# Reguły asocjacyjne, algorytm „Apriori”

(przykład dla zbioru „Sodowrażliwość”)

Przemysław Klęsk
*pklesk@wi.zut.edu.pl*

Zakład Sztucznej Inteligencji
Wydział Informatyki, ZUT

# Reguły asocjacyjne

## Zastosowania

- Analiza koszykowa (hipermarkety, sklepy internetowe):

$$\textit{pieluszki} \rightarrow \textit{piwo i chipsy}$$

  - rozstawianie towarów w sklepie,
  - projektowanie układu katalogów,
  - strategie cenowe na łączenia produktów.

- Bioinformatyka:
  - asocjacje genotyp → fenotyp,
  - częste sekwencje/wzorce w DNA (*DNA frequent motifs*),
  - reguły wiązania białek-DNA (*protein-DNA binding*).

# Algorytm „Apriori” (Agrawal, 1993)

## Elementy i pojęcia

- **Atrybuty binarne**

$$\{a_1, a_2, \ldots, a_n\}$$

(np. artykuły lub kategorie artykułów w sklepie, genotypy, fenotypy).

- **Zbiór danych** $D = \{d_i\}$, np.:

$$d_1 : a_2, a_3, a_5, a_9, a_{12}$$
$$d_2 : a_1, a_7$$
$$d_3 : a_3, a_5, a_6$$
$$\vdots$$

- **Zbiór przedmiotów** (ang. *itemset*) — dowolny podzbiór zbioru atrybutów.

# Algorytm „Apriori” (Agrawal, 1993)

## Elementy i pojęcia

- **Wsparcie zbioru** (ang. *support of itemset*):

$$\begin{aligned} \text{supp}(A) &= \frac{\text{liczba rekordów zawierających } A}{\text{liczba wszystkich rekordów}} \\ &= \frac{\#\{d_i \in D\colon A \subset d_i\}}{\#D} \\ &\approx P(A). \end{aligned} \tag{1}$$

- **Zaufanie reguły** (ang. *confidence of rule*):

$$\begin{aligned} \text{conf}(A \rightarrow B) &= \frac{\text{liczba rekordów zawierających } A \cup B}{\text{liczba rekordów zawierających } A} \\ &= \frac{\#\{d_i \in D\colon (A \cup B) \subset d_i\}}{\#\{d_i \in D\colon A \subset d_i\}} \\ &\approx P(B|A). \end{aligned} \tag{2}$$

# Algorytm „Apriori” (Agrawal, 1993)

## Elementy i pojęcia

- **Zbiór częsty** (ang. *frequent itemset*) — dla zadanej liczby $minSupp > 0$, mówimy, że $A$ jest zbiorem częstym, wtedy i tylko wtedy, gdy

$$\text{supp}(A) \geqslant minSupp.$$

## Monotoniczność miary wsparcia

1. Każdy podzbiór zbioru częstego jest częsty.
2. Każdy nadzbiór zbioru nieczęstego jest nieczęsty. **(!)**

# Algorytm „Apriori” (Agrawal, 1993)

## Dwie główne części

1. **Znajdowanie zbiorów częstych**
   (dla $D$ i zadanego *minSupp*)

2. **Generowanie reguł asocjacyjnych**
   (na zbiorach częstych dla zadanego *minConf*)

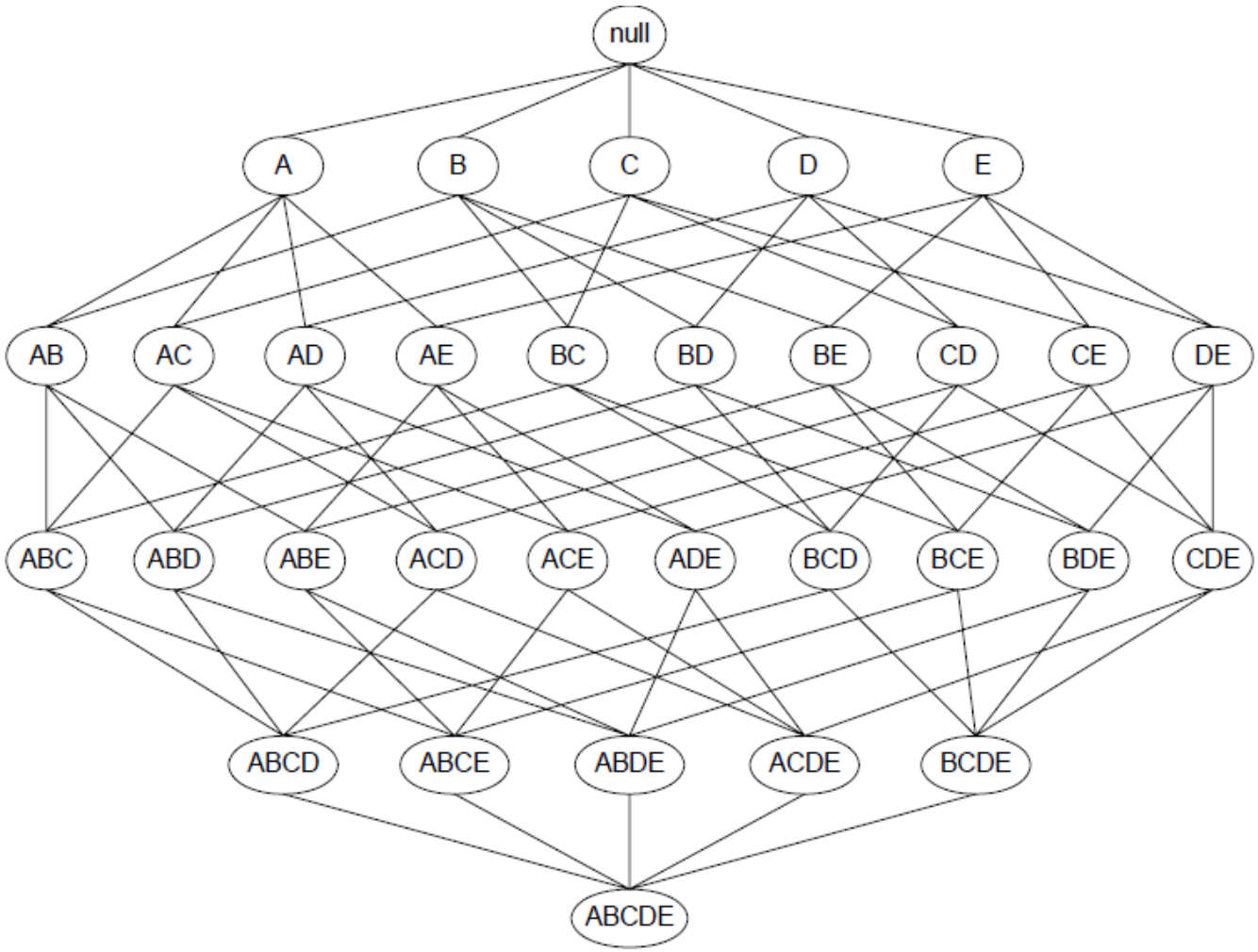
# Znajdowanie zbiorów częstych — część 1

