Logică Matematică şi Computaţională
TEMELE COLECTIVE **8–16**

Claudia MUREȘAN
cmuresan@fmi.unibuc.ro, claudia.muresan@g.unibuc.ro, c.muresan@yahoo.com

Universitatea din București
Facultatea de Matematică și Informatică
București

2023–2024, Semestrul I

Toate temele colective se adresează AMBELOR SERII.
Rezolvarea fiecărei teme colective trebuie trimisă *într–un singur exemplar* de *fiecare grupă a seriei IF* și *fiecare grupă a seriei ID*. Până la remedierea problemei de stocare în OneDrive, rezolvările temelor vor fi încărcate în *folderul din Google drive* indicat în instrucțiunile pentru acest assignment; ca *backup*, dacă vă permite MS Teams–ul, le puteți încărca *și* în acest assignment.

În cerințele de programare în Prolog din următoarele exerciții se va folosi această **convenție** din documentația Prolog–ului pentru scrierea **argumentelor** în cadrul **predicatelor**, în descrierile acestor predicate:

- argumentele precedate de $+$ trebuie furnizate interpretorului de Prolog în interogări;
- argumentele precedate de $-$ vor fi construite de interpretorul de Prolog pe baza argumentelor furnizate în interogări și, desigur, a clauzelor care definesc respectivele predicate.

A se observa și argumentele precedate de ?, care pot avea oricare dintre cele două roluri anterioare (vedeți, de exemplu, descrierea predicatului *append* în documentația de pe site–ul SWI–Prolog).
Ca și la exercițiile de laborator făcute în săptămânile anterioare, **relațiile binare** vor fi reprezentate prin liste de perechi de constante, iar **mulțimile** între care acestea sunt definite vor fi reprezentate prin liste de constante Prolog.
Predicatele scrise la laborator (sau curs) vor fi folosite fără a fi rescrise, utilizând **directiva :- [...].** pentru includerea bazelor de cunoștințe de la laborator (sau curs) în cea curentă, însă respectând modurile indicate mai jos de definire a predicatelor cerute.

Temă colectivă (din AL DOILEA SET DE CURSURI)

Fie A și B mulțimi, iar $f : A \to B$. Considerăm f ca relație binară funcțională totală de la A la B, i.e. o identificăm cu graficul ei. Demonstrați că:

$$f = A \times B \text{ ddacă } \begin{cases} A = \emptyset \\ \text{sau} \\ |B| = 1. \end{cases}$$

Temă colectivă (de programare în Prolog)

Să se definească următoarele predicate:

• *imagine*(+*Functie*, +*SubmultimeADomeniuluiFunctiei*, −*Imagine*), care determină în al treilea argument imaginea mulțimii dată de lista de constante din al doilea argument prin funcția dată de primul argument ca relație funcțională totală, deci ca listă de perechi de constante;

• *preimagine*(+*Functie*, +*SubmultimeACodomeniuluiFunctiei*, −*Preimagine*), care determină în al treilea argument preimaginea mulțimii dată de lista de constante din al doilea argument prin funcția dată de primul argument ca relație funcțională totală, deci ca listă de perechi de constante.

Temă colectivă (din AL DOILEA SET DE CURSURI)

Fie A, B, C și I mulțimi nevide (pot fi și vide, dar le vom considera nevide aici), $P \subseteq C \times A$, $R, S \in \mathcal{P}(A \times B)$ și $T, U \in \mathcal{P}(B \times C)$ relații binare, iar $(R_i)_{i \in I} \subseteq \mathcal{P}(A \times B)$ o familie de relații binare de la A la B. Să se demonstreze că:

- $R \circ \emptyset = \emptyset = \emptyset \circ R$ (considerând $\emptyset \subseteq A^2 = A \times A$, $\emptyset \subseteq A \times B$, respectiv $\emptyset \subseteq B^2 = B \times B$, dar observând că, pentru o mulțime arbitrară D, putem considera, pentru prima egalitate, $\emptyset \subseteq D \times A$ în membrul stâng și $\emptyset \subseteq D \times B$ în membrul drept, iar, pentru a doua egalitate, $\emptyset \subseteq B \times D$ în membrul stâng și $\emptyset \subseteq A \times D$ în membrul drept)
- $\emptyset^{-1} = \emptyset$ (considerând $\emptyset$ ca relație binară între două mulțimi fixate)
- **inversa comută cu diferența și cu diferența simetrică:** $(R \setminus S)^{-1} = R^{-1} \setminus S^{-1}$ și $(R \Delta S)^{-1} = R^{-1} \Delta S^{-1}$
- **compunerea este distributivă față de reuniunile arbitrare, la stânga și la dreapta:** $T \circ (\bigcup_{i \in I} R_i) = \bigcup_{i \in I} (T \circ R_i)$ și $(\bigcup_{i \in I} R_i) \circ P = \bigcup_{i \in I} (R_i \circ P)$
- **compunerea (la stânga și la dreapta) păstrează incluziunile nestricte:** $R \subseteq S$ implică $[T \circ R \subseteq T \circ S$ și $R \circ P \subseteq S \circ P]$

