Constraint Programming: Powerful paradigm of declarative programming that lies at the intersection of AI, operations research (OR) and logic programming. It is used to solve complex combinatorial problems in various fields such as planning, resource optimization, robotics and intelligent transportation systems. The user specifies a problem in terms of constraints, and the machine solves it, making this paradigm attractive for addressing complex challenges.

Constraint Network: A constraint network $N = (X, D, C)$ is defined on a given vocabulary (X, D) where we add a finite set of constraints C. Each constraint $c \in C$ is a pair $\langle var(c), rel(c) \rangle$, where $var(c) \rightarrow$ sequence of variables from X called the scope of the constraint and $rel(c) \rightarrow$ relation on $D^{|var(c)|}$ called the constraint relation which defines the allowed combinations of simultaneous values for the variables in $var(c)$.

Constraint Language: A constraint language is a set $\Gamma = \{r_1, \dots, r_t\}$ of relations over a subset of $\mathbb{Z}^n$.

CSP: A constraint satisfaction problem is the task of finding a solution that satisfies all constraints in a constraint network.

COP: A constraint optimization problem is defined by: a constraint network (X, D, C) and an objective function $f: D^n \rightarrow \mathbb{R}$ to be minimized or maximized. The goal is to find a solution to the constraint network that min/max the objective function.

AC-3: The AC-3 algorithm ensures that each arc (x_i, x_j) in the constraint graph is consistent, meaning that for every value of x_i, there exists a compatible value of x_j. The input is a constraint network (X, D, C), the output a network made arc-consistent, and the usage of the algo is for preprocessing before search to reduce domains.

B&B: Branch and Bound is a systematic search algorithm used in optimization problems (including constraint programming with an objective function like $c = f(X)$ where X represents the variables of the CSP. This is called the objective constraint). It combines backtracking search with pruning using bounds on the objective function.

NP-Hard Problem: A problem is called NP-Hard (Non-deterministic Polynomial-time Hard) when its resolution is extremely challenging, even for powerful computers. This means that for large problem sizes, it is practically impossible to find an optimal solution in a reasonable time.

CP Advantages:

- **Compactness**: CP allows problems to be modeled concisely. Constraints directly express complex relationships between variables, making models simpler and more readable than traditional linear programming equations.
- **Efficiency** (often): Although CP involves searching in a potentially very large solution space, modern solvers are often able to exploit constraint propagation techniques and heuristics search to efficiently find solutions, even for large problems.
- **Expressiveness**: CP provides great flexibility to express a wide variety of constraints, whether linear, non-linear, global or even logical. This allows handling highly diverse problems, from planning to designing complex systems.
- **Natural Modeling**: Many problems, especially in planning, design or optimization, can be naturally formulated in terms of constraints. This declarative model simplifies problem specification compared to other approaches like ILP. * encodings of pseudo-Boolean constraints
- **Flexibility and Adaptability**: CP can easily integrate new constraints or modify existing ones without disrupting the entire model.
- **Separation of Search and Modeling**: In CP, problem modeling is separated from solution searching (constraint solver) allowing designers to focus on defining requirements without worrying about the details of the solving algorithm.

Vocabulary: Pair (X, D) where:

- X is a finite set of variables, denoted $X = \{x_1, \dots, x_n\}$
- D is a finite subset of $\mathbb{Z}$ called the domain containing the possible values the variables can take.

BT: Backtracking is a systematic search method that constructs a partial instantiation by assigning variables one by one. When an instantiation violates a constraint, the method backtracks to try another value. It's simple to implement but inefficient if domains are large or constraints numerous.

FC: Constraint propagation method used during search, especially when assigning variables. At each step, when a variable is instantiated, FC checks the constraints between this variable and the uninstantiated ones, removing incompatible values from their domains.

MAC: Maintaining Arc Consistency combines search and propagation by maintaining arc-consistency after each assignment. Unlike FC, it applies AC3 at each step to ensure all arcs remain consistent.

Heuristics in CSP search: Heuristics significantly enhance search efficiency by guiding which variable to assign first and which value to choose for that variable.

Global constraint: A global constraint is a constraint defined over an arbitrary set of variables that captures a specific property or pattern, often recurring across multiple problem domains.