$F_k$ — rodzina zbiorów częstych $k$-elementowych.
$C_k$ — rodzina kandydatów na zbiory częste $k$-elementowe.

## Algorytm

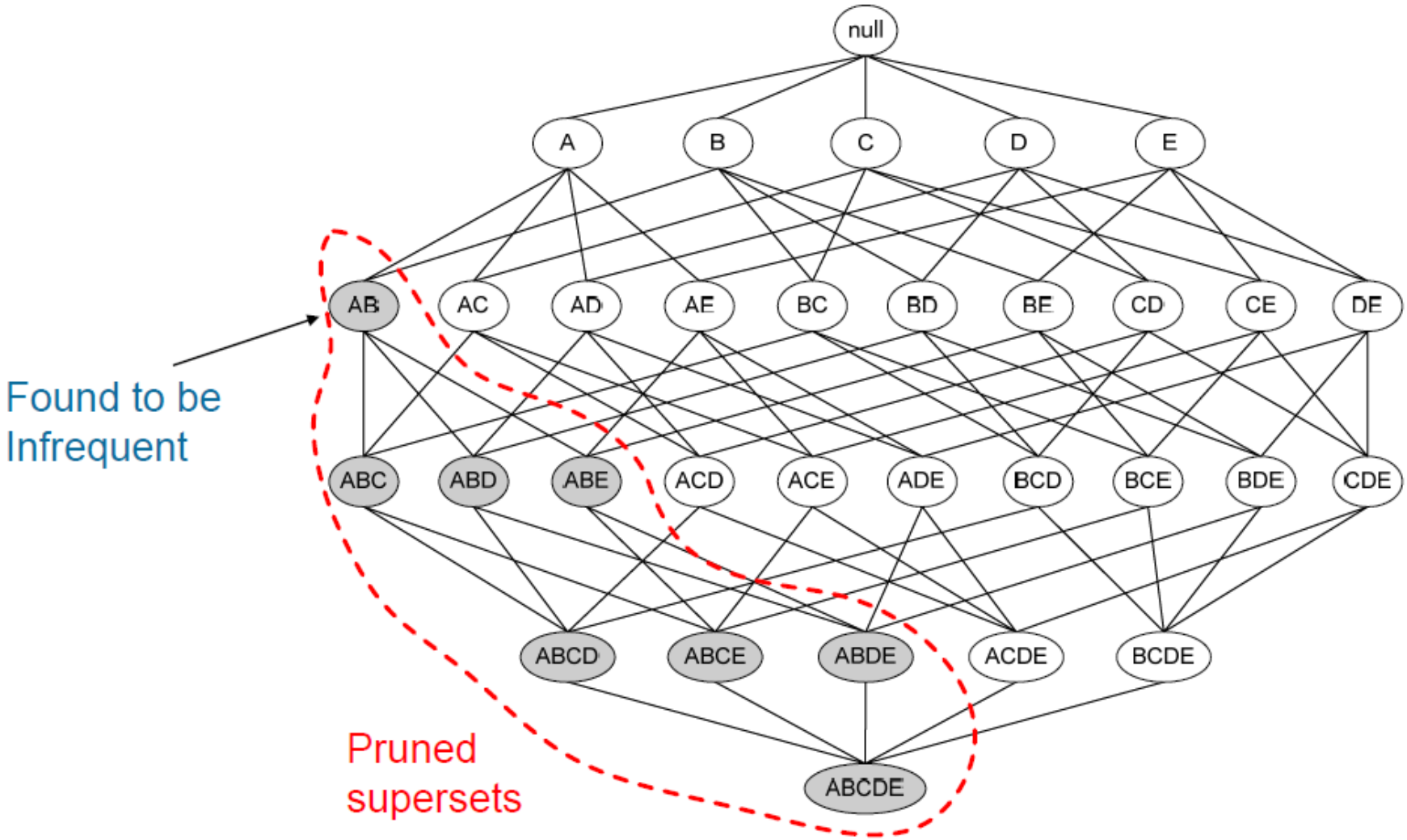
1. Zlicz wsparcia w $D$ dla zbiorów 1-elementowych: $\{a_1\}, \ldots, \{a_n\}$.
2. Zapamiętaj w $F_1$ te zbiory 1-elementowe, które spełniają *minSupp*.
3. $k := 1$.
4. Dopóki $\#F_k \geqslant k+1$ powtarzaj:
   1. Utwórz zbiór kandydatów $C_{k+1} := F_k \times F_k$. *(łączenie)*
   2. Usuń z $C_{k+1}$ tych kandydatów, którzy zawierają w sobie nieczęsty podzbiór o rozmiarze $k$ *(usuwanie poprzez monotoniczność)*
   3. Zlicz wsparcia w $D$ kandydatom. *(koszt rzędu $\#D \cdot \#C_{k+1}$)*
   4. Usuń z $C_{k+1}$ tych kandydatów, którzy zawierają nie spełniają *minSupp*. *(usuwanie na podstawie wsparć)*
   5. $F_{k+1} := C_{k+1}$.
   6. $k := k+1$.
5. Jeżeli $\#F_k = 0$ to $k := k-1$.
6. Zwróć wszystkie zbiory częste: $\bigcup_{i=1}^{k} F_i$.

