# Traženje redeskripcija

Matej Mihelčić

Prirodoslovno-matematički fakultet, Sveučilište u Zagrebu

*matmih@math.hr*

27. svibnja, 2026.

# Traženje redeskripcija (višestrukih opisa)

- Problem traženja redeskripcija pripada skupu **nenadziranih** zadataka dubinske analize podataka.
- Glavni ciljevi zadatka traženja redeskripcija jesu: a) **pronaći skupove entiteta** koji se mogu opisati na **više načina** korištenjem logičkih pravila, b) **pronaći višestruke opise** skupova entiteta.
- Višestrukim opisom smatramo skup pravila koji u idealnom slučaju opisuju točno zadan skup entiteta (ili češće u praksi jako sličan zadanom).
- Pošto svako pravilo opisuje zadani skup entiteta, entiteti sadrže sva svojstva otkrivena u pravilima. Odnosno, otkrivena svojstva su u **relaciji ekvivalencije**.
- Otkrivene redeskripcije ($n$-torke pravila) pružaju opis skupine entiteta slično kao kod konceptualnog klasteriranja, međutim također otkrivaju pravila koja su međusobno u relaciji ekvivalencije, što je snažnija veza od implikacija koju otkrivaju asocijacijska pravila.

# Traženje redeskripcija

- Pristupi traženja redeskripcija kao ulaz koriste jednu, dvije, a neki i više tablica (pogleda) skupa podataka, stoga je zadatak povezan i sa zadatkom **višepoglednog** klasteriranja. Svaka tablica (pogled) opisuje jedan (međusobno identični) skup entiteta $E$, no sadrži međusobno disjunktni skup atributa.
- Otkrivanje relacija ekvivalencija nam pomaže u razumijevanju domenskih problema tako što: a) pronalazi niz svojstava vezanih uz određeni skup entiteta, procesa odnosno pojava (npr. svojstva grupe zemalja po pitanju trgovinskih obrazaca i socio-demografske strukture), b) omogućava povezivanje svojstava koja su poznata i dobro istražena s novim svojstvima koja još nisu dovoljno istražena (npr. povezivanje genetskih faktora s kliničkim).

# Traženje redeskripcija

**$P_1$**

| E | $A_1$ | $A_2$ | ... | $A_{k1}$ |
|---|---|---|---|---|
| $e_1$ | 1 | 0 | ... | 1 |
| $e_2$ | 0 | 0 | ... | 1 |
| . | . | . | . | . |
| . | . | . | . | . |
| . | . | . | . | . |
| $e_{N-1}$ | 1 | 1 | ... | 1 |
| $e_N$ | 1 | 1 | ... | 0 |

**$P_2$**

| E | $B_1$ | $B_2$ | ... | $B_{k2}$ |
|---|---|---|---|---|
| $e_1$ | 'b' | 1 | ... | 0 |
| $e_2$ | 'c' | 1 | ... | 0 |
| . | . | . | . | . |
| . | . | . | . | . |
| . | . | . | . | . |
| $e_{N-1}$ | 'b' | 0 | ... | 1 |
| $e_N$ | 'c' | 1 | ... | 1 |

. . .

**$P_s$**

| E | $H_1$ | $H_2$ | ... | $H_{ks}$ |
|---|---|---|---|---|
| $e_1$ | 5 | 1.4 | ... | 0.95 |
| $e_2$ | 8 | 12.5 | ... | 0.34 |
| . | . | . | . | . |
| . | . | . | . | . |
| . | . | . | . | . |
| $e_{N-1}$ | 9 | 10.3 | ... | 1.5 |
| $e_N$ | 1 | 13.5 | ... | 1.9 |

$$\underbrace{(A_1 \wedge A_5) \vee \neg (A_6 \wedge A_9)}_{p_1} \Leftrightarrow \underbrace{(B_{10} = 'g' \vee B_{20})}_{p_2} \Leftrightarrow \ldots \Leftrightarrow \underbrace{(0.67 \leqslant H_5 \leqslant 1.45 \wedge 5 \leqslant H_9 \leqslant 10 \wedge \neg (0.56 \leqslant H_{34} \leqslant 0.86)) \vee 10 \leqslant H_{77} \leqslant 14}_{p_s}$$

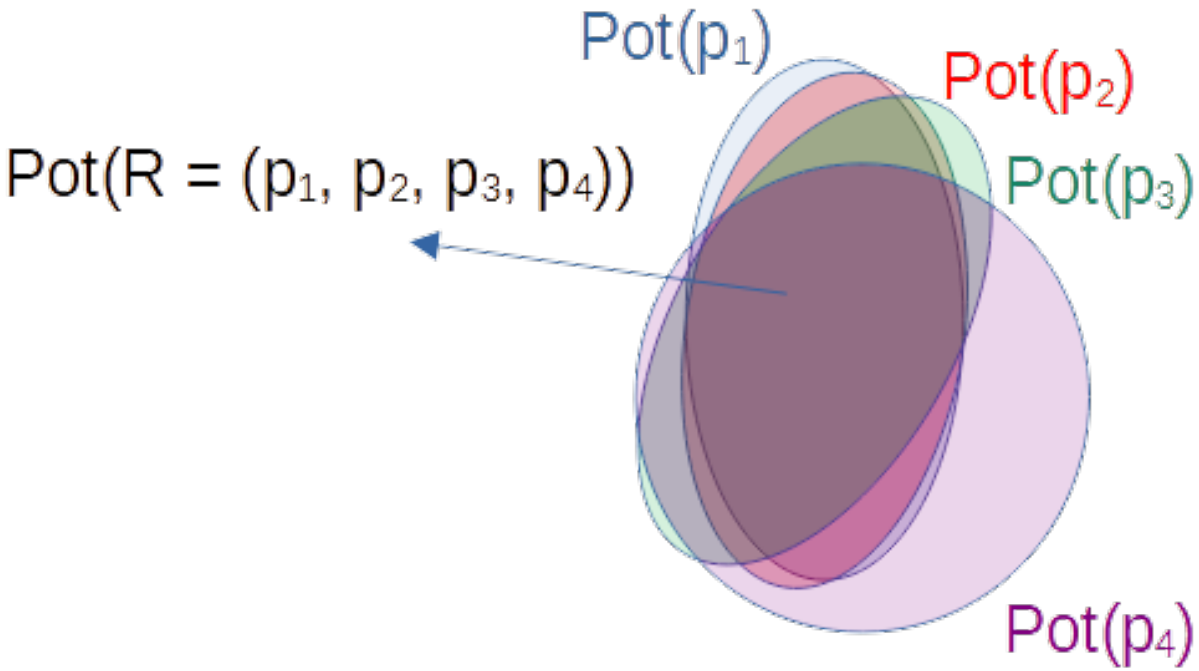
$$R = (p_1, p_2, \ldots, p_s)$$

# Traženje redeskripcija

- Ulazni podaci zadatka traženja redeskripcija se sastoje od $s$ pogleda $W_1, \ldots, W_s$ i $s$ odgovarajućih tablica $D_1, \ldots, D_s$. Svi pogledi opisuju jedan skup od $|E|$ entiteta, međutim svaka tablica $D_i$ sadrži atribute samo iz pogleda $W_i$.
- Pogled $W_i$ sadrži podskup atributa koji se nalazi u tablici $D_i$, tako da $W_i \cap W_j = \emptyset,\ i \neq j$.
- Redeskripcija $R$ je $s$-torka pravila $R = (p_1, \ldots, p_s)$, gdje je svaki $p_i$ logičko pravilo konstruirano na $D_i$ koristeći samo atribute iz $W_i$.
- **Skup potpore pravila** $p_i$, $Pot(p_i)$ je skup entiteta čiji atributi zadovoljavaju logičko pravilo $p_i$. Kardinalitet skupa potpore se zove **potpora pravila**.
- **Skup potpore redeskripcije** je skup entiteta čiji atributi zadovoljavaju logičko pravilo svih komponenta redeskripcija (dakle $p_1, \ldots, p_s$). $Pot(R) = \cap_{i=1}^{s} Pot(p_i)$. **Potpora redeskripcije** je kardinalitet skupa potpore redeskripcije.

# Traženje redeskripcija

- **Točnost redeskripcije** se mjeri kao Jaccard index između skupova potpora pravila od kojih se sastoji. $J(R) = \frac{|\cap_{i=1}^{s} Pot(p_i)|}{|\cup_{i=1}^{s} Pot(p_i)|}$.
- Intuitivno, ukoliko sva pravila $p_i$ koja čine redeskripciju opisuju identičan skup entiteta, tada je $J(R) = 1$. Ukoliko neko pravilo opisuje disjunktan skup entiteta od skupova entiteta opisanih od strane preostalih pravila, tada $J(R) = 0$.



# Traženje redeskripcija

- **Statistička značajnost redeskripcije** se računa kao

$$p(R = (p_1, \ldots, p_s)) = \sum_{k=|Pot(R)|}^{|E|} \binom{|E|}{k} (\prod_{i=1}^{s} p_i')^k \cdot (1 - \prod_{i=1}^{s} p_i')^{|E|-k}.$$

  $p_i' = \frac{|Pot(p_i)|}{|E|}$ predstavlja marginalnu vjerojatnost dobivanja pravila $p_i$.
- Statistička značajnost računa vjerojatnost dobivanja redeskripcije s jednakom ili većom potporom od redeskripcije $R$ s pravilima koje možemo dobiti s vjerojatnosti $p_i'$ **na slučajan način**. Pretpostavka je da primjeri dolaze iz uniformne distribucije, tada se vjerojatnost pojavljivanja pravila može aproksimirati kao kvocjent njegove potpore i broja elemenata u skupu podataka.
- Intuitivno, jednostavno je dobiti na slučajan način redeskripciju velike potpore i točnosti iz pravila koja imaju veliku potporu. Također, jednostavnije je na slučajan način dobiti redeskripcije lošije točnosti nego bolje točnosti.

# Traženje redeskripcija

- Za skup redeskripcija $\mathcal{R}$ i redeskripciju $R_i \in \mathcal{R}$, redundantnost redeskripcije s obzirom na entitete se definira kao:

$$PEJ(R_i, \mathcal{R}) = \frac{1}{|\mathcal{R}| - 1} \cdot \sum_{j=1}^{|\mathcal{R}|} J(Pot(R_i), Pot(R_j)), \ i \neq j.$$

- Za skup redeskripcija $\mathcal{R}$ i redeskripciju $R_i \in \mathcal{R}$, redundantnost redeskripcije s obzirom na atribute se definira kao:

$$PAJ(R_i, \mathcal{R}) = \frac{1}{|\mathcal{R}| - 1} \cdot \sum_{j=1}^{|\mathcal{R}|} J(attrs(R_i), attrs(R_j)), \ i \neq j, \text{ gdje}$$

$attrs(R_i)$ označava skup svih atributa koji se javljaju u svim pravilima redeskripcije $R_i$.

- Kompleksnost pravila redeskripcije se definira kao:

$$comp(R) = \begin{cases} \frac{|attr(R_i)|}{k} & |attr(R_i)| < k \\ 1 & k \leq |attr(R_i)| \end{cases}, \text{ gdje } attr(R_i) \text{ označava}$$

multiskup atributa koji se javljaju u svim pravilima redeskripcije $R_i$.