- $T \circ (\bigcap_{i \in I} R_i) \subseteq \bigcap_{i \in I} (T \circ R_i)$ și $(\bigcap_{i \in I} R_i) \circ P \subseteq \bigcap_{i \in I} (R_i \circ P)$
- $(T \circ R) \setminus (T \circ S) \subseteq T \circ (R \setminus S)$ și $(R \circ P) \setminus (S \circ P) \subseteq (R \setminus S) \circ P$

Să se dea un exemplu de mulțimi A, B, C și relații binare $R \subseteq A \times B$, $S \subseteq A \times B$ și $T \subseteq B \times C$ care să îndeplinească simultan următoarele condiții:

$$R \not\subseteq S \text{ și } S \not\subseteq R, \text{ dar } T \circ R = T \circ S.$$

Pentru acest exemplu, să se calculeze relațiile binare:

1. $T \circ (R \cap S)$ și $(T \circ R) \cap (T \circ S)$,
2. $(T \circ R) \setminus (T \circ S)$ și $T \circ (R \setminus S)$,
3. $(T \circ S) \setminus (T \circ R)$ și $T \circ (S \setminus R)$,

apoi, la fiecare dintre punctele ①, ② și ③, să se scrie incluziunea strictă care are loc între acele două relații binare.

Temă colectivă (de programare în Prolog)

Să se definească următoarele predicate:

• *functionala*(+*Relatie*), care determină dacă o listă *Relatie* de perechi de constante este funcție parțială, adică dacă relația binară *Relatie* este funcțională, direct cu definiția proprietății de funcționalitate;

• *relfunctionala*(−*Relatie*, +*MultimeDomeniu*, +*MultimeCodomeniu*), care generează în argumentul *Relatie* fiecare funcție parțială de la mulțimea (dată de lista Prolog) *MultimeDomeniu* la mulțimea *MultimeCodomeniu*, folosind predicatul scris la laborator pentru generarea relațiilor binare și selectând dintre relațiile binare între *MultimeDomeniu* și *MultimeCodomeniu* pe cele care satisfac predicatul *functionala* de mai sus;

• *relatiifunctionale*(+*MultimeDomeniu*, +*MultimeCodomeniu*, −*MultimeRelatii*), care generează în argumentul *MultimeRelatii* mulțimea funcțiilor parțiale de la mulțimea *MultimeDomeniu* la mulțimea *MultimeCodomeniu*, folosind predicatul *relfunctionala* de mai sus și predicatul predefinit *setof*;

• *totalape*(+*Relatie*, +*MultimeDomeniu*), care determină dacă o relație binară *Relatie* de la o mulțime *MultimeDomeniu* la o altă mulțime este totală, direct cu definiția proprietății de totalitate pentru relații binare între două mulțimi nu neapărat egale;

• *reltotalape*(−*Relatie*, +*MultimeDomeniu*, +*MultimeCodomeniu*), care generează în argumentul *Relatie* fiecare relație binară totală de la mulțimea *MultimeDomeniu* la mulțimea *MultimeCodomeniu*, folosind predicatul scris la laborator pentru generarea relațiilor binare și selectând dintre relațiile binare între *MultimeDomeniu* și *MultimeCodomeniu* pe cele care satisfac predicatul *totalape* de mai sus;
• *relatiitotalepe*(+*MultimeDomeniu*, +*MultimeCodomeniu*, −*MultimeRelatii*), care generează în argumentul *MultimeRelatii* mulțimea relațiilor binare totale de la mulțimea *MultimeDomeniu* la mulțimea *MultimeCodomeniu*, folosind predicatul *reltotala* de mai sus și predicatul predefinit *setof*;
• *completa*(+*Relatie*, +*Multime*), care determină dacă o relație binară *Relatie* pe o mulțime *Multime* este completă, direct cu definiția proprietății de completitudine pentru relații binare pe o mulțime;
• *relcompleta*(−*Relatie*, +*Multime*), care generează în argumentul *Relatie* fiecare relație binară completă pe mulțimea *Multime*, folosind predicatul scris la laborator pentru generarea relațiilor binare și selectând dintre relațiile binare între *Multime* și ea însăși pe cele care satisfac predicatul *completa* de mai sus;
• *relatiicomplete*(+*Multime*, −*MultimeRelatii*), care generează în argumentul *MultimeRelatii* mulțimea relațiilor binare complete pe mulțimea *Multime* , folosind predicatul *relcompleta* de mai sus și predicatul predefinit *setof*.