⇒ **Advantages**:

- **Capture High-Level Properties**: A global constraint expresses a complex property in a single declarative statement.
- **Enhance Propagation**: By using advanced filtering algorithms they achieve higher levels of domain reduction thus decreasing the search space.
- **Encourage Reuse**: Commonly used global constraints can be applied across various problems and domains, fostering modularity.

Data Mining: Often referred to as Knowledge Discovery in Databases (KDD), is a multidisciplinary field that focuses on uncovering meaningful patterns, relationships and insights from large and complex datasets. The ultimate goal of DM is to transform raw data into actionable and valuable knowledge that can be used for decision-making, prediction and exploration.

Challenges in DM: Data Complexity, Scalability, Interpretability, Ethical concerns and User Centering and Human-in-the-loop.

Steps in KDD Process:

1. **Data Selection**: Identifying the relevant datasets for analysis, often from heterogeneous data.
2. **Data Preprocessing**: Cleaning, transforming and normalizing the data to ensure consistency and quality.
3. **Data Transformation**: Converting raw data into formats or features suitable for analysis.
4. **Data Mining**: Applying algorithms to extract patterns, associations or predictive models.
5. **Evaluation & Interpretation**: Assessing the validity, relevance and usefulness of the discovered knowledge.
6. **Knowledge Presentation**: Visualizing and communicating results in a form that stakeholders can readily understand and act upon.

Frequent Pattern Mining: FPM has gained considerable attention due to its broad applicability and theoretical richness. Originally introduced by Agrawal et al., involves identifying and analyzing recurring patterns within datasets.

Transactional dataset D: collection of m itemsets, denoted as $t_1, \dots, t_m$ which are called transactions.

Items: An item is one distinct object (e.g. $\{p_1, \dots, p_n\} = I$)

Itemset: An itemset P is any non-empty subset of these items I.

Cover: The cover of an itemset P, denoted as $cover(P)$, refers to the set of transactions in D that contain P.

Frequency: The frequency of an itemset P in D, denoted as $freq(P)$ is the cardinality of its cover, i.e. $freq(P) = |cover(P)|$. An itemset is said to be frequent if its frequency is above a given threshold α. Otherwise, the itemset is infrequent.

Closed Itemset: Given a dataset D, a minimum frequency α and an itemset P: P is **closed** iff there does not exist any superset $Q \supset P$ such that $freq(Q) = freq(P)$. Respectively, P is **generator** iff there does not exist any subset $Q \subseteq P$ such that $freq(Q) = freq(P)$.

Maximal and Minimal Itemsets: Given a dataset D and a minimum frequency α, the itemset P is **maximal** iff P is frequent and there does not exist any superset $Q \supseteq P$ s.t. $freq(Q) \geq \alpha$. Respectively, P is **minimal** iff P is infrequent and there does not exist any subset $Q \subseteq P$ such that $freq(Q) < \alpha$.

Frequent Itemsets ⊇ Closed Itemsets ⊇ Maximal Itemsets

Minimum Non-Redundant Rule: Given a dataset D, a min frequency α and a min confidence β, a MNR $X \rightarrow Y$ is a valid rule such that there does not exist any rule $r' : X' \rightarrow Y'$ with $X' \subseteq X$, $Y \subseteq Y'$, $freq(r) = freq(r')$, $conf(r) = conf(r')$ and $r \neq r'$.

Association Rule: An association rule captures information of the kind "if we have A and B, the chances to have C are high". It's an implication of form $X \rightarrow Y$, where X and Y are itemsets such that $X \cap Y = \emptyset$ and $Y \neq \emptyset$. X represents the body of the rule and Y represents the head. The **frequency** of the rule $X \rightarrow Y$, is the frequency of the itemset $X \cup Y$, i.e. $freq(X \rightarrow Y) = freq(X \cup Y)$. The **confidence** of a rule captures how often Y occurs in transactions containing X, that is, $conf(X \rightarrow Y) = freq(X \rightarrow Y) / freq(X)$. A rule is known as a valid rule if its frequency and confidence are greater than or equal to user-specified thresholds (α and β). The problem of mining association rules consists in generating all valid rules.