# Traženje redeskripcija - CARTwheels

- Algoritam radi isključivo nad podacima s binarnim atributima, koristeći dva pogleda.
- Neka je zadan skup podataka:

| E | $X_1$ | $X_2$ | $X_3$ | $X_4$ |
|---|---|---|---|---|
| $e_1$ | NE | NE | NE | DA |
| $e_2$ | DA | NE | DA | NE |
| $e_3$ | DA | DA | NE | NE |
| $e_4$ | NE | DA | DA | NE |
| $e_5$ | NE | NE | NE | DA |

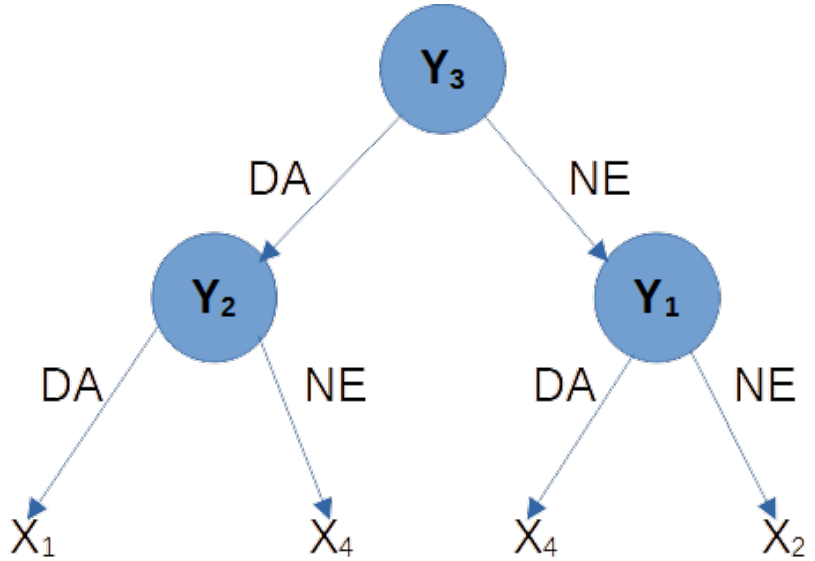
| E | $Y_1$ | $Y_2$ | $Y_3$ | $Y_4$ |
|---|---|---|---|---|
| $e_1$ | DA | NE | NE | DA |
| $e_2$ | DA | DA | NE | DA |
| $e_3$ | NE | DA | DA | NE |
| $e_4$ | NE | DA | NE | NE |
| $e_5$ | NE | NE | DA | DA |

- Definiramo novi skup podataka tako da atribute jedne od tablica (npr. druge) promatramo kao deskriptivne, dok iz atributa (npr prve) stvaramo ciljnu klasu.
- Klasu dodjeljujemo primjerima pohlepno, od $X_1$ do $X_4$.

# Traženje redeskripcija - CARTwheels

| E | $Y_1$ | $Y_2$ | $Y_3$ | $Y_4$ | klasa |
|---|---|---|---|---|---|
| $e_1$ | DA | NE | NE | DA | $X_4$ |
| $e_2$ | DA | DA | NE | DA | $X_1$ |
| $e_3$ | NE | DA | DA | NE | $X_1$ |
| $e_4$ | NE | DA | NE | NE | $X_2$ |
| $e_5$ | NE | NE | DA | DA | $X_4$ |

- Zatim stvaramo CART stablo odlučivanja na skupu podataka.



# Traženje redeskripcija - CARTwheels

- Sljedeći korak je stvoriti klase entiteta koristeći varijable $Y_i$. Tada možemo stvoriti novo stablo odlučivanja korištenjem $X_i$ kao deskriptivne atribute.
- Time dobijemo tablicu:

| E | $X_1$ | $X_2$ | $X_3$ | $X_4$ | klasa |
|---|---|---|---|---|---|
| $e_1$ | NE | NE | NE | DA | $(Y_3 \wedge \neg Y_2) \vee (Y_1 \wedge \neg Y_3)$ |
| $e_2$ | DA | NE | DA | NE | $(Y_3 \wedge \neg Y_2) \vee (Y_1 \wedge \neg Y_3)$ |
| $e_3$ | DA | DA | NE | NE | $(Y_3 \wedge Y_2)$ |
| $e_4$ | NE | DA | DA | NE | $(\neg Y_3 \wedge \neg Y_2)$ |
| $e_5$ | NE | NE | NE | DA | $(Y_3 \wedge \neg Y_2) \vee (Y_1 \wedge \neg Y_3)$ |

- Promatramo u kojem listu stabla se nalazi svaki $e_i$, te stvaramo klasu prema formuli koja odgovara putu ustablu od korjena do toga lista.

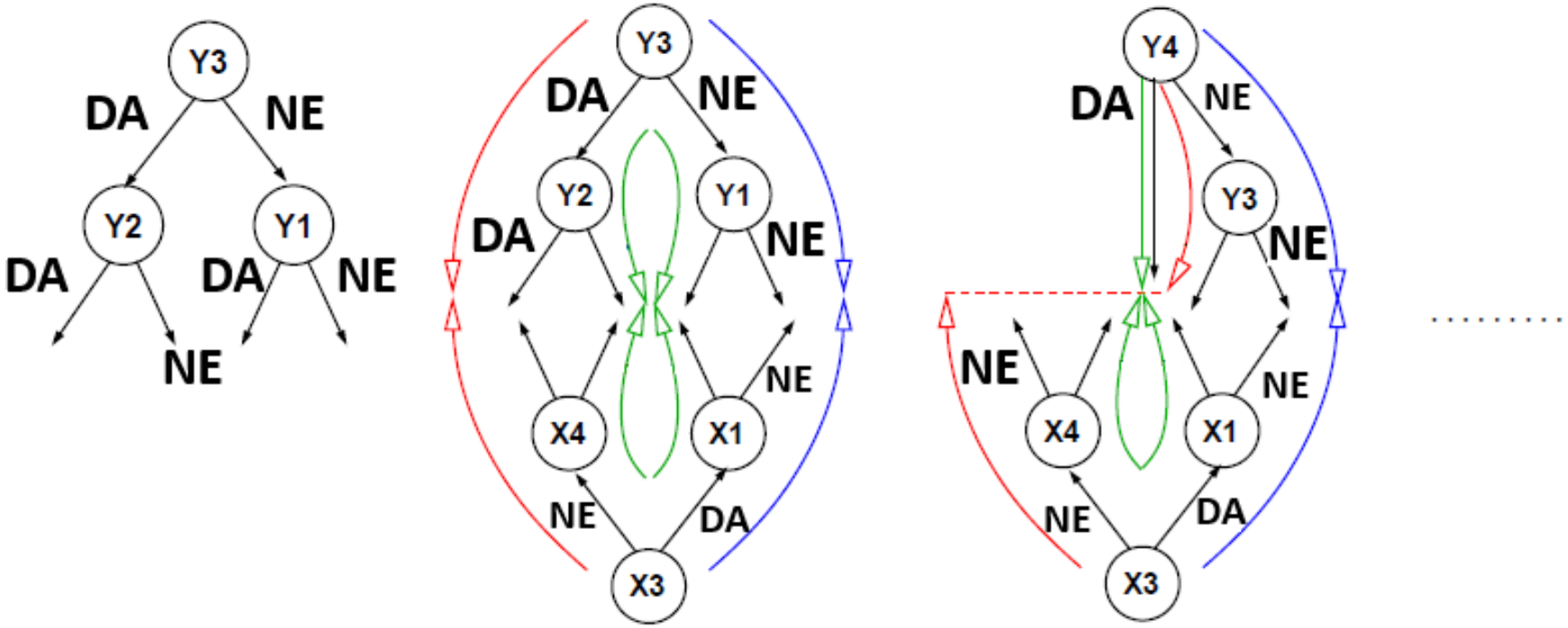
# Traženje redeskripcija - CARTwheels

- Dalje konstruiramo tablicu:

| E | $Y_1$ | $Y_2$ | $Y_3$ | $Y_4$ | klasa |
|---|---|---|---|---|---|
| $e_1$ | DA | NE | NE | DA | $(X_3 \wedge X_1) \vee (X_1 \wedge \neg X_3)$ |
| $e_2$ | DA | DA | NE | DA | $(X_3 \wedge X_1) \vee (X_1 \wedge \neg X_3)$ |
| $e_3$ | NE | DA | DA | NE | $(\neg X_3 \wedge \neg X_4)$ |
| $e_4$ | NE | DA | NE | NE | $(X_3 \wedge \neg X_1)$ |
| $e_5$ | NE | NE | DA | DA | $(X_3 \wedge X_1) \vee (X_1 \wedge \neg X_3)$ |

# Traženje redeskripcija - CARTwheels

- Iterativno stvaramo stabla, tako da u svakom koraku zadržimo stablo iz prethodnog koraka i prema njemu napravimo novo stablo.



# Traženje redeskripcija - CARTwheels

- Redeskripcije tražimo kao puteve u paru stabala koji kreću od korjena jednog stabla (npr. donjeg) i završavaju u korjenu drugog stabla (npr. gornjeg).
- Pošto stabla konstruiramo na takav način da imaju što sličnije entitete u svojim listovima, putevi nam nude alternativne opise (redeskripcije).

---

**Ulaz:** entiteti $E$, tablice $D_1$, $D_2$
**Izlaz:** redeskripcije $\mathcal{R}$
**Parametri:** $\theta$ (Prag Jaccard indeksa), $d$ (dubina stabla), $p$ (# klasa u čijoj definiciji smije sudjelovati svaki deskriptor), $\eta$ (maksimalan # dozvoljenih uzastopnih neuspješnih iteracija).
{**Inicijalizacija:**}
$\mathcal{R} \leftarrow \{\}$
$KP(A_i) \leftarrow 0,\ \forall A_i$ {# definicija klase u kojima atribut sudjeluje}
$\mathcal{F} \leftarrow attrs(D_2)$ {početne značajke skupa podataka za konstrukciju stabla odlučivanja}
$\mathcal{C} \leftarrow attrs(D_1)$ {početne klase skupa podataka za konstrukciju stabla odlučivanja}
$D \leftarrow gradi\_skup(E, \mathcal{F}, \mathcal{C})$
$t \leftarrow$ stvori_stablo$(D, d)$
**if** svi listovi u $t$ imaju istu klasu $c \in \mathcal{C}$ **then**
  $l \leftarrow$ slučajan list iz $t$ s entropijom različitom od nula
  dodijeli_drugu_najbolju_klasu_listu$(t, l)$
**end if**