Temă colectivă (de programare în Prolog)

Să definească următorul predicat în Prolog, (subînțeles de acum încolo:) împreună cu toate predicatele auxiliare necesare pentru scrierea acestuia, în afară de cele predefinite și cele definite la laborator (sau curs):

- *submultimilinord*(+*Multime*, +*Ordine*, −*MultimeLanturi*), care determină în al treilea argument, pentru un poset (*Multime*, *Ordine*) (subînțeles: cu mulțimea suport *Multime* și relația de ordine *Ordine*), mulțimea submulțimilor mulțimii *Multime* care sunt total ordonate în posetul (*Multime*, *Ordine*), adică raportat la ordinea indusă pe acele submulțimi de ordinea *Ordine* de pe *Multime*.

Desigur, veți genera submulțimile S ale mulțimii *Multime* cu predicatul *sublista*, iar, dintre ele, le veți selecta pe cele care satisfac predicatul *totala*(*Ordine*, S), întrucât predicatul *totala* de la laborator nu necesită înlocuirea ordinii *Ordine* cu, restricția ei la S (adică intersecția ei cu $S \times S$). Am procedat în această manieră în laborator pentru a determina sublaticile (mărginite) care nu sunt lanțuri ale laticii $\mathcal{L}_2 \oplus \mathcal{L}_2^2 \oplus \mathcal{L}_2$.
În locul predicatului *totala* din laborator puteți folosi predicatul *completa* din tema anterioară, pentru că relațiile de ordine sunt reflexive, așadar cele totale coincid cu cele complete.

O proprietate a relațiilor de ordine (pe o mulțime arbitrară) spune că orice relație de ordine pe o mulțime se poate completa la o relație de ordine totală pe acea mulțime. În plus, orice submulțime a unei relații de succesiune (acoperire) pe o mulțime este o relație de succesiune pe acea mulțime. Așadar mulțimea relațiilor de succesiune pe o mulțime este mulțimea submulțimilor relațiilor de succesiune asociate ordinilor totale pe acea mulțime.
Relațiile de ordine totale (liniare) pe o mulțime finită sunt date de permutările acelei mulțimi, astfel: dacă mulțimea respectivă este dată de o listă Prolog fără duplicate A, atunci orice permutare $[X_1, X_2, \ldots, X_n]$ (unde $n \in \mathbb{N}$ este cardinalul mulțimii) a listei A corespunde relației de ordine totale asociate relației de succesiune $[(X_1, X_2), (X_2, X_3), \ldots, (X_{n-1}, X_n)]$, calculate în laborator cu predicatul *succlant*. A se vedea în (bazele de cunoștințe de la) laborator determinarea permutărilor unei liste, cu predicatul *permutare*, și a submulțimilor unei mulțimi, cu predicatul *sublista*.

Temă colectivă (de programare în Prolog)

Să se scrie un predicat *ordinile*(+*Multime*, −*Ordinile*) care determină mulțimea (i.e. lista fără duplicate) *Ordinile* a tuturor relațiilor de ordine pe mulțimea (finită) *Multime*, dată de o listă de constante Prolog fără duplicate, folosind proprietatea de mai sus, astfel:

• ca un caz separat (nu e vorba de nicio recursie), $\emptyset$ este singura relație de ordine pe $\emptyset$, așadar, pentru *Multime* $= [\,]$, vom avea *Ordinile* $= [[\,]]$;

• dacă $A = \{x_1, x_2, \ldots, x_n\}$ este o mulțime de cardinal $n \in \mathbb{N}^*$ (deci finită și nevidă), atunci toate relațiile de ordine pe A sunt:

◯ submulțimile care sunt preordini (și implicit ordini, întrucât submulțimile unei relații binare antisimetrice sunt antisimetrice) ale relațiilor de ordine totale pe A, asociate relațiilor de succesiune $\{(x_{\pi(1)}, x_{\pi(2)}), (x_{\pi(2)}, x_{\pi(3)}), \ldots, (x_{\pi(n-1)}, x_{\pi(n)})\}$ corespunzătoare fiecăreia dintre cele $n!$ permutări π ale mulțimii A (i.e. bijecții $\pi : A \to A$), adică permutări $[x_{\pi(1)}, x_{\pi(2)}, \ldots, x_{\pi(n)}]$ ale listei $[x_1, x_2, \ldots, x_n]$,

sau, echivalent:

□ închiderile reflexive ale submulțimilor care sunt relații tranzitive (și implicit ordini stricte) ale relațiilor de ordine strictă asociate ordinilor totale pe A, așadar relațiilor de succesiune $\{(x_{\pi(1)}, x_{\pi(2)}), (x_{\pi(2)}, x_{\pi(3)}), \ldots, (x_{\pi(n-1)}, x_{\pi(n)})\}$ corespunzătoare permutărilor π ale mulțimii A.