Anti-Monotonicity Property: The anti-monotonicity property implies that if a set of items is infrequent, then any larger set that contains this set must also be infrequent. Let $P = \{a_1, \dots, a_m\}$ be an itemset of m items where $a_i \in I$ and $a_i < a_{i+1}$ according to a fixed order on the items.

Tail: The tail of P is the last item in P, i.e. $tail(P) = a_m$.

Sub-itemsets: The sub-itemset of P up to the jth item is: $P(a_j) = \{a_1, \dots, a_j\}$.

Prefix: The prefix of P in the prefix tree, is defined as $prefix(P) = \{a_1, \dots, a_{m-1}\}$.

Apriori Property: Direct consequence of anti-monotonicity which states that if an itemset is frequent, then all of its subsets must also be frequent. Let P be an itemset. If P is frequent, then all subsets of P, $\forall S \subseteq P$, must also be frequent; $freq(P) \geq \alpha$ then $freq(S) \geq \alpha \; \forall S \subseteq P$.

Apriori Algorithm: The basic idea behind the Apriori algo is a breadth-first search where candidate itemsets are generated by joining frequent itemsets from the previous level and infrequent itemsets are pruned using the anti-monotonicity rule.

Closure: A set S has a closure under an operation f if: $\forall x \in S \Rightarrow f(x) \in S$. A closure operation is extensive, idempotent, monotone.

$S \subseteq closure(S) \longleftrightarrow clo(clo(S)) = clo(S)$; $X \subseteq Y \Rightarrow clo(X) \subseteq clo(Y)$

Itemset Closure: The closure of an itemset P is the intersection of all the transactions t in the cover of P: $closure(P) = \bigcap_{t \in cover(P)} t$. An itemset is said to be closed if it is equal to its closure: $P = closure(P)$.

LCM Algorithm (Linear-Time Closed Itemset Mining): The LCM Algo is an efficient algorithm designed for discovering frequent closed itemsets in large transactional datasets. The LCM algo aims at efficiently generate closed itemsets which can provide more compact and meaningful insights into the data compared to regular frequent itemsets. It uses a lexicographic order traversal (depth-first) to systematically explore candidate itemsets and avoid redundancy.

Step-by-Step Process:

1. Start with a frequent closed itemset P.
2. After adding a new item, we compute the closure of the extended itemset $P' = P \cup \{i\}$ → $closure(P') = \bigcap_{t \in cover(P')} t$
3. If the closure of P' equals P' (P' is closed), then P' becomes a valid closed itemset.
4. Recursively repeat the process for each new valid closed itemset until no more can be found.

Why closure extension works?

- **Monotonicity**: The closure of any itemset is always [illegible]
- **Efficient Construction**: The closure extension rule allows LCM algo to systematically discover all closed itemsets by incrementally extending smaller closed itemsets and checking for closure, without having to check for every possible combination of items directly.

CP4IM: Model that relies on 2 vectors of Boolean variables:

- **Decision Variables**: $X_1, \dots, X_n$ where $X_i = 1 \Leftrightarrow$ item i is included in the extracted itemset
- **Auxiliary variables**: $T_1, \dots, T_m$ where $T_t = 1 \Leftrightarrow$ the extracted itemset appears in transaction t of the dataset D

Encoding CP4IM Constraints:

1. **Cover**: $\forall t \in D: T_t = 1 \Leftrightarrow \sum_{a_i \in I} X_i (1 - cover(t, a_i)) = 0$ → $\begin{cases} 1 & \text{if item } a_i \text{ in } t \\ 0 & \text{otherwise} \end{cases}$
2. **MinFreq**: $\sum_{t \in D} T_t \geq \alpha$
3. **Closedness**: $\forall a_i \in I: (X_i = 1) \Leftrightarrow \sum_{t \in D} T_t (1 - cover(t, a_i)) = 0$
4. **Maximality constraint**: $\forall a_i \in I: (X_i = 1) \Leftrightarrow \sum_{t \in D} T_t \, cover(t, a_i) \geq \alpha$

Together, these Boolean variables and constraints allow a CP solver to declaratively find frequent, closed, or maximal itemsets in a dataset.

DIM: Declarative Itemset of Data Mining leverages constraint programming (CP) and propositional satisfiability (SAT) to model and solve data mining tasks in a more flexible way.