# Traženje redeskripcija - CARTwheels

$\mathcal{C} \leftarrow$ putevi do klasa($t$)
zastavica = false; brojac = 0
{**Alternacija:**}
$A_1 \leftarrow attrs(D_1),\ A_2 \leftarrow attrs(D_2)$
$\mathcal{G} = A_1$
**while** brojac $< \eta$ **do**
  $\mathcal{F} \leftarrow \mathcal{G}$
  **if** zastavica = false **then**
    $A_1 \leftarrow \mathcal{G}$; $\mathcal{G} \leftarrow A_2$
  **else**
    $A_2 \leftarrow \mathcal{G}$; $\mathcal{G} \leftarrow A_1$
  **end if**
  $\mathcal{D} \leftarrow$ gradi_skup($E, \mathcal{F}, \mathcal{C}$)
  $t =$ stvori_stablo($\mathcal{D}, d$)
  **if** svi listovi u $t$ imaju istu klasu $c \in \mathcal{C}$ **then**
    $l \leftarrow$ slučajan list iz $t$ s entropijom različitom od nul
    dodijeli_drugu_najbolju_klasu_listu($t, l$)
  **end if**
  $\mathcal{R}_{novi} =$ eval($t, \theta$)
  {Nastavak na idućem slajdu}
**end while**

# Traženje redeskripcija - CARTwheels

**while** brojac $< \eta$ **do**
  {**Nastavak:**} . . .
  **if** $\mathcal{R}_{novi} = \{\}$ **then**
    brojac $\leftarrow$ brojac $+ 1$
  **else**
    brojac $\leftarrow 0$
    **for** each $c \in \mathcal{C}$ **do**
      **if** $c$ je uključen u stvaranje $R \in \mathcal{R}_{new}$ **then**
        $\mathcal{H} =$ atributi($c$)
        **for** svaki atribut $g \in \mathcal{G} \cap \mathcal{H}$ **do**
          povećaj brojac sudjelovanja u klasama od $g$
          **if** $g$ sudjeluje u viese od $> p$ klasa **then**
            brisi $g$ iz $\mathcal{G}$
          **end if**
        **end for**
      **end if**
    **end for**
  **end if**
  $\mathcal{R} = \mathcal{R} \cup \mathcal{R}_{novi}$; zastavica = not(zastavica)
  $\mathcal{C} =$ putevi_do_klasa($t$)
**end while**

# Traženje redeskripcija korištenjem frekventnih zatvorenih skupova objekata

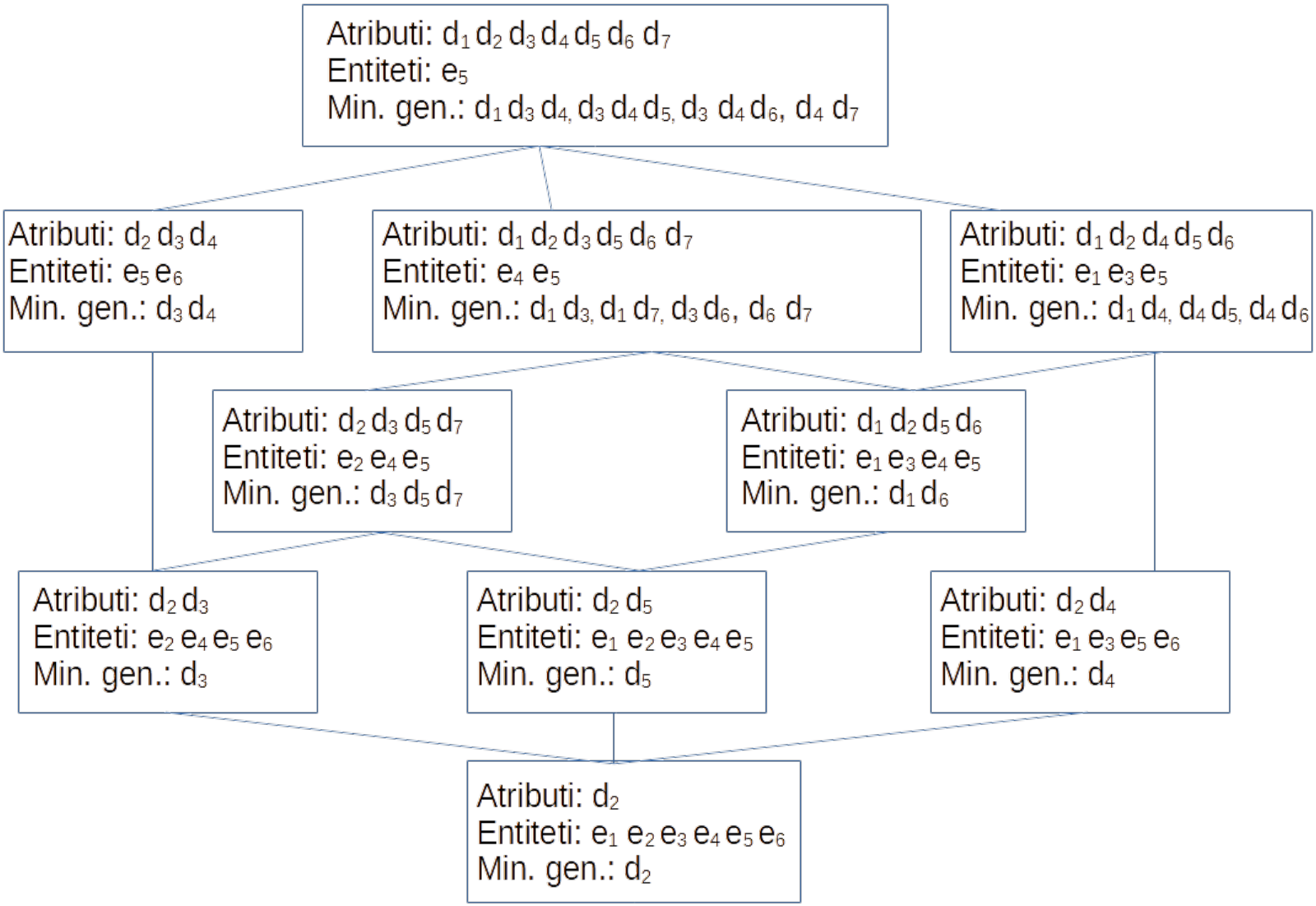
- Pristup radi nad podacima koji koriste binarne atribute i sastoje se od jednog pogleda.
- Glavna ideja je iskoristiti postojeće pristupe generiranja skupova frekventnih zatvorenih skupova objekata (CHARM), uz modifikacije koje bi omogućile stvaranje redeskripcija.
- Stvaramo modificiranu rešetku koja uz frekventni, zatvoreni skup objekata i njegovo pokrivanje, sadrži i skup **minimalnih generatora**.
- Minimalni generatori su skupovi objekata (atributa) iz kojih generiramo redeskripcije kandidate. Tvorimo ih kao skupove razlike frekventnih zatvorenih skupova objekata (čvorova rešetke) i skupova objekata njihove neposredne djece u rešetci.
- Glavna ideja je pronaći različite skupove atributa koji opisuju (pokrivaju) jako slične ili identične skupove entiteta.

# Traženje redeskripcija korištenjem frekventnih zatvorenih skupova objekata

## Definicija

Neka su $X, Y, Z \subseteq attrs(D)$ atributi, a $S \subseteq E$ skup entiteta. Uvjetna redeskripcija za skup entiteta $S$ je pravilo oblika $(X \Leftrightarrow Y)|Z$ takvo da: a) $attrs(X) \neq \emptyset$ i $attrs(Y) \neq \emptyset$, b) $attrs(X) \cap attrs(Y) = attrs(X) \cap attrs(Z) = attrs(Y) \cap attrs(Z) = \emptyset$, c) $Pot(X \wedge Z) = Pot(Y \wedge Z) = S$. Pravilo $Z$ se zove **uvjet**. Pravilo označava da su pravila $X$ i $Y$ ekvivalentna ili opisuju identičan skup entiteta $S$ **uz uvjet** pravila $Z$. Ako je $attrs(Z) = \emptyset$, tada je pravilo **bezuvjetna** redeskripcija za skup entiteta $S$. U tom slučaju vrijede jednostavniji uvjeti: a) $attrs(X) \neq \emptyset$ i $attrs(Y) \neq \emptyset$ , b) $attrs(X) \cap attrs(Y) = \emptyset$ , c) $Pot(X) = Pot(Y) = S$.

# Traženje redeskripcija korištenjem frekventnih zatvorenih skupova objekata



# Traženje redeskripcija korištenjem frekventnih zatvorenih skupova objekata

| Skup entiteta | Redeskripcija |
|---|---|
| $e_1\, e_3\, e_4\, e_5$ | $d_1 \Leftrightarrow d_6$ |
| $e_2\, e_4\, e_5$ | $d_3 \wedge d_5 \Leftrightarrow d_7$ |
| $e_1\, e_3\, e_5$ | $d_1 \Leftrightarrow d_5 \mid d_4$<br>$d_5 \Leftrightarrow d_6 \mid d_4$<br>$d_1 \Leftrightarrow d_6 \mid d_4$ |
| $e_4\, e_5$ | $d_1 \wedge d_3 \Leftrightarrow d_6 \wedge d_7$<br>$d_1 \wedge d_7 \Leftrightarrow d_3 \wedge d_6$<br>$d_3 \Leftrightarrow d_7 \mid d_6$<br>$d_3 \Leftrightarrow d_7 \mid d_1$<br>$d_1 \Leftrightarrow d_6 \mid d_3$<br>$d_1 \Leftrightarrow d_6 \mid d_7$ |
| $e_5$ | $d_1 \Leftrightarrow d_5 \mid d_3 \wedge d_4$<br>$d_1 \Leftrightarrow d_6 \mid d_3 \wedge d_4$<br>$d_5 \Leftrightarrow d_6 \mid d_3 \wedge d_4$<br>$d_1 \wedge d_3 \Leftrightarrow d_7 \mid d_4$<br>$d_3 \wedge d_6 \Leftrightarrow d_7 \mid d_4$<br>$d_3 \wedge d_5 \Leftrightarrow d_7 \mid d_4$ |

# Traženje redeskripcija korištenjem frekventnih zatvorenih skupova objekata

**CHARM-L** $(D)$:
$[\emptyset] = \{d_i : d_i \in attrs(\mathcal{D})\}$
CHARM-L-PROSIRI$([\emptyset], \mathcal{L}_r = \{\emptyset\})$
**return** $\mathcal{L}$ {rešetka zatvorenih frekventnih skupova objekata}

**CHARM-L-Prosiri** $([P], \mathcal{L}_c)$:
**for each** $X_i$ iz $[P]$ uz povećavajući $|Pot(X_i)|$ **do**
  $[X_i] = \emptyset$
  AZURIRAJINDEKS-C$(X_i, [P])$
  **for each** $X_j > X_i$ in $[P]$ **do**
    $X = X_i \cup X_j$, $Pot(X) = Pot(X_i) \cap Pot(X_j)$ **and** $\mathbb{C}(X) = \mathbb{C}(X_i) \cap \mathbb{C}(X_j)$
    CHARM-L-SVOJSTVO$(X, X_i, X_j)$
  **end for**
  $\mathcal{L}_n =$ PROVJERISUBSUMPCIJU$(\mathcal{L}_c, X_i, \mathbb{C}(X_i))$
  CHARM-L-PROSIRI$([X_i], \mathcal{L}_n)$
  **delete** $[X_i]$
**end for**