# Krata zbiorów w algorytmie „Apriori”




Dla $n$ atrybutów liczba wszystkich możliwych zbiorów: $2^n$.

# Krata — pomijanie nieczęstych nadzbiorów



# Generowanie reguł — część 2

## Algorytm

1. Dla każdego $k \geqslant 2$: *(dla których $F_k$ jest określone)*
   1. Dla każdego $f \in F_k$:
      1. Dla każdego $g \subset f$, gdzie $g \neq \emptyset$ i $g \neq f$:
         *(podzbiory niepuste i różne od całego $f$)*

         Zbuduj regułę $g \rightarrow f \setminus g$,
         i jeżeli jej zaufanie

$$\mathrm{conf}(g \rightarrow f \setminus g) = \frac{\mathrm{supp}(g \cup f \setminus g)}{\mathrm{supp}(g)} = \frac{\mathrm{supp}(f)}{\mathrm{supp}(g)} \geqslant minConf, \qquad (3)$$

         *(wykorzystujemy wsparcia zliczone w części pierwszej)*

         to zapamiętaj regułę.
2. Zwróć zapamiętane reguły.

# Monotoniczność miary zaufania

## Obserwacja

Jeżeli $\mathrm{conf}(A \rightarrow B, C) \geqslant minConf$, to także:

$$\mathrm{conf}(A, B \rightarrow C) \geqslant minConf, \tag{4}$$

$$\mathrm{conf}(A, C \rightarrow B) \geqslant minConf. \tag{5}$$

Sprawdzenie:

$$\mathrm{conf}(A \rightarrow B, C) = \frac{\mathrm{supp}(A \cup B \cup C)}{\mathrm{supp}(A)} \tag{6}$$

$$\leqslant \frac{\mathrm{supp}(A \cup B \cup C)}{\mathrm{supp}(A \cup B)} = \mathrm{conf}(A, B \rightarrow C). \tag{7}$$

# Przykład do „ręcznego” policzenia

- Dane:

$$
\begin{aligned}
d_1 &: 1,2 \\
d_2 &: 1,3,5 \\
d_3 &: 4 \\
d_4 &: 1,2,4,5 \\
d_5 &: 2 \\
d_6 &: 2,3 \\
d_7 &: 3,4 \\
d_8 &: 1,2,3,5 \\
d_9 &: 1,4,5 \\
d_{10} &: 1,3,4
\end{aligned}
$$

- Znaleźć wszystkie reguły asocjacyjne dla $minSupp = \frac{2}{10}$ i $minConf = \frac{6}{10}$.

# Sodowrażliwość

- Istnieją osoby, które po spożyciu sodu (m.in. sól kuchenna, benzoesan sodu) doświadczają skoków ciśnienia tętniczego. Pojawia się to także u osób, które nie chorują nominalnie na nadciśnienie.
- Pomorska Akademia Medyczna (obecnie: PUM) zebrała zbiór danych na grupie 106 osób (bez i z sodowrażliwością).
- Eksperyment trwał 3 tygodnie: tydzień na diecie bezsolnej, tydzień na diecie solnej, tydzień na diecie mieszanej.
- Zbiór danych zawiera: 106 przykładów i 24 atrybuty (wśród atrybutów wejściowych jest 19 genetycznych i 4 inne).

**(!) Własność: Zakład Biochemii Klinicznej i Molekularnej, Pomorski Uniwersytet Medyczny w Szczecinie (prof. A. Ciechanowicz).**

