Referáty

- Referáty jsou studentům přiděleny v první polovině semestru, způsobem popsaným na aktuální webovské stránce předmětu, kde je uveden i nejpozdější termín přidělení. Student, kterému případně nebyl referát příslušným způsobem přidělen k danému termínu, se musí bez zbytečného prodlení přihlásit cvičícímu (třeba emailem).

- Rychlé prověření podkladů k referátu s otestováním skutečného porozumění proběhne v termínech ke konci semestru (jak bude sděleno na web-stránce předmětu). Na referátu je nejdůležitější vaše **osobní prezentace**, kdy prokážete, že jste tématu plně porozuměli a prezentaci si pečlivě připravili tak, aby jste podstatu věci stihli kolegům a vyučujícímu rozumně vysvětlit (ilustrovat) v čase **nejvýše 15 minut**. Z technických důvodů nebudete ve skutečnosti 15 minut prezentovat, ale otestujeme, zda jste na to připraveni.

- K tématu můžete čerpat informace a podklady z jakýchkoli veřejných zdrojů, ale musíte je sami za sebe srozumitelně a uceleně písemně zpracovat. **Písemný podklad** nám nezasílejte, přineste si jej k onomu otestování, pak nám ho odevzdáte. Může být jen (vaší) rukou napsaný, s příslušnými obrázky apod. Musí obsahovat dostatečné informace k posouzení, zda je to vámi promyšleně sestavený podklad k prezentaci (a nikoli jen např. překopírované kusy textu z jiných zdrojů).

- **Vyučující k referátům v zásadě nebudou poskytovat vysvětlující konzultace**. Jistě vám neodmítnou krátkou radu, ale nebudou suplovat to, co máte prokázat samostatnou prací.

Referát č. 1

Vysvětlete operaci levého kvocientu pro jazyky, tedy operaci $L_2 \backslash L_1$.

Začněte definicí $\{a\}\backslash L$ (zkráceně psáno $a\backslash L$, kde a je jedno písmeno), pak přejděte k $\{w\}\backslash L$ (zkráceně psáno $w\backslash L$, kde w je jedno slovo), a pak zobecněte na $L_2\backslash L_1$. Vždy ilustrujte malými, ale ne zcela triviálními, příklady.

Vysvětlete, proč pro konečný automat $A = (Q, \Sigma, \delta, q_0, F)$ je jazyk $w\backslash L(A)$ roven jazyku L_q, kde $L_q = \{v \mid q \xrightarrow{v} F\}$, pro takový stav q, pro nějž platí $q_0 \xrightarrow{w} q$.

Ukažte, že pro jakýkoli jazyk $L \subseteq \Sigma^*$ je (levý kvocient) $L\backslash L(A)$ sjednocením jazyků L_q pro vybrané stavy q, a vysvětlete, čím jsou ty vybrané stavy určeny. Demonstrujte také na malém (ale netriviálním) příkladu, v němž zvolte L neregulární.

Referát č. 2

Vysvětlete jádro tzv. Knuth-Morris-Prattova algoritmu (založeného na konečných automatech) pro vyhledávání vzorku v textu a ilustrujte jej na příkladu.

Referát č. 3

Vysvětlete, proč pro každé n existuje nedeterministický automat A_n s n stavy takový, že minimální deterministický konečný automat přijímající $L(A_n)$ má 2^n stavů. (Ilustrujte např. na konkrétním příkladu pro $n = 5$. Ten můžete najít např. ve starším materiálu `http://www.cs.vsb.cz/jancar/TEORET-INF/teoret-inf.pdf`.) (Samozřejmě musíte ukázat, že ve vašem deterministickém automatu jsou všechny stavy dosažitelné a žádné dva různé stavy nejsou ekvivalentní, tedy každé dva různé stavy lze rozlišit nějakým slovem ...) Zdůraznění: máte ukázat pro každé n, nejen pro $n = 5$.

Referát č. 4 (Hltavý algoritmus 1)

Vysvětlete (na vhodně zvoleném případu), jak lze problém (ze studijního textu)

> *Název problému*: *Výběr aktivit*
>
> *Vstup*: množina konečně mnoha aktivit $\{1, 2, \ldots, n\}$ s pevně určenými časovými intervaly (s_1, f_1), (s_2, f_2), ..., (s_n, f_n), kde $(\forall i, 1 \leq i \leq n) : s_i < f_i$
> *Výstup*: množina obsahující největší možný počet vzájemně kompatibilních aktivit (tj. aktivit s vzájemně se nepřekrývajícími intervaly)

řešit hltavým (greedy) algoritmem.

Ukažte myšlenku induktivního důkazu, podle počtu aktivit n, prokazující, že uvedený přístup skutečně vede k optimálnímu řešení.

Referát č. 5 (Hltavý algoritmus 2)
Připomeňte na vhodně zvoleném případu, jak se řeší problém konstrukce minimální kostry grafu hltavým přístupem, a *ilustrujte myšlenku důkazu* toho, že tento přístup skutečně vede k optimu. (Můžete vyjít z popisu ve studijním textu a podle potřeby použít další materiály.)

Referát č. 6 (Dynamické programování)
Algoritmus (Cocke-Younger-Kasami) pro rozpoznávání bezkontextových jazyků (aplikace metody dynamického programování):

Mějme dánu bezkontextovou gramatiku G v tzv. Chomského normální formě, tedy s pravidly pouze typu $\boxed{X \longrightarrow YZ}$ a $\boxed{X \longrightarrow a}$. Algoritmus pro zadané (terminální) slovo w zjistí, zda $w \in L(G)$.