# Traženje redeskripcija korištenjem frekventnih zatvorenih skupova objekata

**ProvjeriSubsumpciju**$(\mathcal{L}_c, X, \mathbb{C}(X))$:
$\mathcal{P} = \{Z \in \mathcal{C} \mid Z.\text{cid} \in \mathbb{C}(X)\}$ {provjeravamo nadskupe od X u rešetci}
**for each** $Z \in \mathcal{P}$ **do**
**if** $|Pot(X)| = |Pot(Z)|$ **then**
**return** $\mathcal{L}_c$ {ukoliko postoji nadskup s istom potporom, izbacimo skup objekata}
**end if**
**end for**
$\mathcal{L}_n = X$
$\mathcal{L}_c$.roditelji.add($\mathcal{L}_n$), $\mathcal{L}_n$.djeca.add($\mathcal{L}_c$) {Dodamo $X$ kao roditelja od $\mathcal{L}_c$}
$\mathcal{P}^{\min} = \{Z \in \mathcal{P} \mid Z \text{ je minimalan}\}$
**for all** $Z \in \mathcal{P}^{\min}$ **do**
$\mathcal{L}_n$.roditelji.add($Z$), $Z$.djeca.add($\mathcal{L}_n$)
**for all** $Z_c \in Z$.djeca **do**
**if** $Z_c \subset \mathcal{L}_n$ **then**
$Z_c$.roditelji.remove($Z$), $Z$.djeca.remove($Z_c$) {ukoliko vrijedi $Z_c \subset \mathcal{L}_n \subseteq Z$, ažuriraj veze u rešetci}
**end if**
**end for**
**end for**
**return** $\mathcal{L}_n$

# Traženje redeskripcija korištenjem frekventnih zatvorenih skupova objekata

- `AZURIRAJINDEKS-C` ažurira indekse skupova objekata u rešetci.
- `CHARM-L-SVOJSTVO` ubacuje novo kreirani skup objekata $X = X_i \cup X_j$ u novu klasu $[X_i]$. Također, provjerava svojstva zatvorenosti skupova objekata: a) ako $Pot(X_i) \subseteq Pot(X_j)$ zamijenimo $X_i$ s $X_i \cup X_j$, b) ako $Pot(X_i) \supset Pot(X_j)$ zamijenimo $X_j$ s $X_i \cup X_j$.
- Sljedeći korak algoritma je generiranje **minimalnih generatora**.
- Minimalni generator $Z$ zatvorenog frekventnog skupa objekata $X$ je minimalan skup objekata koji je podskup od $X$, ali nije podskup niti jednog direktnog zatvorenog skupa objekata koji je podskup od $X$ u rešetci (nije podskup niti jednog djeteta od $X$ u rešetci).
- Neka je $\mathcal{Y} = \{Y_1, Y_2, \ldots, Y_k\}$ skup zatvorenih podskupa, susjeda od $X$ u rešetci $\mathcal{L}$, a $\mathcal{M}(Y_i)$ skup minimalnih generatora skupa objekata $Y_i$. Definiramo skup $\Delta_i = X \setminus Y_i$ (skup objekata koji su u $X$, a nisu u $Y_i$). $\Delta = \Delta_1, \Delta_2, \ldots, \Delta_k$.

# Traženje redeskripcija korištenjem frekventnih zatvorenih skupova objekata

- Skup objekata $Z$ se zove **dodirni skup** od $\Delta$ ako i samo ako $Z \cap \Delta_i \neq \emptyset,\ \forall i \in \{1, \ldots, k\}$.
- Skup objekata $Z$ se zove **minimalni dodirni skup** od $\Delta$ ako ne postoji niti jedan drugi dodirni skup $Z'$ takav da $Z' \subset Z$.

**Teorem**

Za dani skup objekata $X$, skup minimalnih generatora $\mathcal{M}(X)$ odgovara skupu minimalnih dodirnih skupova od $\Delta$.

# Traženje redeskripcija korištenjem frekventnih zatvorenih skupova objekata

**Ulaz:** Zatvoreni skup objekata $X$, Skup susjednih zatvorenih podskupa $\mathcal{Y}$ iz $\mathcal{L}$

**MinimalniGeneratori**$(X, \mathcal{Y})$:

$H(X) = \emptyset$;

$\Delta = \{\Delta_i = X - Y_i \mid Y_i \in \mathcal{Y}\}$;

**for each** $k$-torka $(z_1, z_2, \cdots, z_k)$, gdje $z_i \in \Delta_i$ **do**

$Z = \{z_1, z_2, \cdots, z_k\}$; {otkloni duplikate $z_i$}

$H(X) = H(X) \cup \{Z\}$;

**end for**

$\mathcal{M}(X) = \{Z \in H(X) \mid Z$ je minimalan u $H(X)\}$;

---

**GenerirajRedeskripcije**$(X \in \mathcal{C})$:

**for sve parove** $Y, Z \in \mathcal{M}(X)$ **do**

$Q = Y \cap Z$;

vrati : $Pot(X) : Y - Q \Longleftrightarrow Z - Q \mid Q$

**end for**

# Traženje redeskripcija - ReReMi

- ReReMi je **pohlepan** algoritam za traženje redeskripcija.
- Koristi podatke koji sadrže dva pogleda, te može bez predprocesiranja raditi s podacima koji sadrže binarne, kategorijske i numeričke atribute uz postojanje nedostajućih vrijednosti.
- Logički operatori i interpretacije se moraju proširiti da bi omogućili traženje redeskripcija uz postojanje nedostajućih vrijednosti.
- Svaka interpretacija nad formulom koja sadrži jedan atribut, takav da entitet za njega ima nedostajuću vrijednost, vraća nedostajuću vrijednost ? kao rezultat logičke evaluacije za taj entitet.
- Logički operatori se definiraju kao: $\neg ? = ?$; $a \wedge ? = ?$ ako i samo ako $a \neq 0$,inače 0; i $a \vee ? = 1$ ako i samo ako $a = 1$, inače ?.
- Definiramo skupove $E_{(i_1,\ldots,i_n)}(R)$, gdje $i_1, i_2, \ldots, i_n \in \{0, 1, ?\}$ odgovara pravilima redeskripcije $R = (q_1, q_2, \ldots, q_n)$, $n = |\mathcal{W}|$. $E_{(i_1,\ldots,i_n)}$ označava skup entiteta za koje je pravilo $q_k$ istinito ako $i_k = 1$, lažno ako $i_k = 0$, ili ima logičku vrijednost ? ako $i_k = ?$.

# Traženje redeskripcija - ReReMi

- Npr. $E_{(1,1,\ldots,1)}$ je skup entiteta opisanih od strane svakog pravila redeskripcije, $E_{(0,0,\ldots,0)}$ je skup entiteta koji nije opisan od niti jednog od $|\mathcal{W}|$ pravila redeskripcije, a $E_{(?,?,\ldots,?)}$ je skup entiteta koji imaju nedostajuću vrijednost za sva pravila redeskripcije.
- Označimo sa $S$ skup svih $n$-torki $I = (i_1, \ldots, i_{|\mathcal{W}|})$, $i_k \in \{0, 1, ?\}$.
  *Optimistični Jaccard indeks*:

$$J_{opt}(R) = \frac{\sum_{I \in S,\ 0 \notin I} |E_I|}{\sum_{I \in S,\ 1 \in I \vee 0 \notin I} |E_I|}.$$

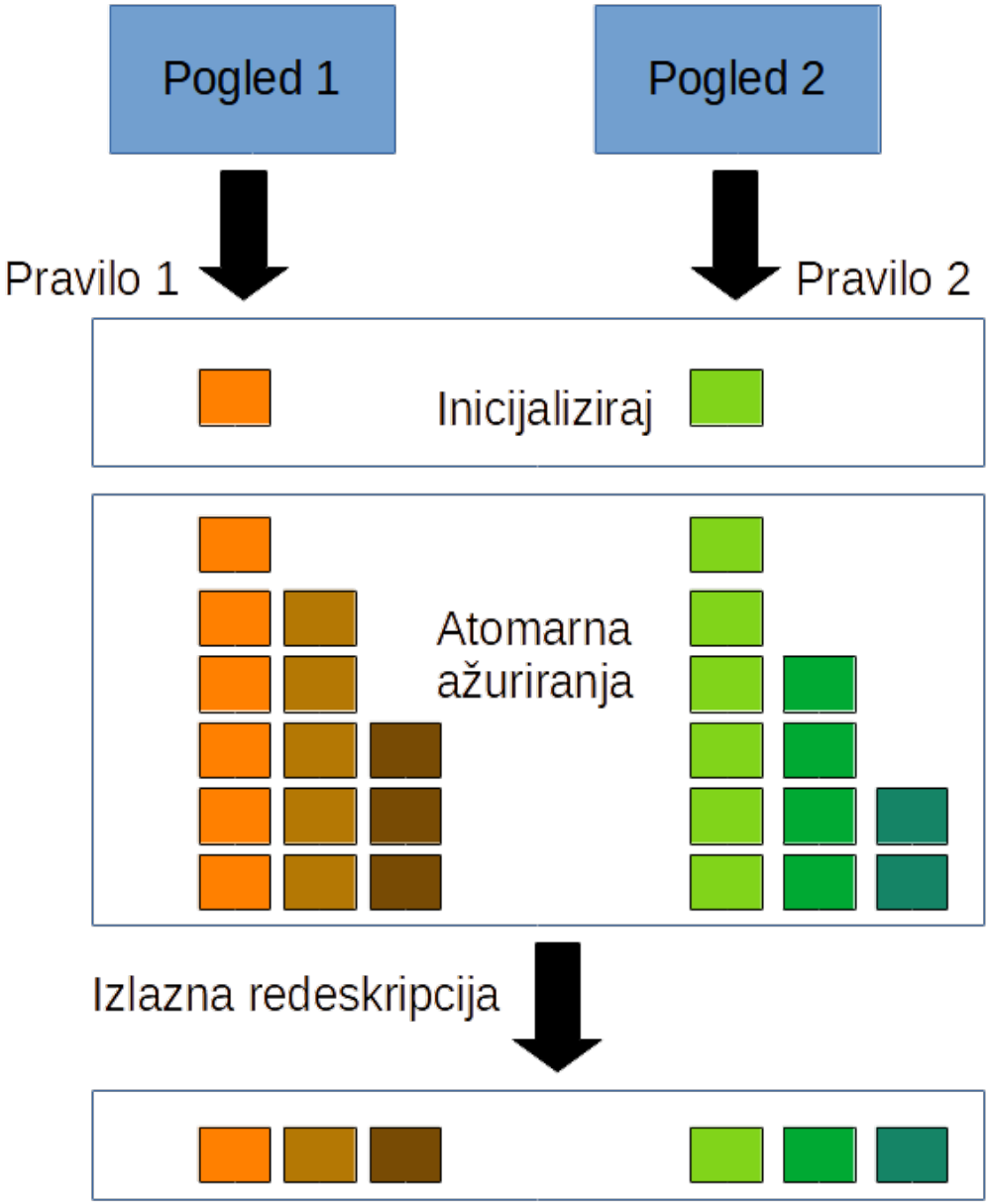
  *Pesimistični Jaccard indeks*:

$$J_{pess}(R) = \frac{|E_{1,1,\ldots,1}|}{\sum_{I \in S,\ 1 \in I \vee ? \in I} |E_I|}.$$

  *Jaccard indeks uz egzaktnu evaluaciju skupa potpore pravila*:

$$J_{qnm}(R) = \frac{|E_{1,1,\ldots,1}|}{\sum_{I \in S,\ 1 \in I} |E_I|}.$$

# Traženje redeskripcija - ReReMi



# Traženje redeskripcija - ReReMi

- Algoritam radi diskretizaciju numeričkih atributa u niz pod-intervala.
- Prvi korak izgradnje redeskripcije se sastoji od odabira dva atributa (ili pod-intervala) kao inicijalni par pravila za izgradnju redeskripcije (npr. $(22 \leq \text{temperatura} \leq 30,\ 2 \leq \text{VodaZePice} \leq 6)$). Biramo par koji ima najveću točnost.
- Zatim u koracima primijenjujemo atomarna ažuriranja na tu inicijalnu redeskripciju s ciljem povećavanja Jaccard indeksa (točnosti redeskripcije). U svakom koraku ažuriramo redeskripciju kandidatom koji najviše poveća Jaccard indeks (pohlepni pristup).
- Atomarno ažuriranje se sastoji od izabira novog atributa (pod-intervala) i primjene jednog od operatora ($\wedge,\ \vee$) uz potencijalno operator $\neg$.
- Npr. $(22 \leq \text{temperatura} \leq 30\ \vee\ \text{KvarCjevovoda},$ $2 \leq \text{VodaZePice} \leq 6)$ je jedno atomarno ažuriranje.
- U slučaju podataka s nedostajućim vrijednostima specificiramo verziju Jaccard indeksa koji optimiramo, inače se optimira standardni Jaccard indeks.