# Sodowrażliwość — atrybuty

1. płeć — $\{F, M\}$,
2. wiek — $\{< 39.5, \geqslant 39.5\}$,
3. BMI — Body Mass Index $\{< 23.45, \geqslant 23.45\}$,
4. NT — nadciśnienie tętnicze $\{0, 1\}$,
5. dSS — wskaźnik sodowrażliwości (atrybut decyzyjny) $\{< 8, \geqslant 8\}$,
6. PROK — $\{?, AA, AB, AH, AI, AK, AQ, AR, BB, BH, BI, BK, HI, HK, HQ, IK\}$,
7. GSL — $\{?, CC, CT, TT\}$,
8. BE16 — $\{AA, AG, GG\}$,
9. BE27 — $\{CC, GC, GG\}$,
10. BE1 — $\{?, CC, CG, GG\}$,
11. G3NB — $\{CC, CT, TT\}$,
12. ACE — $\{DD, ID, II\}$,
13. HPA — $\{WM, WW\}$,
14. SYAL — $\{CC, TC, TT\}$,
15. ESC — $\{CC, CG, GG\}$,
16. ADD — $\{GG, GT, TT\}$,
17. AT1R — $\{AA, AC, CC\}$,
18. ATG — $\{AA, AG, GG\}$,
19. KAL1 — $\{CC, GC, GG\}$,
20. KAL3 — $\{GA, GG\}$,
21. KAL4 — $\{AA, AG, GG\}$,
22. KAL5 — $\{AA, AC, CC\}$,
23. eNOS — $\{GG, GT, TT\}$.

# „Sodowrażliwość” — binaryzacja atrybutów

| | | |
|---|---|---|
| płeć $\in \{F, M\}$ | $\rightarrow$ | dwa atrybuty binarne:<br>(płeć $= F$) $\in \{0, 1\}$,<br>(płeć $= M$) $\in \{0, 1\}$ |
| GSL $\in \{?, CC, CT, TT\}$ | $\rightarrow$ | cztery atrybuty binarne:<br>(GSL $= ?$) $\in \{0, 1\}$,<br>(GSL $= CC$) $\in \{0, 1\}$,<br>(GSL $= CT$) $\in \{0, 1\}$,<br>(GSL $= TT$) $\in \{0, 1\}$ |
| $\vdots$ | $\vdots$ | $\vdots$ |

Ostatecznie razem powstało 80 atrybutów binarnych.

# Przebieg indukcji dla *minSupp* = 30/106

```
>> tic; [F, supports] = findFrequentItemSets(transactions, 30 / 106); toc;
-----------------------------------------------------------------------------------------------------
induction at step: (k = 1) -> (k + 1 = 2)
candidates generated, #C_2 = 3160
counting supports for candidates...
candidates with insufficient support eliminated, F_2 := C_2, #F_2 = 273
-----------------------------------------------------------------------------------------------------
induction at step: (k = 2) -> (k + 1 = 3)
candidates generated, #C_3 = 3012
candidates with non-frequent subsets eliminated, #C_3 = 1261
counting supports for candidates...
candidates with insufficient support eliminated, F_3 := C_3, #F_3 = 470
-----------------------------------------------------------------------------------------------------
induction at step: (k = 3) -> (k + 1 = 4)
candidates generated, #C_4 = 3664
candidates with non-frequent subsets eliminated, #C_4 = 430
counting supports for candidates...
candidates with insufficient support eliminated, F_4 := C_4, #F_4 = 253
-----------------------------------------------------------------------------------------------------
induction at step: (k = 4) -> (k + 1 = 5)
candidates generated, #C_5 = 1300
candidates with non-frequent subsets eliminated, #C_5 = 59
counting supports for candidates...
candidates with insufficient support eliminated, F_5 := C_5, #F_5 = 38
-----------------------------------------------------------------------------------------------------
induction at step: (k = 5) -> (k + 1 = 6)
candidates generated, #C_6 = 106
candidates with non-frequent subsets eliminated, #C_6 = 0
counting supports for candidates...
candidates with insufficient support eliminated, F_6 := C_6, #F_6 = 0
Elapsed time is 100.964000 seconds.
```