Size Independent Constraint Network: A SICN is essentially a CP model for a data-mining problem such as itemset mining or association rule mining that does not directly encode any constraint on the size of the target patterns. Instead, it focuses on the core relationships among the variables and the domain consistency properties needed to prune the search space. This approach still allows the use of global constraints but abstracts away from size-based filtering.

SAT-Based Approaches: SAT-based itemset mining translates the mining task into a Purely Boolean formula, which can be handled efficiently by modern SAT solvers. These methods have been successfully applied to finding closed and maximal itemsets, top-k frequent itemsets and association rules. Compared to CP4IM, SAT-based approaches gain efficiency from:

- on the fly [illegible] learning
- Parallel and incremental solving
- Specialized [illegible]

Global Constraints: GC in DM offer 2 main benefits:
- **High-level Expression**: They allow complex mining tasks to be defined compactly. They enable CP solvers to
- **Enhanced Propagation**: prune the search space more effectively by leveraging advanced DM techniques and data structures.

Closed Pattern GC: Def pag 6 — 3 propagation rules

Generator GC: Def — Time complex pag 7

FreqSub & InfreqSup: Both GC are used to help identify maximal and minimal itemsets by enforcing additional frequency conditions on their subsets or supersets. Def for both pag 7

FreqSub → It requires that every proper subset Q of itemset P_X must be frequent.
InfreqSup → It ensures that for a candidate itemset P_X every proper superset Q is infrequent.

FreqRare GC: This GC tackles the problem of using a single minimum frequency threshold in itemset mining, which can either miss interesting combinations of frequent rare items, or generate too many irrelevant itemsets. Instead of one global threshold α, FreqRare allows each item i to have its own minimum frequency threshold α_i. An itemset P is then considered frequent if $freq(P) \geq \min_{i \in P} \alpha_i$.

Confident GC: The confident GC ensures that an association rule meets a specified minimum confidence threshold β in the dataset. Pag 9

Generic CP Model: The main idea is that CP models offer a highly flexible, declarative framework for itemset mining. Instead of preprocessing data to create multiple specialized sub-datasets, CP allows users to directly specify both
- **what to mine**: user-specific constraints on the itemset (e.g. only mining itemsets that involve certain items)
- **where to mine**: constraints on items or transactions (e.g. considering trans over a certain amount or within specific time windows).

This approach is encapsulated in a generic CP model, such as the ItemSet model, which efficiently integrates all these constraints to generate only interesting itemsets without additional implementation complexity. The vocabulary of GCP model is defined using 4 boolean vbles:
- **Candidate Itemset** $X = \langle X_1, \ldots, X_n \rangle$ where $X_i = 1$ indicates that item i is in the candidate.
- **Transactions**: $T = \langle T_1, \ldots, T_m \rangle$ represents trans covered by the extracted candidate itemset.
- $H = \langle H_1, \ldots, H_n \rangle$ determines which items are considered for mining in the sub-dataset. If $H_i = 0$, then item i is ignored in the mining process.
- $V = \langle V_1, \ldots, V_m \rangle$ determines which transactions are considered for mining in the sub-dataset. If $V_j = 0$, then the transaction j is ignored.

The pair (H,V) defines the sub-dataset where mining occurs, allowing the CP solver to explore different parts of the dataset by selecting or excluding specific items and transactions. Meanwhile, the pair (X,T) captures the candidate itemset and its corresponding cover.

Constraints of GCP Model: Consists of 3 sets of constraints: $ITEMSET(\langle X,T \rangle, \langle H,V \rangle) = \begin{cases} DATASET(\langle H,V \rangle) \\ CHANNELING(\langle X,T \rangle, \langle H,V \rangle) \\ MINING(\langle X,T \rangle, \langle H,V \rangle) \end{cases}$
- DATASET: Expresses user-specific constraints on items (H) and/or trans (V). It defines the sub-datasets that will be used for mining.
- CHANNELING: These constraints link the sub-datasets (items and trans chosen via H and V) with the mining variables that define the candidate itemset and its cover.
- MINING: These constraints express the properties of the itemset you want to mine, such as frequency, closedness, size or any user-specific pattern restrictions.