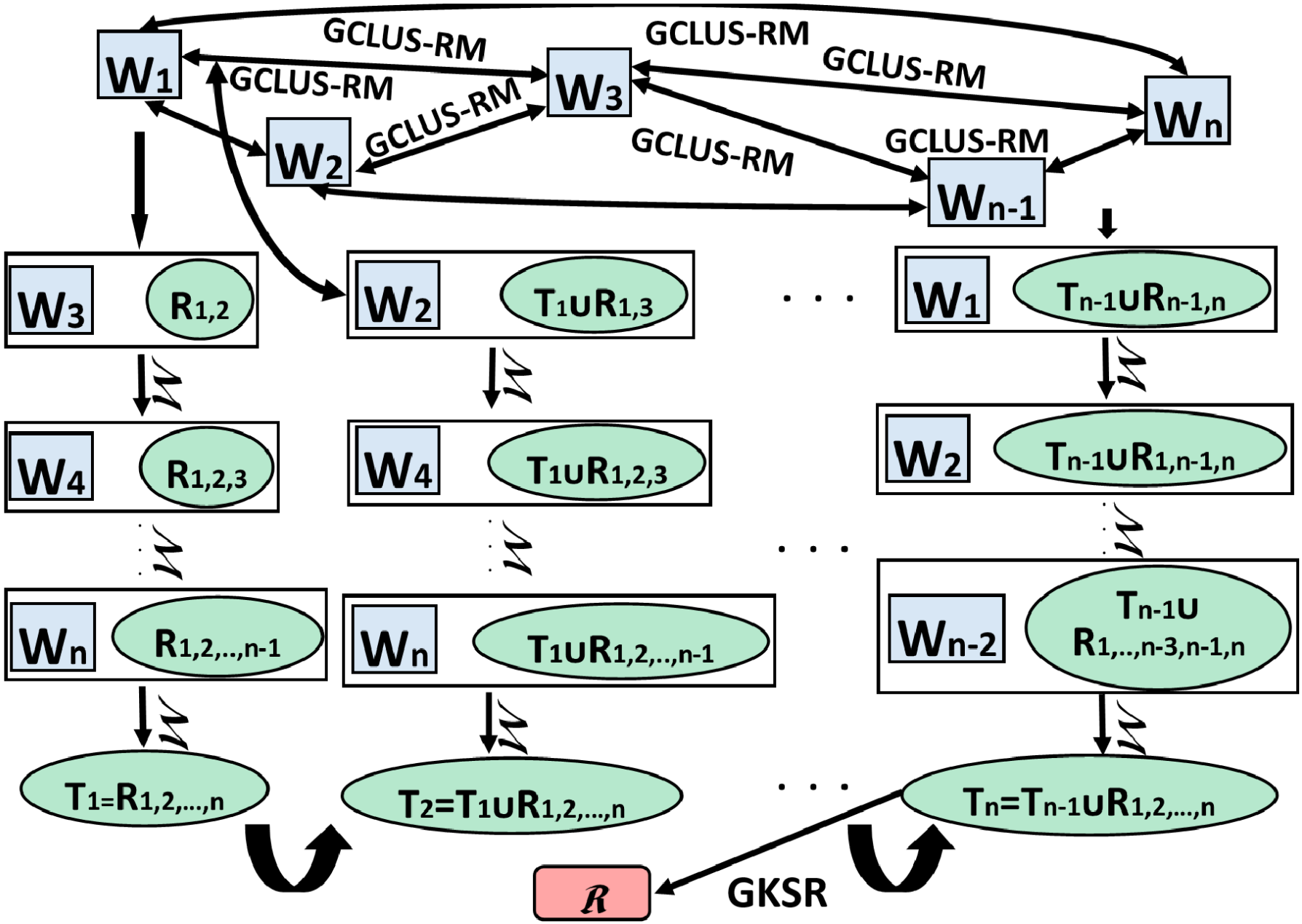
# Traženje redeskripcija - ReReMi

**Ulaz:** Skup $\mathcal{D} = (V_L, V_D, E, D_L, D_D)$, ne-negativni cijeli brojevi $k_p$, $k_i$, uvjeti $\mathcal{C}$.
**Izlaz:** Skup redeskripcija, $\mathcal{R}$.
$\mathcal{R} \leftarrow \emptyset$
$\mathcal{I} \leftarrow \{k_p$ najboljih inicijalnih redeskripcija s pravilima koja sadrže samo jedan atribut$\}$
**for** $S \in \mathcal{I}$ **do**
  $\mathcal{K} \leftarrow \{S\}$
  $S_L(S), S_D(S) \leftarrow$ slobodne varijable za $S$
  **if** $S_L(S) \neq \emptyset$ ili $S_D(S) \neq \emptyset$ **then**
    $\mathcal{E} \leftarrow \{S\}$
    **while** $\mathcal{E} \neq \emptyset$ **do**
      **for each** $R \in \mathcal{E}$ **do**
        **for** stranu $s \in \{\mathrm{L}, \mathrm{D}\}$ i operator $\circ \in \{\vee, \wedge\}$ **do**
          **if** $R$ može biti proširen na strani $s$ operatorom $\circ$ i literal $l \in S_s(R)$ **then**
            $\mathcal{K} \leftarrow \mathcal{K} \cup \{$najbolje takvo proširenje od $R$ koje zadovoljava uvjete $C\}$
      $\mathcal{K} \leftarrow \{k_i$ najboljih redeskripcija iz $\mathcal{K}$, s ažuriranim slobodnim varijablama.$\}$
      $\mathcal{E} \leftarrow \{R \in \mathcal{K} : S_L(R) \neq \emptyset$ ili $S_D(R) \neq \emptyset\}$
  $\mathcal{R} \leftarrow \mathcal{R} \cup \mathcal{K}$
Filtriraj $\mathcal{R}$ uvažavajući uvjete $C$
**return** $\mathcal{R}$

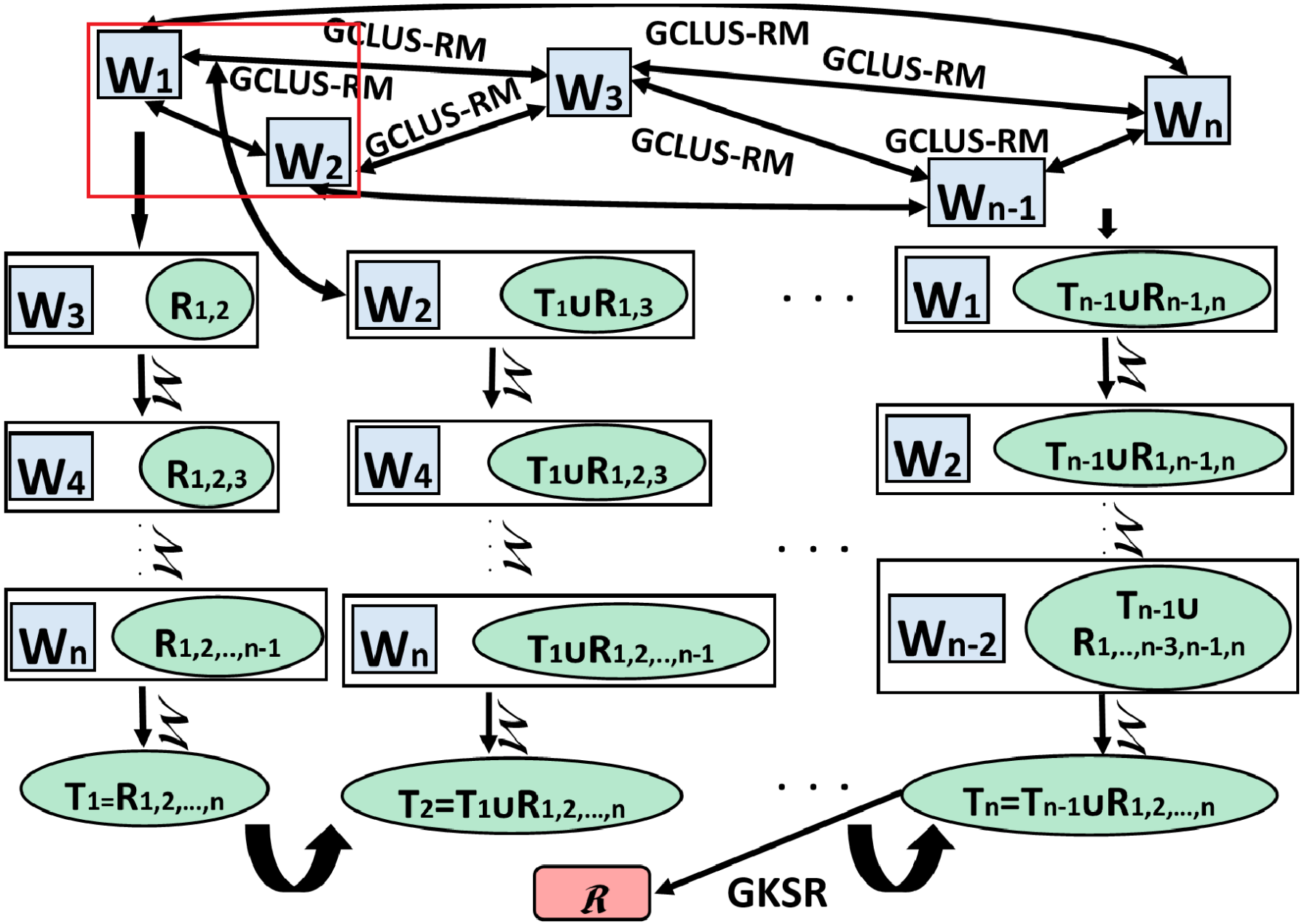
# Traženje redeskripcija - CLUS-RM

- Algoritam CLUS-RM je baziran na Prediktivnim stablima klasteriranja.
- Podržava binarne, kategorijske i numeričke podatke bez predprocesiranja.
- Podržava nedostajuće vrijednosti te proizvoljan (konačan) broj pogleda.
- Zbog korištenja Prediktivnih stabala klasteriranja, koja podržavaju zadatke višeciljne klasifikacije/regresije, algoritam u jednoj iteraciji (alternaciji) može pronaći znatno veći broj redeskripcija od algoritama baziranih na CARTwheels principu.
- Algoritam koristi **višekriterijsku optimizaciju** za stvaranje skupova redeskripcija predefinirane veličine.
- Glavni koraci algoritma su: a) inicijalizacija, b) izabir para pogleda, alternacije i kreiranje dvo-poglednih redeskripcija, c) nadopunjavanje redeskripcija i selekcija kandidata, d) višekriterijska optimizacija skupa redeskripcija.