# Reguły asocjacyjne dla *minConf* = 1

```
>> tic; [rulesPremises, rulesConsequences, rulesConfidences] = findRules(F, supports, 1.0,
numbersToItemsMapper); toc;
r1: IF KAL5=AA THEN KAL3=GG.  supp(A)=44, supp(A,B)=44, conf(A->B)=1.
r2: IF KAL4=AA THEN KAL3=GG.  supp(A)=44, supp(A,B)=44, conf(A->B)=1.
r3: IF SCA=WM THEN KAL3=GG.  supp(A)=31, supp(A,B)=31, conf(A->B)=1.
r4: IF SYAL=TC AND ADD=GG THEN KAL3=GG.  supp(A)=43, supp(A,B)=43, conf(A->B)=1.
r5: IF płeć=M AND G3NB=CT THEN HPA=WW.  supp(A)=30, supp(A,B)=30, conf(A->B)=1.
r6: IF płeć=M AND BMI=>=23.45 THEN HPA=WW.  supp(A)=30, supp(A,B)=30, conf(A->B)=1.
r7: IF ESC=CC AND KAL5=AC THEN KAL4=AG.  supp(A)=33, supp(A,B)=33, conf(A->B)=1.
r8: IF SYAL=TC AND ESC=CC THEN KAL3=GG.  supp(A)=33, supp(A,B)=33, conf(A->B)=1.
r9: IF KAL4=AA AND KAL5=AA THEN KAL3=GG.  supp(A)=43, supp(A,B)=43, conf(A->B)=1.
r10: IF ADD=GG AND KAL5=AA THEN KAL3=GG.  supp(A)=30, supp(A,B)=30, conf(A->B)=1.
r11: IF ADD=GG AND KAL4=AA THEN KAL3=GG.  supp(A)=30, supp(A,B)=30, conf(A->B)=1.
r12: IF płeć=M AND SYAL=TC THEN KAL3=GG.  supp(A)=39, supp(A,B)=39, conf(A->B)=1.
r13: IF ADD=GG AND KAL1=GC THEN KAL3=GG.  supp(A)=33, supp(A,B)=33, conf(A->B)=1.
r14: IF HPA=WW AND KAL5=AA THEN KAL3=GG.  supp(A)=37, supp(A,B)=37, conf(A->B)=1.
r15: IF G3NB=CT AND ADD=GG THEN KAL3=GG.  supp(A)=30, supp(A,B)=30, conf(A->B)=1.
r16: IF SYAL=TC AND AT1R=AA THEN KAL3=GG.  supp(A)=31, supp(A,B)=31, conf(A->B)=1.
r17: IF G3NB=CC AND SYAL=TC THEN KAL3=GG.  supp(A)=30, supp(A,B)=30, conf(A->B)=1.
r18: IF wiek=<39.5 AND KAL5=AA THEN KAL4=AA.  supp(A)=34, supp(A,B)=34, conf(A->B)=1.
r19: IF wiek=<39.5 AND KAL4=AA THEN KAL3=GG.  supp(A)=35, supp(A,B)=35, conf(A->B)=1.
r20: IF SYAL=TC AND ATG=AG THEN KAL3=GG.  supp(A)=36, supp(A,B)=36, conf(A->B)=1.
r21: IF dSS=>=8 AND SYAL=TC THEN KAL3=GG.  supp(A)=33, supp(A,B)=33, conf(A->B)=1.
r22: IF HPA=WW AND KAL5=AA THEN KAL4=AA.  supp(A)=37, supp(A,B)=37, conf(A->B)=1.
r23: IF płeć=M AND G3NB=CT THEN KAL3=GG.  supp(A)=30, supp(A,B)=30, conf(A->B)=1.
r24: IF ATG=AG AND KAL5=AC THEN KAL4=AG.  supp(A)=30, supp(A,B)=30, conf(A->B)=1.
r25: IF BE1=CC AND KAL5=AC THEN KAL4=AG.  supp(A)=38, supp(A,B)=38, conf(A->B)=1.
r26: IF BE1=CC AND KAL1=GC THEN KAL3=GG.  supp(A)=32, supp(A,B)=32, conf(A->B)=1.
r27: IF wiek=<39.5 AND KAL5=AA THEN KAL3=GG.  supp(A)=34, supp(A,B)=34, conf(A->B)=1.
r28: IF HPA=WW AND KAL4=AA THEN KAL3=GG.  supp(A)=38, supp(A,B)=38, conf(A->B)=1.
r29: IF HPA=WW AND KAL5=AA THEN KAL3=GG AND KAL4=AA.  supp(A)=37, supp(A,B)=37, conf(A->B)=1.
...
```

# Reguły asocjacyjne (c.d.) dla *minConf* = 1

```
r30: IF płeć=M AND G3NB=CT THEN HPA=WW AND KAL3=GG.  supp(A)=30, supp(A,B)=30, conf(A->B)=1.
r31: IF wiek=<39.5 AND KAL5=AA THEN KAL3=GG AND KAL4=AA.  supp(A)=34, supp(A,B)=34, conf(A->B)=1.
r32: IF płeć=M AND HPA=WW AND SYAL=TC THEN KAL3=GG.  supp(A)=36, supp(A,B)=36, conf(A->B)=1.
r33: IF BE1=CC AND ADD=GG AND KAL4=AG THEN KAL5=AC.  supp(A)=31, supp(A,B)=31, conf(A->B)=1.
r34: IF BE1=CC AND ADD=GG AND KAL5=AC THEN KAL4=AG.  supp(A)=31, supp(A,B)=31, conf(A->B)=1.
r35: IF wiek=<39.5 AND BE1=CC AND KAL5=AC THEN KAL4=AG.  supp(A)=32, supp(A,B)=32, conf(A->B)=1.
r36: IF HPA=WW AND KAL3=GG AND KAL5=AA THEN KAL4=AA.  supp(A)=37, supp(A,B)=37, conf(A->B)=1.
r37: IF HPA=WW AND KAL4=AA AND KAL5=AA THEN KAL3=GG.  supp(A)=37, supp(A,B)=37, conf(A->B)=1.
r38: IF BE1=CC AND KAL3=GG AND KAL4=AG THEN KAL5=AC.  supp(A)=37, supp(A,B)=37, conf(A->B)=1.
r39: IF BE1=CC AND KAL3=GG AND KAL5=AC THEN KAL4=AG.  supp(A)=37, supp(A,B)=37, conf(A->B)=1.
r40: IF wiek=<39.5 AND SYAL=TC AND ADD=GG THEN KAL3=GG.  supp(A)=35, supp(A,B)=35, conf(A->B)=1.
r41: IF płeć=M AND G3NB=CT AND HPA=WW THEN KAL3=GG.  supp(A)=30, supp(A,B)=30, conf(A->B)=1.
r42: IF płeć=M AND G3NB=CT AND KAL3=GG THEN HPA=WW.  supp(A)=30, supp(A,B)=30, conf(A->B)=1.
r43: IF wiek=<39.5 AND KAL3=GG AND KAL5=AA THEN KAL4=AA.  supp(A)=34, supp(A,B)=34, conf(A->B)=1.
r44: IF wiek=<39.5 AND KAL4=AA AND KAL5=AA THEN KAL3=GG.  supp(A)=34, supp(A,B)=34, conf(A->B)=1.
r45: IF BE16=AG AND KAL3=GG AND KAL4=AG THEN KAL5=AC.  supp(A)=30, supp(A,B)=30, conf(A->B)=1.
r46: IF płeć=M AND wiek=<39.5 AND SYAL=TC THEN KAL3=GG.  supp(A)=33, supp(A,B)=33, conf(A->B)=1.
r47: IF HPA=WW AND SYAL=TC AND ATG=AG THEN KAL3=GG.  supp(A)=33, supp(A,B)=33, conf(A->B)=1.
r48: IF BE1=CC AND HPA=WW AND KAL5=AC THEN KAL4=AG.  supp(A)=35, supp(A,B)=35, conf(A->B)=1.
r49: IF HPA=WW AND ESC=CC AND KAL5=AC THEN KAL4=AG.  supp(A)=30, supp(A,B)=30, conf(A->B)=1.
r50: IF dSS=>=8 AND HPA=WW AND SYAL=TC THEN KAL3=GG.  supp(A)=30, supp(A,B)=30, conf(A->B)=1.
r51: IF HPA=WW AND SYAL=TC AND ADD=GG THEN KAL3=GG.  supp(A)=37, supp(A,B)=37, conf(A->B)=1.
r52: IF ESC=CC AND KAL3=GG AND KAL5=AC THEN KAL4=AG.  supp(A)=31, supp(A,B)=31, conf(A->B)=1.
r53: IF wiek=<39.5 AND BE1=CC AND KAL3=GG AND KAL4=AG THEN KAL5=AC.  supp(A)=31, supp(A,B)=31, conf(A->B)=1
r54: IF wiek=<39.5 AND BE1=CC AND KAL3=GG AND KAL5=AC THEN KAL4=AG.  supp(A)=31, supp(A,B)=31, conf(A->B)=1
r55: IF płeć=M AND wiek=<39.5 AND HPA=WW AND SYAL=TC THEN KAL3=GG.  supp(A)=30, supp(A,B)=30, conf(A->B)=1.
r56: IF BE1=CC AND HPA=WW AND KAL3=GG AND KAL4=AG THEN KAL5=AC.  supp(A)=34, supp(A,B)=34, conf(A->B)=1.
r57: IF BE1=CC AND HPA=WW AND KAL3=GG AND KAL5=AC THEN KAL4=AG.  supp(A)=34, supp(A,B)=34, conf(A->B)=1.
r58: IF BE1=CC AND ADD=GG AND KAL3=GG AND KAL4=AG THEN KAL5=AC.  supp(A)=30, supp(A,B)=30, conf(A->B)=1.
r59: IF BE1=CC AND ADD=GG AND KAL3=GG AND KAL5=AC THEN KAL4=AG.  supp(A)=30, supp(A,B)=30, conf(A->B)=1.
Elapsed time is 3.542000 seconds.
```