CP Model for Mining Borders: It leverages the duality between minimal and maximal itemsets to efficiently distinguish between frequent and infrequent itemsets. The key idea is that an itemset with only frequent subsets and only infrequent supersets is on the border, either as a maximal frequent itemset (positive border) or a minimal infrequent itemset (negative border). These constraints (maximal frequent itemsets) capture the necessary properties to decide border membership: Such model is called $Borders_\alpha$ and operates on a boolean vector X (where each X_i indicates the presence of an item) and includes a reified boolean parameter α (≠ frequency support) that indicates whether the itemset is frequent (α = True) or not (α = False). If α = True, the model enforces: $freq(X) \geq \alpha$, FreqSub(X) (all proper subsets of X are frequent) and InfreqSup(X) (all proper supersets of X are infrequent). If α = False, the model enforces: $freq(X) < \alpha$, FreqSub(X) and InfreqSup(X) (→ minimal infrequent itemsets)

Mining Association Rules: For mining AR, 3 vectors are needed, one for the body X, one for the head of the rule Y and Z for the rule as a whole: X_i, Y_i and Z_i represent the presence of item i in the body, head and rule (body or head). The model involves 5 constraints:

$Rule(X,Y,Z) = \begin{cases} \forall i \in I: \neg X_i \vee \neg Y_i \to \text{Ensures that a given item can't be in the body and head simultaneously} \\ \bigvee_{i \in I} Y_i \to \text{Ensures head of rule is not empty} \\ \forall i \in I: Z_i \leftrightarrow X_i \vee Y_i \to \text{channeling constraint ensuring } Z = X \cup Y \\ Confident(X,Y) \to \text{GC ensuring that extracted rule is confident} \\ Frequent(Z) \to \text{GC ensuring rule is frequent} \end{cases}$

Extension to MNR Rules: An AR $r: X \to Y$ is an MNR iff 1) r is a valid rule, 2) X is a generator, 3) X∪Y is closed itemset. To implement such extension we define the model: $MNRule(X,Y,Z) = \begin{cases} Rule(X,Y,Z) \\ Generator(X) \\ ClosedPattern(Z) \end{cases}$ (Z = X∪Y)

AC-3 Algorithm: 1) Determine the initial propagation queue (all constraints from both sides → and ←) 2) Recall all domains per variable 3) Add arcs reaching the origin variable to the queue.

Example: $Q = \{\overrightarrow{X_1 > X_2}, \overrightarrow{X_2 < X_1}, \overrightarrow{X_2 = X_3}, \overrightarrow{X_3 = X_2}\}$

1) $\overrightarrow{X_1 > X_2}$, $d_{X_1} = \{1,2,3\}$, $d_{X_2} = \{1,2\}$
$X_1 = 1 > 1$ ✗
$X_1 = 2 > 1$ ✓ ⇒ $d_{X_1} = \{2,3\}$ ⇒ $Q = Q \cup \{\text{All arcs reaching } X_1\}$ (since we have just started no need)
$X_1 = 3 > 1$ ✓

2) $\overrightarrow{X_2 < X_1}$, $d_{X_1} = \{2,3\}$, $d_{X_2} = \{1,2\}$
$X_2 = 1 < 2$ ✓
$X_2 = 2 < 3$ ✓ ⇒ $d_{X_2} = \{1,2\}$ ⇒ No changes

3) $\overrightarrow{X_2 = X_3}$, $d_{X_2} = \{1,2\}$, $d_{X_3} = \{2,3\}$
$X_2 = 1 \neq 2,3$ ✗ ⇒ $d_{X_2} = \{2\}$ ⇒ $Q = Q \cup \{\overrightarrow{X_1 > X_2}\}$
$X_2 = 2 = 2$ ✓

4) $\overrightarrow{X_3 = X_2}$, $d_{X_2} = \{2\}$, $d_{X_3} = \{2,3\}$
$X_3 = 2 = 2$ ✓ ⇒ $d_{X_3} = \{2\}$ ⇒ $Q = Q \cup \{\overrightarrow{X_2 = X_3}\}$ (If the end, you study them)
$X_3 = 3 \neq 2$

After AC-3 no further changes (domain reductions) are needed, then the network has reached arc consistency because arcs are consistent with the constraints. Arc-consistency means that for each vble, for each possible * value in its domain, there must be a valid value in the domain of other variables that satisfies all constraints.