# Traženje redeskripcija - CLUS-RM



# Traženje redeskripcija - CLUS-RM



# Traženje redeskripcija - CLUS-RM

- GCLUS-RM algoritam pronalazi nepotpune redeskripcije iz izabranog para pogleda.
- Prvi korak u kreiranju redeskripcija je stvaranje inicijalizacijskog skupa.
- Inicijalizacijski korak stvara $E$ umjetnih entiteta, gdje je vrijednost svakog atributa umjetnog entiteta dobivena iz slučajno odabranog originalnog entiteta. Na taj način se zadržavaju distribucije vrijednosti, međutim narušavaju se ili uništavaju postojeće ekvivalencije i korelacije između atributa.
- U donjem primjeru, umjetni entitet $E_1'$ iz pogleda 1 je kreiran kopiranjem vrijednosti atributa $W_1A_1$ iz originalnog entiteta $E_4$, vrijednosti atributa $W_1A_2$ od $E_1$, a vrijednosti $W_1A_3$ od $E_5$.

# Traženje redeskripcija - CLUS-RM

| Entitet | $W_1A_1$ | $W_1A_2$ | $W_1A_3$ |
|---|---|---|---|
| $E_1$ | 1.1 | 2.5 | 3.4 |
| $E_2$ | 1.5 | 2.2 | 4.0 |
| $E_3$ | 5.5 | -0.6 | -0.2 |
| $E_4$ | 4.4 | -0.2 | 2.0 |
| $E_5$ | 3.2 | 1.7 | 2.9 |

(a) Originalni skup podataka pogleda 1

| Entity | $W_2A_1$ | $W_2A_2$ | $W_2A_3$ |
|---|---|---|---|
| $E_1$ | TRUE | FALSE | FALSE |
| $E_2$ | TRUE | TRUE | FALSE |
| $E_3$ | FALSE | FALSE | TRUE |
| $E_4$ | TRUE | TRUE | TRUE |
| $E_5$ | TRUE | FALSE | TRUE |

(b) Originalni skup podataka pogleda 2

| Entity | $W_1A_1$ | $W_1A_2$ | $W_1A_3$ | Target |
|---|---|---|---|---|
| $E_1$ | 1.1 | 2.5 | 3.4 | 1.0 |
| $E_2$ | 1.5 | 2.2 | 4.0 | 1.0 |
| $E_3$ | 5.5 | -0.6 | -0.2 | 1.0 |
| $E_4$ | 4.4 | -0.2 | 2.0 | 1.0 |
| $E_5$ | 3.2 | 1.7 | 2.9 | 1.0 |
| $E_1$' | 4.4 | 2.5 | 2.9 | 0.0 |
| $E_2$' | 3.2 | -0.6 | 4.0 | 0.0 |
| $E_3$' | 3.2 | -0.6 | 2.9 | 0.0 |
| $E_4$' | 4.4 | -0.2 | 4.0 | 0.0 |
| $E_5$' | 5.5 | 1.7 | 2.9 | 0.0 |

(c) Inicijalni skup za pogled 1

| Entity | $W_2A_1$ | $W_2A_2$ | $W_2A_3$ | Target |
|---|---|---|---|---|
| $E_1$ | TRUE | FALSE | FALSE | 1.0 |
| $E_2$ | TRUE | TRUE | FALSE | 1.0 |
| $E_3$ | FALSE | FALSE | TRUE | 1.0 |
| $E_4$ | TRUE | TRUE | TRUE | 1.0 |
| $E_5$ | TRUE | FALSE | TRUE | 1.0 |
| $E_1$' | TRUE | FALSE | TRUE | 0.0 |
| $E_2$' | FALSE | FALSE | TRUE | 0.0 |
| $E_3$' | TRUE | TRUE | TRUE | 0.0 |
| $E_4$' | FALSE | TRUE | FALSE | 0.0 |
| $E_5$' | FALSE | FALSE | TRUE | 0.0 |

(d) Inicijalni skup za pogled 2

# Traženje redeskripcija - CLUS-RM

- Na svakom od inicijalnih skupova gradimo Prediktivno stablo klasteriranja i pretvaramo ih u skupove pravila.
- Pamtimo jednu listu pravila za svaki pogled, a u listu dodajemo pravilo ako i samo ako se razlikuje od postojećiih pravila za predefinirani broj atributa.
- Kreiramo varijable cilja prema pravilima dobiveniim u prethodnom koraku i u inicijalizaciji.
- Pravila dobivena na pogledu $W_1$ koristimo za stvaranje varijabli cilja za klasteriranje na $W_2$ i obrnuto.
- Za svaki entitet u skupu podataka, dodajemo ciljnu varijablu vrijednosti 1.0 ukoliko je entitet pokriven specifičnim pravilom, inače definiramo vrijednost 0.0.
- Atribut $W_2T_1$ iz skupa podataka koji odgovara pogledu $W_1$ predstavlja pravilo $IF\ W_2A_1 = TRUE$ (konstruirano na skupu podataka koje odgovara pogledu $W_2$). Pravilo opisuje entitete $E_1$, $E_2$, $E_4$, $E_5$.

# Traženje redeskripcija - CLUS-RM

- Dodavanjem odgovarajuće varijable cilja ($W_2 T_1$) u skup podataka koji odgovara pogledu 1 vodimo Prediktivno stablo klasteriranja da stvori klaster koji sadrži identičan skup entiteta s atributima pogleda 1. Pravilo koje zadovoljava to svojstvo je $IF\ W_1 A_3 > 0$.

| E | $W_1 A_1$ | $W_1 A_2$ | $W_1 A_3$ | $W_2 T_1$ | $W_2 T_2$ |
|---|---|---|---|---|---|
| $E_1$ | 1.1 | 2.5 | 3.4 | 1.0 | 0.0 |
| $E_2$ | 1.5 | 2.2 | 4.0 | 1.0 | 0.0 |
| $E_3$ | 5.5 | -0.6 | -0.2 | 0.0 | 0.0 |
| $E_4$ | 4.4 | -0.2 | 2.0 | 1.0 | 0.0 |
| $E_5$ | 3.2 | 1.7 | 2.9 | 1.0 | 1.0 |

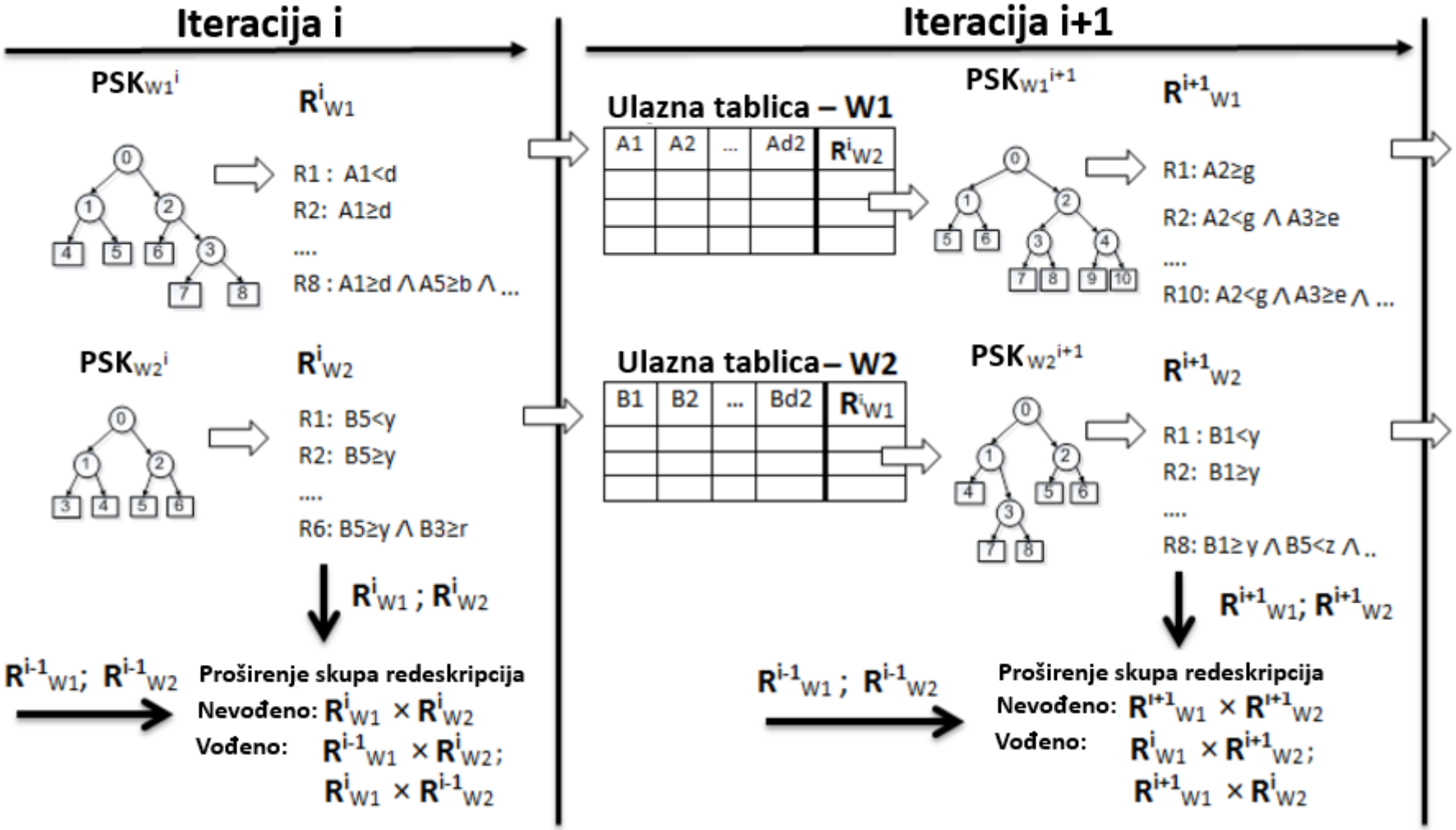
(e) Skup podataka pogleda 1

| E | $W_2 A_1$ | $W_2 A_2$ | $W_2 A_3$ | $W_1 T_1$ | $W_1 T_2$ |
|---|---|---|---|---|---|
| $E_1$ | TRUE | FALSE | FALSE | 0.0 | 1.0 |
| $E_2$ | TRUE | TRUE | FALSE | 0.0 | 1.0 |
| $E_3$ | FALSE | FALSE | TRUE | 1.0 | 0.0 |
| $E_4$ | TRUE | TRUE | TRUE | 1.0 | 0.0 |
| $E_5$ | TRUE | FALSE | TRUE | 1.0 | 1.0 |

(f) Skup podataka pogleda 2

# Traženje redeskripcija - CLUS-RM

- Postupak se iterativno ponavlja predefinirani broj iteracija (alternacija).