# Wybrane reguły dla $minConf \geqslant 0.9$

Dla $minConf = 0.95$ otrzymano reguł 333. Dla $minConf = 0.9$ otrzymano reguł 902.

- Reguły trywialne:

```
r2: IF BMI=<23.45 THEN wiek=<39.5. conf(A->B)=0.9322.
r12: IF NT=0 THEN wiek=<39.5. conf(A->B)=0.94737.
```

- Reguły „skutek → przyczyna” (?):

```
r56: IF płeć=M THEN HPA=WW AND KAL3=GG. conf(A->B)=0.92537.  % złożona konkluzja (rzadkie)
r66: IF płeć=M AND NT=1 THEN HPA=WW. conf(A->B)=0.95745.
r101: IF dSS=>=8 AND BE16=AG THEN KAL3=GG. conf(A->B)=0.96774.
r115: IF wiek=<39.5 AND dSS=>=8 THEN HPA=WW. conf(A->B)=0.91111.
r195: IF dSS=>=8 AND ESC=CC THEN HPA=WW. conf(A->B)=0.94595.
r210: IF NT=1 AND dSS=>=8 THEN KAL3=GG. conf(A->B)=0.93182.
r232: IF dSS=<8 AND BE1=CC THEN KAL3=GG. conf(A->B)=0.94444.
r295: IF dSS=>=8 AND ADD=GG THEN KAL3=GG. conf(A->B)=0.95745.
r392: IF dSS=>=8 AND KAL1=GG THEN KAL3=GG. conf(A->B)=0.94444.
r449: IF NT=0 AND HPA=WW THEN wiek=<39.5 AND KAL3=GG. conf(A->B)=0.91176.
r724: IF dSS=<8 AND BE1=CC AND HPA=WW THEN KAL3=GG. conf(A->B)=0.9375.
```

- Reguły „korelacje międzygenowe lub międzygenotypowe”:

```
r57: IF KAL4=AA THEN KAL3=GG AND KAL5=AA. conf(A->B)=0.97727.
r112: IF SCA=WW AND ESC=CC THEN HPA=WW. conf(A->B)=0.90244.
r826: IF BE1=CC AND ADD=GG AND KAL5=AC THEN KAL3=GG AND KAL4=AG. conf(A->B)=0.96774.
```

# Wybrane reguły dla $minConf \geqslant 0.7$

Otrzymano wszystkich reguł 2464.