MAC-Algorithm: Queens in Chess
$X = \{C_1, \ldots, C_N\}$ where $C_i = a$ means that in row i there is a queen in column a.
$D = \{1, \ldots, N\}$ and search space $|D|^{|X|} = N^N$
C = {Column Const ∪ Diagonal Const} =
$\{\forall i,j \in \{1,\ldots,N\}$ s.t $i<j$, then $C_i \neq C_j\} \cup \{\forall i,j \in \{1,\ldots,N\}$ s.t $i<j$, then $|C_i - C_j| \neq |i-j|\}$

Partial Instantiation: $C_1 = 1$; $C_2 = 3$
First C_3 and then C_4. For C_3 we know that $C_3 \neq C_i \, \forall i \Rightarrow C_3 = 2$ or 4. Besides, $\forall i \in \{1,2\}$ s.t $i \neq 3$, $|C_3 - C_i| \neq |3-i|$.

Case: $C_3 = 2$
- $|C_3 - C_1| = |2-1| = 1 \neq 2 = |3-1|$ ✓
- $|C_3 - C_2| = |2-3| = 1 = 1 = |3-2|$ ✗

However, the condition is not satisfied $\forall i \in \{1,2\}$.

Case: $C_3 = 4$
- $|C_3 - C_1| = |4-1| = 3 \neq 2 = |3-1|$ ✓
- $|C_3 - C_2| = |4-3| = 1 = 1 = |3-2|$ ✗

However, the condition not satis $\forall i \in \{1,2\}$. Therefore no possible value can be assigned to C_3. We continue with C_4, that needs to satisfy that $C_4 = 2$ or $C_4 = 4$.

Case: $C_4 = 2$
- $|C_4 - C_1| = |2-1| = 1 \neq 3 = |4-1|$ ✓
- $|C_4 - C_2| = |2-3| = 1 \neq |4-2| = 2$ ✓

All conditions are satisfied, we conclude $C_4 = 2$.

Case: $C_4 = 4$
- $|C_4 - C_1| = |4-1| = 3 = |4-1|$ ✗
- $|C_4 - C_2| = |4-3| = 1 \neq |4-2| = 2$ ✓

Condition not satisfied $\forall i \in \{1,2\}$. The final assignation is $C_4 = 2$.

Horizontal Transactional Database Representation

Trans	Items					
t1		B	C	D		
t2	A	B	C		E	
t3	A	B	C	D		F
t4				D	E	

H_{D_1} ↑

Vertical Representation:

$V_{D_1} = \begin{pmatrix} A & B & C & D & E & F \\ t_2 & t_1 & t_1 & t_1 & t_2 & t_3 \\ t_3 & t_2 & t_2 & t_3 & t_4 & 0 \\ 0 & t_3 & t_3 & t_4 & 0 & 0 \end{pmatrix}$

The Matrix Representation:

$M_{D_1} = \begin{pmatrix} 0 & 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 & 0 \end{pmatrix}$

Support = Absolute Frequency:
$support(BC) = |\{t_1, t_2, t_3\}| = 3$
$support(A) = |\{t_2, t_3\}| = 2$

$R_F(BC) = \frac{3}{4}$
$R_F(A) = \frac{2}{4} = 1/2$

Relative Frequency:
$R_F(Itemset) = \frac{support(Itemset)}{|Dataset|}$

Finding Frequent Itemsets:
1) Find support for each individual itemset (item)
2) Combine only those satisfying the minimum support values in lexical order (to ease task)

Comparable Itemset:
Two itemsets X,Y are comparable ⟺ $X \subseteq Y \vee Y \subseteq X$
Non-comparable Itemset:
Two itemsets X,Y are not comparable ⟺ $X \not\subseteq Y \wedge Y \not\subseteq X$

Maximal Itemset Exercise: $M_\alpha = \{ABC^3, DE^2, EF^5\}$ is a set of max itemsets. The list of frequent itemsets: {ABC, AB, AC, BC, A, B, C, DE, D, E, EF, F}. Remember that by the Apriori property, if an itemset is frequent then all of its subsets are frequent. If they ask for all frequent itemsets for a support threshold α = 3, the list would be: {ABC, AB, AC, BC, A, B, C, E, F, EF}. The little number in ABC^3 means that all subsets of itemsets in ABC have a support greater or equal to 3.

