-Miner.Core.zip, which is a windows version software.
- Install LISp-Miner on Windows operation system.
- If you use Mac:

    1. Install parallels desktop or virtualbox
    2. Install windows operation system
    3. Install LISp-Miner.

# Dealing with missing values

In LISp-Miner, there are three ways how to handle missing values

- Deleting – missing values are ignored, they are not involved in the algorithm
- Optimistic – missing values support the relationship we are interested in
- Secured – missing values do not support the relationship we are interested in

# Lisp Miner

For each attribute, we should specify which set of literals will be created. This definition is determined
by:

1. Minimal and maximal **length of a literal**.
2. **The type of coefficient** – subsets, intervals, cyclical intervals, left cuts, right cuts, cuts, one particular value
3. One from the following **options**:

- Generate only positive literals – no literals with negation are created
- Generate only negative literals – only literals with negation are created
- Generate both positive and negative literals

# The type of coefficient

- Subsets
- Intervals
- Cyclical intervals
- Left cuts
- Right cuts
- Cuts
- One particular value

# Subsets.

Creation of all possible combinations of categories of the defined length (the order does not matter)

***Example:*** *Create literals of the attribute Type of therapy with its categories {diet, medicaments, operation, none} with minimal length 1 and maximal length 2: Type of therapy (diet), Type of therapy (medicaments), Type of therapy (operation), Type of therapy (none), Type of therapy (diet, medicaments), Type of therapy (diet, operation), Type of therapy (diet, none), Type of therapy (medicaments, operation), Type of therapy (medicaments, none), Type of therapy (operation, none)*

# Intervals.

Sequences of the defined length are created.

**Example:** Create literals of the attribute Age with its categories {⟨20; 30), ⟨30; 40), ⟨40; 50), ⟨50; 60),⟨60; 70)} with minimal length 2 and maximal length 3.

Age [⟨20; 30), ⟨30; 40)], Age [⟨30; 40), ⟨40; 50)], Age [⟨40; 50), ⟨50; 60)], Age [⟨50; 60), ⟨60; 70)], Age [⟨20; 30), ⟨30; 40), ⟨40; 50)], Age [⟨30; 40), ⟨40; 50), ⟨50; 60)], Age [⟨40; 50), ⟨50; 60), ⟨60; 70)].

# Cyclical intervals.

Sequences of the defined length are created, cycles are permitted.
**Example:** Create literals of the attribute Day with its categories {sun, mo, tue, we, thu, fri, sat} with minimal length 3 and maximal length 4.

Day (sun, mo, tue), Day (mo, tue, we), Day (tue, we, thu), Day (we, thu, fri), Day (thu, fri, sat), Day (fri, sat, sun), Day (sat, sun, mo), Day (sun, mo, tue, we), Day (mo, tue, we, thu), Day (tue, we, thu, fri), Day (we, thu, fri, sat), Day (thu, fri, sat, sun), Day (fri, sat, sun, mo), day (sat, sun, mo, tue).

# Left cuts.

Sequences containing only the first category are created.

**Example:** Create literals of the attribute Age with its categories {⟨20; 30), ⟨30; 40), ⟨40; 50), ⟨50; 60),⟨60; 70)} with maximal length 4 (minimal length is by default 1).

Age ⟨20; 30),

Age [⟨20; 30), ⟨30; 40)],

Age [{⟨20; 30), ⟨30; 40), ⟨40; 50)],

Age [⟨20; 30), ⟨30; 40), ⟨40; 50), ⟨50; 60)].

# Right cuts.

Sequences containing only the last category are created.
Example: Create literals of the attribute Age with its categories {⟨20; 30), ⟨30; 40), ⟨40; 50), ⟨50; 60),⟨60; 70)} with maximal length 4 (minimal length is by default 1).
Age ⟨60; 70),

Age [⟨60; 70), ⟨50; 60)],

Age [⟨60; 70), ⟨50; 60), ⟨40; 50)],

Age [⟨60; 70), ⟨50; 60), ⟨40; 50), ⟨30; 40)].

# One particular value.

Only one literal with a particular category will be used.

**Example:** Create literal of the attribute Type of therapy with its category {diet, medicaments, operation, none} containing only operation.

Type of therapy (operation).

# Antecedent and consequent

•A => B, where A (called **antecedent**) and B (called **consequent**) are sets of items.