- Reguły dla sodowrażliwości (posortowane wg *conf*):

```
r19: IF BMI=>=23.45 THEN dSS=>=8. conf(A->B)=0.70213.
r736: IF płeć=M AND ADD=GG THEN dSS=>=8. conf(A->B)=0.70213.
r486: IF płeć=M AND ESC=CC THEN dSS=>=8. conf(A->B)=0.7381.
r564: IF płeć=M AND NT=1 THEN dSS=>=8. conf(A->B)=0.76596.
r1769: IF płeć=M AND HPA=WW AND ESC=CC THEN dSS=>=8. conf(A->B)=0.76923.
r2343: IF płeć=M AND NT=1 AND HPA=WW AND KAL3=GG THEN dSS=>=8. conf(A->B)=0.76744.
r1920: IF płeć=M AND NT=1 AND HPA=WW THEN dSS=>=8. conf(A->B)=0.77778.
```

- Reguły dla nadciśnienia (posortowane wg *conf*):

```
r132: IF płeć=M THEN NT=1. conf(A->B)=0.70149.
r63: IF KAL1=GG THEN NT=1. conf(A->B)=0.7069.
r130: IF G3NB=CT THEN NT=1. conf(A->B)=0.73913.
r6: IF eNOS=GT THEN NT=1. conf(A->B)=0.7561.
r299: IF BE1=CC AND ESC=CC THEN NT=1. conf(A->B)=0.78049.
r75: IF BMI=>=23.45 THEN NT=1. conf(A->B)=0.78723.
r359: IF BMI=>=23.45 AND ADD=GG THEN NT=1. conf(A->B)=0.81081.
```

# Przebieg indukcji dla *minSupp* = 50/106

```
>> tic; [F, supports] = findFrequentItemSets(transactions, 50 / 106); toc;
-------------------------------------------------------------------------------------------------------
induction at step: (k = 1) -> (k + 1 = 2)
candidates generated, #C_2 = 3160
counting supports for candidates...
candidates with insufficient support eliminated, F_2 := C_2, #F_2 = 44
-------------------------------------------------------------------------------------------------------
induction at step: (k = 2) -> (k + 1 = 3)
candidates generated, #C_3 = 314
candidates with non-frequent subsets eliminated, #C_3 = 35
counting supports for candidates...
candidates with insufficient support eliminated, F_3 := C_3, #F_3 = 22
-------------------------------------------------------------------------------------------------------
induction at step: (k = 3) -> (k + 1 = 4)
candidates generated, #C_4 = 82
candidates with non-frequent subsets eliminated, #C_4 = 3
counting supports for candidates...
candidates with insufficient support eliminated, F_4 := C_4, #F_4 = 2
Elapsed time is 37.986000 seconds.
```

# Reguły asocjacyjne dla *minConf* = 0.95

Brak reguł dla *minConf* = 1.0.

```
r1: IF BE1=CC THEN KAL3=GG.  supp(A)=69, supp(A,B)=66, conf(A->B)=0.95652.
r2: IF AT1R=AA THEN KAL3=GG.  supp(A)=57, supp(A,B)=55, conf(A->B)=0.96491.
r3: IF ADD=GG THEN KAL3=GG.  supp(A)=77, supp(A,B)=74, conf(A->B)=0.96104.
r4: IF SYAL=TC THEN KAL3=GG.  supp(A)=60, supp(A,B)=59, conf(A->B)=0.98333.
r5: IF ESC=CC THEN KAL3=GG.  supp(A)=61, supp(A,B)=58, conf(A->B)=0.95082.
r6: IF płeć=M THEN KAL3=GG.  supp(A)=67, supp(A,B)=65, conf(A->B)=0.97015.
r7: IF wiek=<39.5 THEN KAL3=GG.  supp(A)=85, supp(A,B)=81, conf(A->B)=0.95294.
r8: IF płeć=M THEN HPA=WW.  supp(A)=67, supp(A,B)=64, conf(A->B)=0.95522.
r9: IF BE1=CC AND ADD=GG THEN KAL3=GG.  supp(A)=54, supp(A,B)=52, conf(A->B)=0.96296.
r10: IF HPA=WW AND ADD=GG THEN KAL3=GG.  supp(A)=68, supp(A,B)=65, conf(A->B)=0.95588.
r11: IF płeć=M AND wiek=<39.5 THEN KAL3=GG.  supp(A)=58, supp(A,B)=56, conf(A->B)=0.96552.
r12: IF płeć=M AND HPA=WW THEN KAL3=GG.  supp(A)=64, supp(A,B)=62, conf(A->B)=0.96875.
r13: IF płeć=M AND KAL3=GG THEN HPA=WW.  supp(A)=65, supp(A,B)=62, conf(A->B)=0.95385.
r14: IF wiek=<39.5 AND ADD=GG THEN KAL3=GG.  supp(A)=59, supp(A,B)=57, conf(A->B)=0.9661.
r15: IF BE1=CC AND HPA=WW THEN KAL3=GG.  supp(A)=62, supp(A,B)=59, conf(A->B)=0.95161.
r16: IF HPA=WW AND SYAL=TC THEN KAL3=GG.  supp(A)=53, supp(A,B)=52, conf(A->B)=0.98113.
r17: IF NT=1 AND ADD=GG THEN KAL3=GG.  supp(A)=52, supp(A,B)=50, conf(A->B)=0.96154.
r18: IF płeć=M AND wiek=<39.5 AND HPA=WW THEN KAL3=GG.  supp(A)=55, supp(A,B)=53, conf(A->B)=0.96364.
```

Wśród powyższych reguł, brak reguł dotyczących atrybutu *dSS* (fenotypu sodowrażliwości).