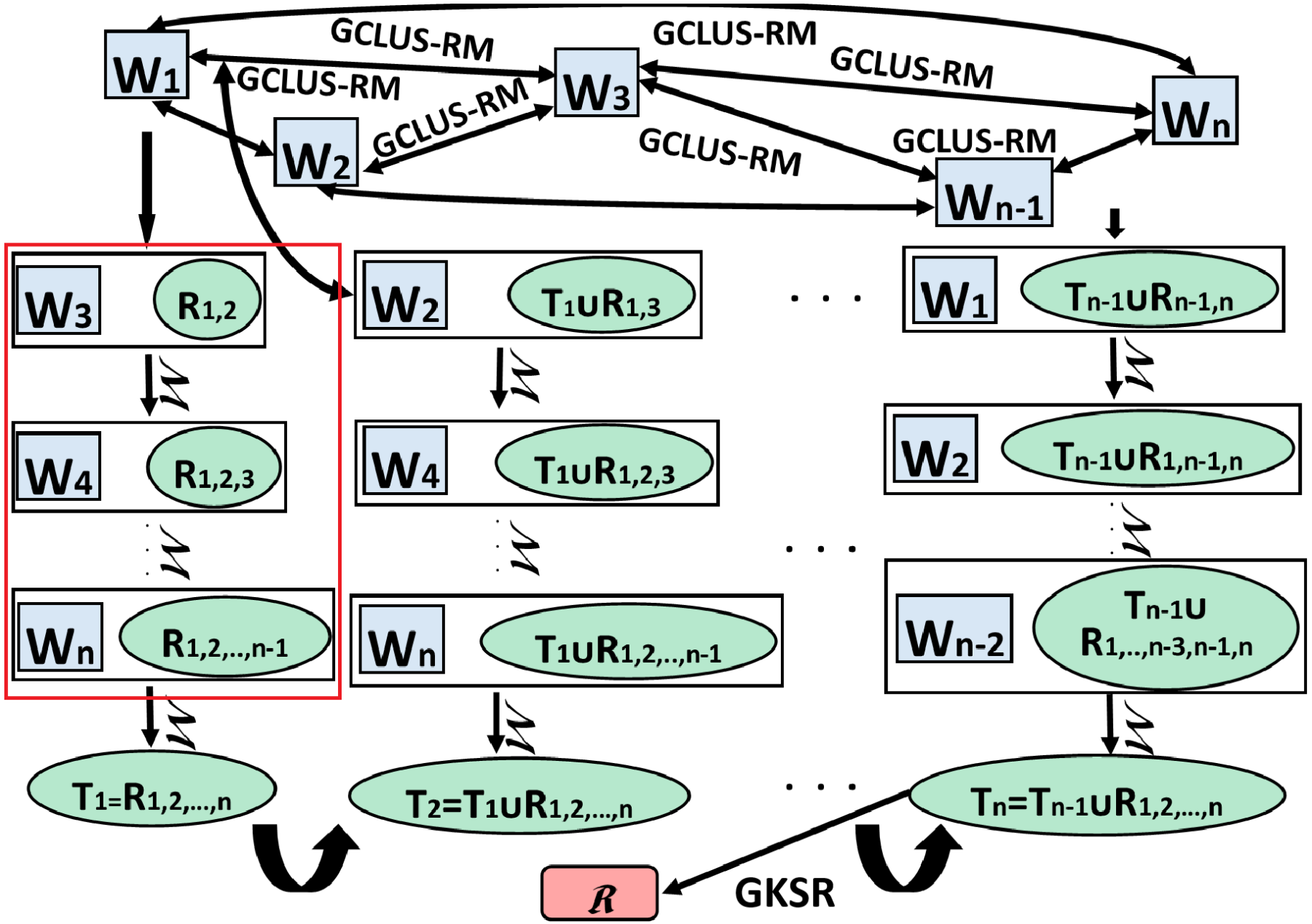


## **Algoritam** GCLUS-RM algoritam

**Ulaz:** Prvi pogled ($W_i$), Drugi pogled ($W_j$), Uvjeti $\mathcal{C}$, Postavke $\mathcal{S}$, Algoritmi modela $Alg$ i pomoćnog modela $Alg'$
**Izlaz:** Skup redeskripcija $\mathcal{R}$

$[\mathcal{M}_{Wi_{inic}}, \mathcal{M}_{Wj_{inic}}] \leftarrow$ **kreirajInicijalneM**($W_i$, $W_j$, $Alg$)
$[r_{W_i}, r_{W_j}] \leftarrow$ **kreirajPravilaIzM**($\mathcal{M}_{Wi_{inic}}, \mathcal{M}_{Wj_{inic}}$)
**while** TrenInd$<\mathcal{S}$.**maksIter do**
  $[D_{W_i}, D_{W_j}] \leftarrow$ **konstruirajCiljeve**($r_{W_i}$,$r_{W_j}$)
  $[\mathcal{M}_{W_i}, \mathcal{M}_{W_j}] \leftarrow$ **kreirajM**($D_{W_i}, D_{W_j}, Alg$)
  $[r_{W_i}, r_{W_j}] \leftarrow$**kreirajPravilaIzM**($\mathcal{M}_{W_i}, \mathcal{M}_{W_j}$)
  **if** ($\mathcal{C}.brojPomModela > 0$) **then**
    $[\mathcal{M}'_{W_i}, \mathcal{M}'_{W_j}] \leftarrow$ **createMs**($D_{W_i}, D_{W_j}, Alg'$)
    $[r_{W_i}, r_{W_j}] \leftarrow$ **kreirajPravilaIzM**($\mathcal{M}'_{W_i}, \mathcal{M}'_{W_j}$)
  **end if**
  **if** ($|\mathcal{R}| \leq \mathcal{C}.MaksVelicinaProsirenja$) **then**
    $\mathcal{R} \leftarrow \mathcal{R} \cup$ **stvoriRedeskripcije**($r_{W_i}, r_{W_j}, \mathcal{C}$)
  **else**
    $\mathcal{R}' \leftarrow$ **stvoriRedeskripcije**($r_{W_i}, r_{W_j}, \mathcal{C}$)
    **for** ($R \in \mathcal{R}'$) **do**
      $R_k \leftarrow argmax_{R' \in \mathcal{R}}(J(R) - J(R') - (1 - elemJ(R, R'))),\ J(R) > J(R')$, **nViews**($R'$)$< \mathcal{S}.n$
      $\mathcal{R} \leftarrow \mathcal{R} \setminus R_k \cup R$
    **end for**
  **end if**
  **if** ($\mathcal{C}.brojPomocnihModela > 0$) **then**
    $[r_{W_i}, r_{W_j}] \leftarrow$**brisiPomocnaPravila**($r_{W_i}, r_{W_j}$)
  **end if**
**end while**
**return** $\mathcal{R}$

# Traženje redeskripcija - CLUS-RM

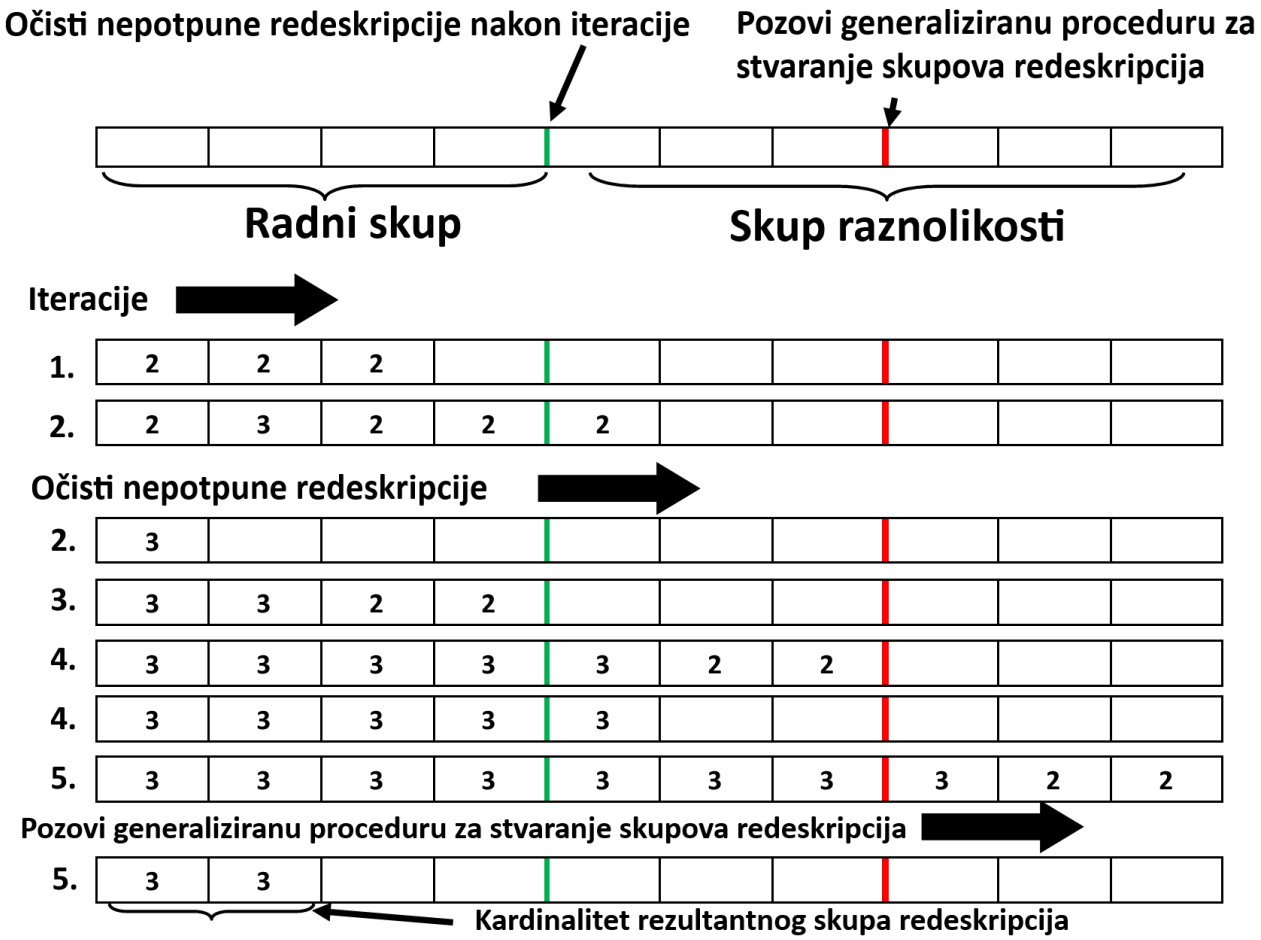


# Traženje redeskripcija - CLUS-RM

- Popunjavanja redeskripcija se odvija na sličan način kao i traženje redeskripcija iz dva pogleda.
- Umjesto pravila, ciljevi se stvaraju prema skupovima potpore nepotpunih redeskripcija.
- Tim postupkom omogućavamo stvaranje Prediktivnih stabala klasteriranja nad pogledima koje nismo pretražili, tako da opisuju identične (ili slične) podskupove entiteta kao podskup opisan postojećim nepotpunim redeskripcijama.
- Iterativno od redeskripcije, npr. $R = (q_1, q_2, ?, \ldots, ?)$ dobivamo: $R = (q_1, q_2, q_3, \ldots, ?)$, $\ldots$, $R = (q_1, q_2, q_3, \ldots, q_n)$.
- U svakom koraku pazimo da je točnost redeskripcije dovoljno velika i da je redeskripcija signifikantna.
- Za redeskripciju $R = (q_1, q_2, ?, \ldots, ?)$, točnost redeskripcije $R' = (q_1, q_2, q_3, ? \ldots, ?)$ možemo izračunati kao $\frac{|Pot(R) \cap Pot(q_3)|}{|\cup_{i<3} Pot(q_i) \cup Pot(q_3)|}$.
- $|\cup_{i<3} Pot(q_i)|$ već imamo u memoriji nakon računanja točnosti od $R$.

