Пролог – 2024/2025

Тодор Дуков

Задача 1. *Нека* $\mathcal{A} = \langle \{a,\, b\},\, \{1,\, \dots,\, n\},\, 1,\, \delta,\, F \rangle$ *е детерминиран тотален краен автомат. Представяне на автомата* $\mathcal{A}$ *в Пролог ще наричаме терма* $(\mathtt{Delta}, \mathtt{FinalStates})$*, където* `Delta` *е списък, който представя графиката на* δ *(тоест списъкът* $[(\mathtt{1}, \mathtt{a}, \delta(\mathtt{1}, \mathtt{a})), (\mathtt{1}, \mathtt{b}, \delta(\mathtt{1}, \mathtt{b})), \dots, (\mathtt{n}, \mathtt{a}, \delta(\mathtt{n}, \mathtt{a})), (\mathtt{n}, \mathtt{b}, \delta(\mathtt{n}, \mathtt{b}))]$*) и* `FinalStates` *е списък с елементи измежду числата* $1, \dots, n$*.*
Да се дефинира на Пролог предикат `minimise_automaton(A, MinA)`*, който при подадено представяне* `A` *на автомат* $\mathcal{A}$ *генерира в* `MinA` *представяне на* $\mathcal{A}_{\min}$*, за който:*

- $\mathcal{A}_{\min}$ *е минимален автомат;*
- $\mathcal{L}(\mathcal{A}) = \mathcal{L}(\mathcal{A}_{\min})$.

Забележка. Можем да улесним съвсем малко задачата, като дадем еднобуквена азбука. Тогава и представянето на δ може да бъде масив с n елемента, който съдържа числа измежду 1 и n.

Задача 2. *Нека* $G = \langle \{a,\, b\},\, 1, \dots, n,\, 1,\, R \rangle$ *е безконтекстна граматика без* ε*-правила, която има безкраен език. Представяне на граматиката* G *в Пролог ще наричаме всеки списък* `Rules`*, който е представя множеството* R *(тоест* $i \to_G \alpha_1 \cdots \alpha_k$ *т.с.т.к.* $(\mathtt{i}, [\alpha_{\mathtt{1}}, \dots, \alpha_{\mathtt{k}}])$ *е елемент на* `Rules`*).*
Да се дефинира на Пролог предикат `derive(Rules, Word)`*, който при подадено представяне* `Rules` *на граматика* G *при преудовлетворяване генерира в* `Word` *всеки списък от вида:*

$$[\alpha_{\mathtt{1}}, \dots, \alpha_{\mathtt{k}}], \text{ където } \alpha_1 \cdots \alpha_k \in \mathcal{L}(G).$$

Забележка. Не е задължително граматиката да генерира безкраен език. Човек може да провери дали една граматика G има безкраен език алгоритмично, като провери дали съществува $\alpha \in \mathcal{L}(G)$, за която $p < |\alpha| < 2p$ (където p е числото от Бар-Хилел лемата), и след това може да ограничи дължината на извода на думата, в случай че езика е краен, иначе генерира всички изводи и филтрира думите. Мисля, че този детайл е най-добре да се спести в полза на студентите.

Задача 3. *Нека* $\mathcal{N} = \langle \{a, b\}, Q, S, \Delta, F\rangle$ *е недетерминиран краен автомат. Представяне на автомата* $\mathcal{N}$ *в Пролог ще наричаме терма* $(\mathtt{Q}, \mathtt{S}, \mathtt{D}, \mathtt{F})$*, където:*

- `Q` *е списък, който представя множеството* Q*;*
- `S` *е списък, който представя множеството* S*;*
- `D` *е списък, който представя множеството* Δ*;*
- `F` *е списък, който представя множеството* F*.*

Да се дефинира на Пролог предикат `convert_nfa_to_total_dfa(N, A)`*, който при подадено представяне* `N` *на автомат* $\mathcal{N}$ *генерира в* `A` *представяне на* $\mathcal{A}$*, за който:*

- $\mathcal{A}$ *е детерминиран тотален краен автомат;*
- $\mathcal{L}(\mathcal{N}) = \mathcal{L}(\mathcal{A})$.

Задача 4. *Дефинираме представянето на регулярен израз* r *над азбуката* $\{a, b\}$ *в Пролог индуктивно:*

- *представянията на* a, b, ε *и* $\varnothing$ *са съответно* `a`, `b`, `eps` *и* `nothing`*;*
- *представянията на* $(r_1 + r_2)$ *и* $(r_1 \cdot r_2)$ *са съответно* `(R1 + R2)` *и* `(R1 * R2)`*, където* `R1` *и* `R2` *са представянията на* r_1 *и* r_2*;*
- *представянето на* r^* *е* `star(R)`*, където* `R` *е представянето на* r*.*

Да се дефинира на Пролог предикат `regex(R)`*, който при преудовлетворяване генерира представянето на всеки регулярен израз.*

Задача 5. *Представяне на едно множество* $\{a_1, \ldots, a_n\}$ *в Пролог е масивът* $[\mathtt{a}_1, \ldots, \mathtt{a}_\mathtt{n}]$*, където* $\mathtt{a}_1, \ldots, \mathtt{a}_\mathtt{n}$ *са представянията на обектите* $a_1, \ldots, a_n$*. Например можем да представим* $\{\varnothing, \{\varnothing\}, \{\varnothing, \{\varnothing\}\}\}$ *като списъка* `[[], [[]], [[], [[]]]]`*. Дефинираме множествата* V_n *по рекурсия така:*

$$V_0 = \varnothing$$
$$V_{n+1} = \mathcal{P}(V_n).$$

Да се дефинира на Пролог предикат `hereditarily_finite_set(S)`*, който при преудовлетворяване генерира в* `S` *представянето на всеки елемент на* $\bigcup\limits_{n<\omega} V_n$*.*

Забележка. Тази задача може малко да се усложни, ако поискаме допълнително да се махнат ординалите, или някой друг интересен клас от множества.

Задача 6. *В Пролог ще представяме релациите като списъци от двойки. Например* $[(1,2),(2,2),(3,1)]$ *представя релацията* $\{\langle 1,\ 2\rangle, \langle 2,\ 2\rangle, \langle 3,\ 1\rangle\}$.
Да се дефинира на Пролог предикат `topo_sort(PO, LO)`*, който при подадено представяне на частична наредба* `PO` *при преудовлетворяване генерира в* `LO` *представянето на всяка линейна наредба, която разширява представяната от* `PO` *наредба над същото поле.*

Задача 7. *Нека* $\langle P, \leq\rangle$ *е ч.н.м.* $F \subseteq P$ *ще наричаме филтър в* $\langle P, \leq\rangle$*, ако:*

- $F \neq \varnothing$;
- *за всяко* $x \in F$ *и за всяко* $y \in P$*, ако* $x \leq y$*, то тогава* $y \in F$*;*
- *за всяко* $x, y \in F$ *съществува* $z \in F$*, за което* $z \leq x$ *и* $z \leq y$*.*

Да се дефинира на Пролог предикат `gen_filter(P, Leq, F)`*, който при подадено представяне* `(P, Leq)` *на някоя частична наредба* $\langle P, \leq\rangle$ *при преудовлетворяване генерира в* `F` *представянето на всеки един филтър в* $\langle P, \leq\rangle$*.*

Забележка. Тази задача може малко да се усложни, ако поискаме да генерират ултрафилтър вместо филтър.