Closed Itemset Exercise: $C_\alpha = \{ABC^3, ABE^5, DE^2, EF^5\}$. We know that a closed itemset is a frequent itemset for which no proper superset has the same support. To provide the list of frequent itemsets, we reuse the Apriori property, and then all subsets are frequent depending on the little number.

APRIORI-ALGORITHM: Let's consider the dataset.

Trans	Items				
t1	A		C	D	
t2		B	C		E
t3	A	B	C		E
t4		B			E
t5	A	B	C		E
t6		B	C		E

The Apriori algorithm involves generating itemsets of increasing size and filtering out those that do not meet the minimum support threshold. We proceed with the algo:

Step 1: Initialize the set of candidates (1-itemsets)
cover(A) = 3 ✓ cover(D) = 1 ✗
cover(B) = 5 ✓ cover(E) = 5 ✓ ⇒ **Step 2**: Prune — Keep: A, B, C, E; Prune: D
cover(C) = 5 ✓

Step 3: We generate all possible 2-itemsets from the frequent 1-itemsets:
cover(AB) = 2 ✗ cover(BC) = 4 ✓
cover(AC) = 3 ✓ cover(BE) = 5 ✓ ⇒ **Step 4**: Prune — Keep: AC, BC, BE, CE; Prune: AB, AE
cover(AE) = 2 ✗ cover(CE) = 4 ✓

Step 5: Generate all possible 3-itemsets from the frequent 2-itemsets.
cover(ABC) = 2 ✗ cover(ABE) = 2 ✗
cover(ACE) = 2 ✗ ⇒ **Step 6**: Prune — Keep: CBE; Discard: ABC, ACE, ABE
cover(CBE) = 4 ✓
Stop: no more freq itemsets of size 4.

Results: 1-itemsets: {A, B, C, E}; 2-itemsets: {AC, BC, BE, CE} and 3-itemsets: {CBE}.

How to determine maximal/closed itemsets? Let's say you have already computed the frequent itemsets per size:
1-itemsets: {A, B, C, E} (3,5,5,5)
2-itemsets: {AC, BC, BE, CE} (3,4,5,4)
3-itemsets: {BCE} (4)

Then, it's easy to deduce that none of the 1-itemsets are maximal as adding C, C, C, E and C to A, B, C, E respectively generates a frequent itemset of size 2. Then, adding as well E, C and B to BC, BE and CE respectively generates a 3-itemset BCE which is frequent. In conclusion, the only 2 itemsets which are maximal are AC and BCE (because there are no more frequent itemsets of size 4).

Now, in terms of closed itemsets, we need the supports. A closed itemset has no superset with the same cover, then you need to check each frequent itemset and search if he has a superset with same support, if not then it is closed. Remember that you can partially check: the maximal itemsets from before should be also closed.

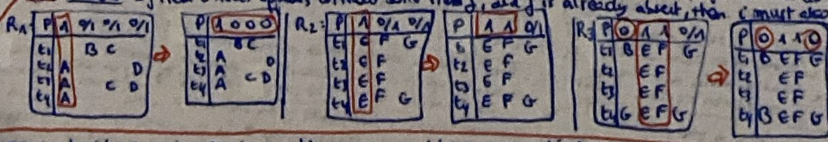
LCM ALGORITHM: In our LCM implementation we assume the items are ordered lexicographically (A<B<C<D<E<F) and we "grow" itemsets from the empty set (whose cover is all transactions) by appending items in that order.
1) $cover(A) = |\{t_2, t_3, t_5, t_6, t_9, t_{10}\}|$ (support = 6 ≥ 3 = α). As it is a valid itemset, we compute the closure: $closure(A) = t_2 \cap t_3 \cap t_5 \cap t_6 \cap t_9 \cap t_{10} = A$. As closure(A) equals the itemset A itself, we conclude that A is a closed itemset.
Next, we consider all extensions from A: 1.1) cover(AB) = 5 ≥ 3 ⇒ $closure(AB) = t_2 \cap t_3 \cap t_5 \cap t_9 \cap t_{10} = AB$ ⇒ AB is a closed itemset.
If closure(itemset) ≠ itemset ⇒ add closure(itemset) as new closed itemset and move on to next case.
Next, we consider the extensions from AB: 1.1.1) cover(ABC) = 2 < 3 ✗
1.1.2) cover(ABD) = 1 < 3 ✗ 1.1.3) cover(ABE) = 2 < 3 ✗ 1.1.4) cover(ABF) = 3 ≥ 3 ⇒ closure(ABF) = ABF ⇒ ABF is a closed itemset. Now we would try with adding to ABF but we just saw that adding C, D, E to A, B leads to pruning.
1.2) cover(AC) .... At the end, you reach the item F in step 5) and after checking if he is a closed itemset you won't be able to add more items because he is the last item. The algorithm finishes, you can present the results and don't forget the empty set (also closed itemset).