# Implementacja w MATLABie — wskazówki

## Przydatne funkcje

- `union` — suma zbiorów (podanych jako wektory),
- `intersect` — przecięcie zbiorów (część wspólna),
- `setdiff` — różnica zbiorów,
- `ismember` — dla podanych dwóch wektorów (lub macierzy) zwraca wektor (lub macierz) wyników $\{0, 1\}$ wskazujących, czy dany element pierwszego wektora (macierzy) jest elementem drugiego wektora (macierzy).
- `nchoosek` — (nazwa ma kojarzyć się z $\binom{n}{k}$) dla podanego wektora $v$ i liczby $k$ zwraca wszystkie $k$-elementowe kombinacje (czyli podzbiory) wektora $v$; wynik zwracany jest jako macierz, gdzie kolejne kombinacje pisane są wierszami,
- wyniki `union`, `intersect`, `setdiff` są posortowane leksykograficznie.

# Implementacja w MATLABie — wskazówki

## Mapy haszujące

- W MATLABie można korzystać ze struktur danych języka Java.
- Do szybkich sprawdzeń *„czy pewien podzbiór jest częsty?”* warto wykorzystać klasę `java.util.HashMap` — struktura przechowująca pary *(klucz, wartość)*, dająca szybkość kosztem pamięci. Wszystkie istotne operacje są o stałej złożoności $O(1)$.
- Zbiory częste można przechować np. jako wektor komórkowy map haszujących na zasadzie: `F{k}=java.util.HashMap;`
- Podstawowe metody:
  - `put(key, value)` — dodanie pary do mapy,
  - `get(key)` — pobranie wartości spod klucza,
  - `containsKey(key)` — flaga logiczna „czy klucz figuruje w mapie?”,
  - `remove(key)` — usunięcie wpisu (pary) spod klucza,
  - `keySet().toArray()` — zwraca zbiór kluczy jako tablicę (umożliwia przeiterowanie się po niej w MATLABowe pętli `for`).
  - `valueSet().toArray()` — zwraca zbiór wartości jako tablicę (możliwe iterowanie po niej w MATLABowe pętli `for`).
  - Dokumentacja (Java 6, 7):
    *http://docs.oracle.com/javase/6/docs/api/java/util/HashMap.html*
    *http://docs.oracle.com/javase/7/docs/api/java/util/HashMap.html*

# Implementacja w MATLABie — wskazówki

## Mapy haszujące c.d.

- „Pod spodem” znajduje się tablica o pewnym ustalonym rozmiarze początkowym.
- Każde włożenie, np. `H.put('toJestPewienKlucz', 7)`, realizowane jest poprzez wyznaczenie indeksu w tablicy, pod który ma trafić podana para, poprzez obliczenie funkcji haszującej na kluczu.
- Przykład funkcji haszującej dla napisu $s$:

$$s_0 32^{n-1} + s_1 31^{n-2} + \cdots + s_{n-2} 31^1 + s_{n-1} \qquad \text{mod rozmiar tablicy.} \tag{8}$$

- Operacja pobrania, np. `H.get('toJestPewienKlucz')`, lub sprawdzenia czy klucz istnieje, `H.containsKey('toJestPewienKlucz')`, także sprowadza się do wyznaczenia indeksu poprzez funkcję haszującą — wstrzelenie się od razu we właściwe miejsce.

# Implementacja w MATLABie — wskazówki

## Mapy haszujące c.d.

- Dwukrotne włożenie z użyciem tego samego klucza, powoduje nadpisanie starej pary poprzez nową.
- W razie konfliktów — gdy dwa różne klucze dają tę samą wartość funkcji haszującej — elementy tablicy zaczynają przechowywać **krótkie listy** wpisanych par (nie ma w tym przypadku nadpisania).
- Gdy zacznie być „za gęsto” — dużo krótkich list — mapa haszująca dynamicznie rozszerza się (np. podwaja swój rozmiar). Wymaga to alokacji nowej tablicy i przepisania starych wpisów (*rehashing*).
- A zatem większość pojedynczych operacji `put` ma koszt $O(1)$, ale co jakiś czas pewna operacja będzie o koszcie $O(n)$, gdzie $n$ to aktualny rozmiary mapy.
- Można pokazać, że zamortyzowany koszt przypadający średnio na każdy element ma nadal złożoność stałą $O(1)$.

# Mapy haszujące, zamortyzowany koszt put

Niech początkowy rozmiar wynosi 1, a współczynnik rozszerzania się mapy wynosi 2. Niech liczba elementów, które chcemy włożyć będzie przedstawialna jako $n = 2^m$.

Przykład chronologii *dopisywania* ($d$) i *przepisywania* ($p$) wkładanych elementów:

$$\underbrace{d}_{m=0} \,|\, \underbrace{p,d}_{m=1} \,|\, \underbrace{p,p,d,d}_{m=2} \,|\, \underbrace{p,p,p,p,d,d,d,d}_{m=3} \,|\cdots$$

Koszt włożenia $n$ elementów:

$$f(n) = 1 + \underbrace{\sum_{k=0}^{m-1} 2^k}_{\text{dopisania}} + \underbrace{\sum_{k=0}^{m-1} 2^k}_{\text{przepisania}} = 1 + \sum_{k=0}^{m-1} 2^{k+1} = 1 + \frac{2 - 2^{m+1}}{1-2} = 2^{m+1} - 1. \tag{9}$$

Zamortyzowany (średni) koszt włożenia elementu:

$$\frac{f(n)}{n} = \frac{f(2^m)}{2^m} = \frac{2^{m+1}-1}{2^m} = 2 - \frac{1}{2^m} \leqslant 2, \tag{10}$$

a zatem $O(f(n)/n) = O(1)$.

# Kombinatoryka

**Problem**

Dla $n$ atrybutów, ile jest wszystkich możliwych reguł asocjacyjnych?