Atenție: • trebuie eliminate duplicatele din lista relațiilor de ordine astfel obținute;

• nu e suficient să considerăm submulțimile relațiilor de succesiune asociate ordinilor totale, apoi să calculăm relațiile de ordine asociate acestor submulțimi; de exemplu, dacă $A = \{a, b, c\}$, cu $|A| = 3$, atunci relația de ordine $\{(a, a), (a, c), (b, b), (b, c), (c, c)\}$ pe A (care determină pe A un poset cu elementele minimale a, b și maximul c) are relația de succesiune asociată $\{(a, c), (b, c)\}$, care nu este submulțime a relației de succesiune asociate niciunei ordini totale pe A, pentru că într–un lanț $(A, \leq)$ nu putem avea $a \neq b$, $a \prec c$ și $b \prec c$, i.e. elementul c nu poate avea doi predecesori diferiți a și b, adică două elemente diferite a și b nu pot avea același succesor c.

A se vedea în laborator predicatele pentru eliminarea duplicatelor dintr–o listă, de exemplu *elimdup*, sau colectarea termenilor cu *setof*. Puteți folosi, pentru varianta ◯ de mai sus, predicatul *ordinepe* de la consultațiile din 28 ianuarie, iar pentru varianta □ predicatul *ordinepe* de la consultațiile din 30 ianuarie; **atenție** la *corectura* din addenda la aceste consultații: selectarea acelor submulțimi care sunt relații tranzitive. În locul ordinii pe termeni folosite în predicatul *eord*, puteți implementa ordinea lexicografică între perechi.

Temă colectivă (de programare în Prolog)

Să se scrie un predicat *latdistrib*(+*Multime*, +*Ordine*) care determină dacă posetul (*Multime*, *Ordine*) este latice Ore distributivă, folosind predicatele din laborator pentru determinarea infimumului și supremumului unei submulțimi a unui poset. Vă amintesc că, cele două legi de distributivitate sunt echivalente (nu pentru fiecare triplet de elemente în parte, ci prima pentru toate tripletele este echivalentă cu a doua pentru toate tripletele), așadar este suficient să verificați una dintre cele două legi.

Temă colectivă (de programare în Prolog)

Să se scrie un predicat *algBoole*(+*Multime*, +*Ordine*) care determină dacă posetul (*Multime*, *Ordine*) este laticea Ore subiacentă unei algebre Boole, folosind predicatul *latdistrib* de mai sus și predicatul din laborator care testează dacă un poset este latice mărginită complementată.

Temă colectivă (din AL PATRULEA SET DE CURSURI)

1. Fie $n \in \mathbb{N}^*$, iar $k_1, \ldots, k_n \in \mathbb{N}$, astfel încât există (cel puțin) un $i \in \overline{1,n}$ cu $k_i > 2$. Să se demonstreze că $\prod_{i=1}^{n} \mathcal{L}_{k_i}$ nu este algebră Boole. (De exemplu, $\mathcal{L}_4^8$ nu e algebră Boole.)
2. Fie I o mulțime nevidă arbitrară, iar $(T_i)_{i \in I}$ o familie de lanțuri (nu neapărat nevide, nu neapărat finite), date prin mulțimile lor suport, astfel încât $(\exists j \in I)\,(|T_j| \notin \{1,2\})$. Să se demonstreze că $\prod_{i \in I} T_i$ nu este algebră Boole.

Indicație: La ① se poate folosi unicitatea descompunerii unei latici în produs direct de lanțuri, rezultată (nu tocmai direct) din indecompozabilitatea lanțurilor (arbitrare, nu neapărat finite) raportat la produsul direct de poseturi, sau se poate demonstra direct cazul general ②, din care rezultă cerința ①. Cerința ② poate fi demonstrată utilizând definițiile pe componente ale operațiilor unei structuri algebrice produs direct, astfel: dacă măcar unul dintre lanțurile din familia $(T_i)_{i \in I}$ nu este mărginit, atunci laticea distributivă $\prod_{i \in I} T_i$ nu este mărginită, iar, dacă toate lanțurile din familia $(T_i)_{i \in I}$ sunt mărginite, atunci laticea distributivă mărginită $\prod_{i \in I} T_i$ nu este complementată, adică are elemente fără complement.