FREQRARE: $\alpha_A = 4$; $\alpha_B = 3$; $\alpha_C = 3$; $\alpha_D = 1$

Trans	Items			
t1	A	B		D
t2	A		C	D
t3	A	B	C	D
t4		B	C	
t5	A	B	C	

$freq(AB) = 3 \geq \min\{\alpha_A, \alpha_B\} = \min\{4,3\} = 3$ — AB is frequent
$freq(ABC) = 2 < \min\{\alpha_A, \alpha_B, \alpha_C\} = 3$ ⇒ ABC not frequent
$freq(ABCD) = 1 \geq \min\{4,3,3,1\}$ ⇒ ABCD frequent

CLOSED PATTERN PROPAGATION RULES:
- **Rule 2**: If an item can join the partial pattern without dropping below α, we force it in ($X_i = 1$) (If item i decreases frequency under α, we remove 1 from dom(P_j))
- **Rule 1**: If adding an item i would drop below α, we force it out ($X_i = 0$) (If item i is an extension we remove 0 from dom(P_j))
- **Rule 3**: If item i never appears without some item j, and j is already absent, then i must also be absent too.



CP4IM: m auxiliary variables and (2n+m) reified constraints of arity (n+1) and (m+1)

Generic CP: Large number of Auxiliary variables and High number of reified constraints with significant arity.

CP4IM Constraints Example: I = {a, b, c, d}

Trans	Items			
t1	a	b		
t2	a		c	
t3	a	b	c	
t4		b		d
t5		b	c	d

- **Itemset Variables**: For each item $a_i \in I$, let $X_i \in \{0,1\}$ indicate whether a_i is included in the chosen itemset X.
- **Transaction variables**: For each transaction $t \in D$, let $T_t \in \{0,1\}$ indicate whether transaction t covers the chosen itemset X.

We also define a helper function $cover(t, a_i) = \begin{cases} 1 & \text{if item } a_i \text{ is in } t \\ 0 & \text{otherwise} \end{cases}$

1) Cover Enforcement: $\forall t \in D: (T_t = 1) \Leftrightarrow \sum_{a_i \in I} X_i(1 - cover(T_t, a_i)) = 0 \Leftrightarrow X_a(1-cover(T_t,a)) + X_b(1-cover(T_t,b)) + X_c(1-cover(T_t,c)) + X_d(1-cover(T_t,d)) = 0 \; \forall t \in D$ ⇝ For t = 1, $X_c + X_d = 0$ because $cover(T_1, a) = 1$, $cover(T_1, b) = 1$
2) Minimum Frequency: $\sum_{t \in D} T_t \geq \alpha$ ⇝ For α = 2, $T_1 + T_2 + T_3 + T_4 + T_5 \geq 2$
3) Closedness: $\forall a_i \in I: (X_i = 1) \Leftrightarrow \sum_{t \in D} T_t \cdot cover(T_t, a_i) = \sum_{t \in D} T_t$ ⇝ $\sum_{t=1}^{5} T_t \cdot cover(T_t, a_i) = \sum_{t=1}^{5} T_t \; \forall a_i \in I$ with $X_i = 1$
4) Maximality: $\forall a_i \in I$ with $X_i = 0$: $\sum_{t \in D} T_t \cdot cover(T_t, a_i) < \alpha$ ⇝ $\sum_{t=1}^{5} T_t \cdot cover(T_t, a_i) < \alpha$