# Traženje redeskripcija - CLUS-RM

- Kreiranje višepoglednih redeskripcija često dovodi do stvaranja velikog broja kandidata. Zbog toga algoritam koristi mehanizme adaptivnog oslobađanja memorije.

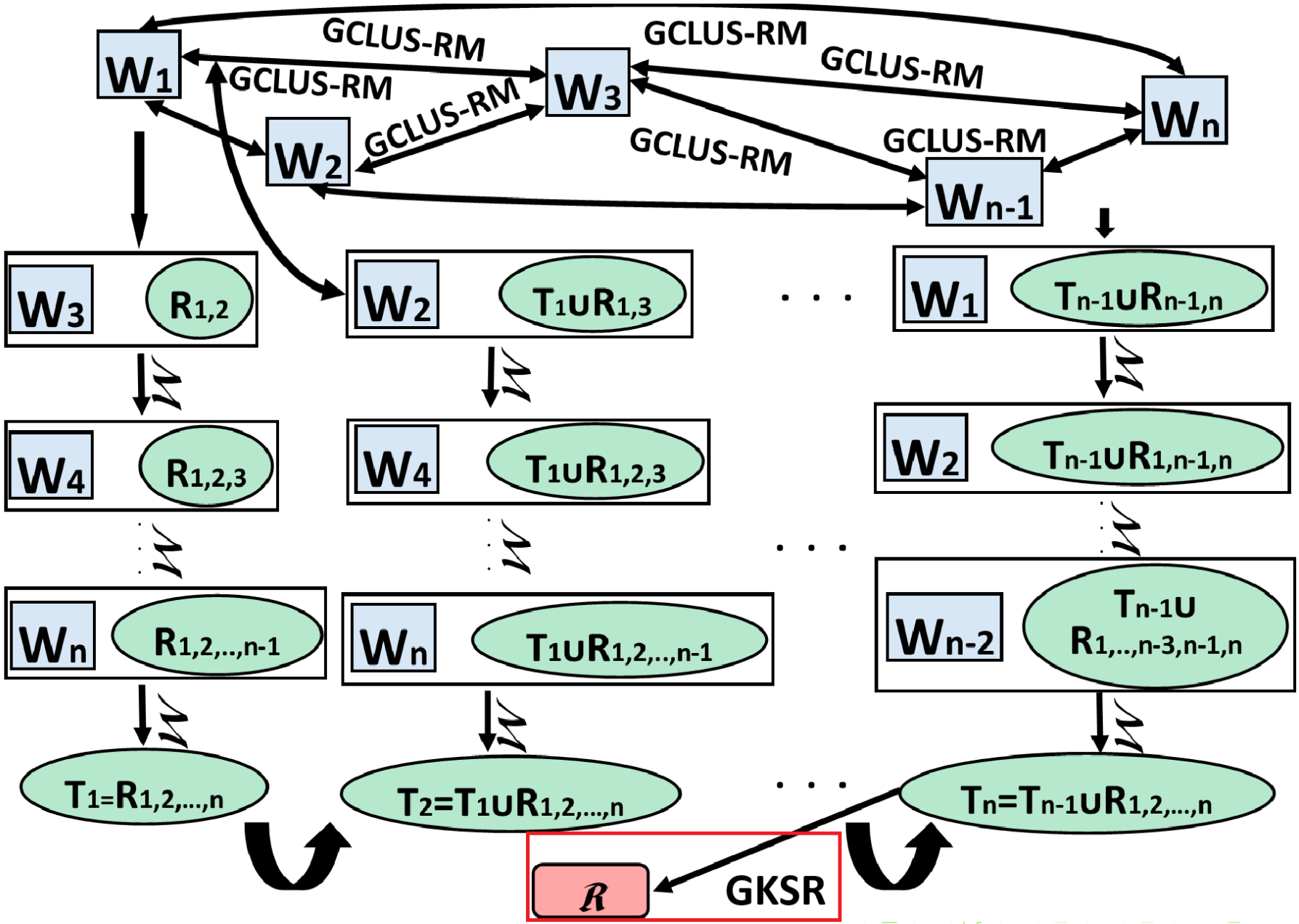


## **Algoritam** Algoritam za višepogledno traženje redeskripcija

**Ulaz:** Pogledi $MW = \{W_1, \ldots, W_n\}$, Uvjeti $\mathcal{C}$, Postavke $\mathcal{S}$, Algoritmi $Alg$ i pomoćni $Alg'$
**Izlaz:** Familija skupova redeskripcija zadane veličine $\mathcal{R}$

$\mathcal{R}_{svi} \leftarrow \emptyset$
**for** (in = 0; in<$\mathcal{S}.NSIInic$; in++) **do**
  $MW' = \{W'_1, \ldots, W'_n\} \leftarrow$`inicijalizirajPoglede()`
  **for** (i=0; i<$|MW| - 1$; i++) **do**
    **for** (j=i+1; j<$|MW|$; j++) **do**
      Ind$\leftarrow$ 0, $\mathcal{R}_{i,j} \leftarrow$ `GCLUS_RM`$(W'_i, W'_j, \mathcal{C}, Alg, Alg', \mathcal{S})$, $R_{all} \leftarrow R_{all} \cup R_{i,j}$
      **for** (k=0, $k \notin \{i, j\}$; k<$|MW|$; k++) **do**
        $DW_k \leftarrow$`konstruirajCiljeve`$(W_k, \mathcal{R}_{all})$
        $\mathcal{M}_k \leftarrow$ kreirajM$(DW_k, Alg)$, $r_k \leftarrow$`kreirajPravilaIzM`$(\mathcal{M}_k)$
        **if** ($\mathcal{C}.brojPomocnihModela > 0$) **then**
          $\mathcal{M}'_k \leftarrow$ kreirajPomocneM$(DW_k, Alg')$
          $r_k \leftarrow$`kreirajPravilaIzM`$(\mathcal{M}'_k)$
        **end if**
        $\mathcal{R}_{all} \leftarrow$`upotpuniRedeskripcije`$(\mathcal{R}_{all}, r_k, \mathcal{S}.Op, \mathcal{C}, W'_k)$
        **if** ($\mathcal{C}.brojPomocnihModela > 0$) **then**
          $r_k \leftarrow$brisiPomocnaPravila$(r_k)$
        **end if**
        $\mathcal{R}_{all} \leftarrow$`normalizirajMemoriju`$(\mathcal{R}_{all}, \mathcal{C}, \mathcal{S})$
      **end for**
    **end for**
  **end for**
**end for**
**for** ($nws = 2$; $nws < |MW|; nws++$) **do**
  $\mathcal{R}_{all} \leftarrow$`brisiNepotpune`$(\mathcal{R}_{all}, nws)$
**end for**
$\mathcal{R}_S \leftarrow$`GKSR`$(\mathcal{R}_{all}, \mathcal{S}.\mathcal{W}, \mathcal{S}.r)$
**return** $\mathcal{R}_S$

# Traženje redeskripcija - CLUS-RM



# Traženje redeskripcija - CLUS-RM

- Postupak generalizirane konstrukcije (GKSR) skupova redeskripcija koristi višekriterijsku optimizaciju za kreiranje skupova redeskripcija predefiniranog kardinaliteta (ili manjeg).
- Postupak koristi sljedeće mjere kvalitete redeskripcije:
  - $k_J(R) = 1.0 - J(R)$
  - $$k_p(R) = \begin{cases} \frac{log_{10}(pV(R))}{17} + 1 & , pV(R) \geq 10^{-17} \\ 0 & , pV(R) < 10^{-17} \end{cases}$$
  - $$k_{komp} = \begin{cases} \frac{|attr(R)|}{k} & , |attr(R)| < k \\ 1 & , k \leq |attr(R)| \end{cases}$$
  - $k_{slElem}(R_i) = max_j \ J(supp(R_i), supp(R_j)), \ j = 1, \ldots, |\mathcal{R}_{red}|$
  - $k_{slAtrib}(R) = max_j \ J(attrs(R_i), attrs(R_j)), \ j = 1, \ldots, |\mathcal{R}_{red}|$
  - $k_{pojEl}(R) = \frac{\sum_{e_k \in Pot(R)} E_{ocur}[k]}{\sum_{j=1}^{|E|} E_{ocur}[j]}$.
  - $k_{pojAt}(R) = \frac{\sum_{a_k \in attrs(R)} A_{ocur}[k]}{\sum_{j=1}^{|V_1|+|V_2|} A_{ocur}[j]}$

# Traženje redeskripcija - CLUS-RM

- $R_{prvi} = argmin_R\ (w_0 \cdot (1.0 - J(R)) + w_1 \cdot k_p(R) + w_2 \cdot k_{pojEl}(R) + w_3 \cdot score_{kpojAt}(R) + w_4 \cdot k_{komp}(R))$
- $R_{trenutni} = argmin_R\ (w_0 \cdot (1.0 - J(R)) + w_1 \cdot (\frac{k}{n} \cdot k_p(R) + (1 - \frac{k}{n}) \cdot \frac{Pot(R)}{|E|}) + w_2 \cdot k_{slElem}(R) + w_3 \cdot k_{slAtrib}(R) + w_4 \cdot k_{komp}(R))$, gdje $k$ označava broj redeskripcija u skupu koji konstruiramo u trenutnom koraku izgradnje.
- Postupak garantirano bira Pareto optimalnu (ne-dominiranu) redeskripciju u svakom koraku, stoga je postupak izgradnje skupa pohlepan.

# Softver

- Python - **Siren**, `https://cs.uef.fi/siren/main/intro.html` (`ReReMi`, `CARTwheels`, `SplitTrees`, `LayeredTrees`).
- Java - **CLUS-RM**, `https://github.com/matmih/GeneralizedMWRM` (`CLUS-RM`).