# Support

$$R: (A_1 = \omega_1) \wedge \ldots \wedge (A_Q = \omega_Q) \wedge (B_1, \alpha_1 \rightarrow \beta_1) \wedge \ldots \wedge (B_P, \alpha_P \rightarrow \beta_P) \Rightarrow (D, k_1 \rightarrow k_2)$$

- Where $(A_1, \ldots, A_Q)$ are stable attributes
- $(\omega_1, \ldots, \omega_Q)$ are values of stable attributes $(A_1, \ldots, A_Q)$
- $\{B_1, \ldots, B_P\}$ are flexible attributes
- $\{(\alpha_1 \rightarrow \beta_1), \ldots, (\alpha_P \rightarrow \beta_P)\}$ are changes of values of flexible attributes $\{B_1, \ldots, B_P\}$
- D is a decision
- $(k_1 \rightarrow k_2)$ is a change of decision from $k_1$ to $k_2$

# Support

- CPL(R)...number of objects matching $(\omega_1,\ldots,\omega_Q,\alpha_1,\ldots,\alpha_P, k_1)$, i.e. number of objects matching the state before a change which also match the state of decision before the change
- CPR(R): number of objects matching $(\omega_1,\ldots,\omega_Q,\beta_1,\ldots,\beta_P, k_2)$, i.e. number of objects matching the state after a change which also match the state of decision after the change
- CVL(R): number of objects matching $(\omega_1,\ldots,\omega_Q,\alpha_1,\ldots,\alpha_P)$, i.e. number of all objects matching the state before a change
- CVR(R): number of objects matching $(\omega_1,\ldots,\omega_Q,\beta_1,\ldots,\beta_P)$, i.e. number of all objects matching the state after a change
- $\text{LeftSup(R)} = \frac{\text{CPL(R)}}{n}$, where n is a total number of objects in the database
- $\text{RightSup(R)} = \frac{\text{CPR(R)}}{n}$

# Output

| Nr. | **Hypothesis** | D%-Sum | **Df-Conf** | Df-AFUI | Df-FUE | Df-Avg | R-Conf | R-DFUI |
|---|---|---|---|---|---|---|---|---|
| | R-FUE | R-Avg | H-Conf | H-DFUI | H-FUE | H-Avg | B:a | B:b |
| | B:c | B:d | B:r | B:n | B:Conf | B:DConf | B:EConf | B:Supp |
| | B:Cmplt | B:AvgDf | B:Lift | B:LBound | B:UBound | B:ELBound | B:EUBound | B:DLBound |
| | B:DUBound | B:Fisher | B:Chi-Sq | B:PSep | B:bMean | B:bVAR | B:bStDev | B:P(>=90%) |
| | B:P(>=95%) | A:a | A:b | A:c | A:d | A:r | A:n | A:Conf |
| | A:DConf | A:EConf | A:Supp | A:Cmplt | A:AvgDf | A:Lift | A:LBound | A:UBound |
| | A:ELBound | A:EUBound | A:DLBound | A:DUBound | A:Fisher | A:Chi-Sq | A:PSep | A:bMean |
| | A:bVAR | A:bStDev | A:P(>=90%) | A:P(>=95%) | | | | |
| 255 | **Survey_Type(Field) : (Benchmark__Service_Tech_Equipped_to_do_Job(9) & Benchmark__Service_Tech_Promised_in_Expected_Timeframe(9) -> Benchmark__Service_Tech_Equipped_to_do_Job(10) & Benchmark__Service_Tech_Promised_in_Expected_Timeframe(10)) > ⇒ < (CustomerStatus(Active) -> CustomerStatus(Leaving))** | | | | | | | |
| 1.42 | **0.9104716857** | -0.0087952481 | | -0.619438249 | 0.6520948122 | 40.8252873563 | | 0.4668259929 |
| | 0.2874103066 | -0.5644572759 | | 40.8252873563 | | 2.1421257923 | 3.4793463459 | -1.7716132693 |
| | 154 | 11 | 19833 | 6455 | 165 | 26453 | 0.9333333333 | 0.0077007701 |
| | 0.2498393377 | 0.0058216459 | 0.0077050083 | 0.2352762629 | 1.2352762629 | 0.0921864586 | 0.9473117288 | 1 0 |
| | 0.9999999999 | 0 | 0.0000000017 | 28.4098845456 | | 127.8571428571 | | 0.9281437126 |
| | 0.9281437126 | 0.019924405 | 0.9117630263 | 0.1291964958 | 58 | 2479 | 979 | 22937 |