Nástin: Označme $w = a_1 a_2 \dots a_n$. Systematicky vyplňujeme (dvourozměrné) pole D tak, že na závěr bude $D[i,j]$ $(1 \leq i \leq n,\ 0 \leq j \leq n-i)$ obsahovat množinu právě těch neterminálů X, z nichž lze odvodit $a_i a_{i+1} \dots a_{i+j}$.

Vysvětlete tento algoritmus ilustrací na vhodném příkladu a objasněte jeho časovou složitost.
(Další podklady je možno najít např. na
`http://www.cs.vsb.cz/jancar/VYCSLOZ/vycsloz.htm`, referát 4.)

Referát č. 7
Vysvětlete pojem regulárních gramatik (RG) a jejich vztah ke konečným automatům. Na vhodných příkladech ilustrujte převody mezi (N)KA a RG a naopak. (Můžete vyjít z materiálu
`http://www.cs.vsb.cz/jancar/TEORET-INF/teoret-inf.pdf`
a/nebo z jiných zdrojů.)

Referát č. 8 (Redukce bezkontextové gramatiky)
Vysvětlete, co dělá algoritmus popsaný v sekci 3 publikace

> J. Esparza, P. Rossmanith, and S. Schwoon. A uniform framework for problems on context-free grammars. EATCS Bulletin, 72:169-177, October 2000,

která by měla být přístupná na
`http://www7.in.tum.de/um/bibdb/author-esparza.shtml`.
(Ilustrujte na vhodném příkladu.)
Pak stručně vysvětlete, jak je možné tento algoritmus využít při redukci gramatiky (tj. při "Identifying useless variables" na str. 8 zmíněné publikace).

Referát č. 9 (Chomského normální forma bezkontextových gramatik)
Při převodu bezkontextové gramatiky do Chomského normální formy (který je naznačen např. v textu `http://www.cs.vsb.cz/jancar/TEORET-INF/teoret-inf.pdf`)
se po odstranění pravidel typu $A \to \varepsilon$ provádí krok, který odstraňuje pravidla typu $X \to Y$ (neterminál se přepisuje na neterminál). (Pak tedy dostaneme jen pravidla typu $A \to \alpha$, kde α je terminál nebo $|\alpha| \geq 2$, přičemž generovaný jazyk se nezměnil.)
Na vhodném příkladu ilustrujte algoritmus odstraňující ona pravidla $X \to Y$ a *ukažte jeho korektnost.*

Referát č. 10 (Pumping lemma pro regulární jazyky)
Vysvětlete tzv. pumping lemma pro regulární jazyky
(můžete vyjít např. z `http://www.cs.vsb.cz/jancar/TEORET-INF/teoret-inf.pdf`)
a naznačte na příkladu, jak jej lze využít pro důkaz neregularity nějakého jazyka.

Referát č. 11 (Pumping lemma pro bezkontextové jazyky)
Připomeňte pumping lemma pro bezkontextové jazyky (např. ze studijního textu) a vysvětlete souvislost se hrou dvou hráčů popsanou např. v
`http://www.cs.vsb.cz/jancar/TEORET-INF/teoret-inf.pdf`.

Referát č. 12 (Simulace mezi různými variantami Turingových strojů 1)
Důkladně si promyslete, popište a (za pomoci vhodných obrázků) vysvětlete následující konstrukci.

Mějme standardní Turingův stroj (předpokládající oboustranně nekonečnou pásku) $M = (Q, \Sigma, \Gamma, \delta, q_0, F)$. Sestrojme k němu Turingův stroj $M' = (Q', \Sigma, \Gamma', \delta', q_0', F')$, který předpokládá jen jednostranně (tj. pravostranně) nekonečnou pásku—tedy z nejlevější buňky (na níž stojí hlava na počátku) nemůže přejít doleva—a přitom simuluje stroj M.
Naznačíme možný způsob konstrukce:
$Q' = \{q_0', q_1\} \cup \{q_x \mid x \in \Sigma\} \cup \{q_U \mid q \in Q\} \cup \{q_D \mid q \in Q\}$
$\Gamma' = \Sigma \cup (\Gamma \times \Gamma) \cup \{\not\in, \square\}$
$F' = \{q_U \mid q \in F\} \cup \{q_D \mid q \in F\}$
$\delta'(q_0', x) = (q_x, \not\in, +1)$ … pro $x \in \Sigma$
$\delta'(q_x, y) = (q_y, (x, \square), +1)$ … pro $x, y \in \Sigma$
$\delta'(q_x, \square) = (q_1, (x, \square), -1)$ … pro $x \in \Sigma$
$\delta'(q_1, z) = (q_1, z, -1)$ … pro $z \neq \not\in$
$\delta'(q_1, \not\in) = ((q_0)_U, \not\in, +1)$
Obrázkem si znázorněte pásku a (na malém příkladu) počáteční fázi práce stroje M' (pozn.: asi vás napadne pojem ‘dvoustopá páska’); doplňte pak instrukce stroje M' (tedy dodefinujte zobrazení δ') tak, aby skutečně simuloval M. (Ještě kousek nápovědy: U v indexu u stavu znamená ‘up’, D znamená ‘down’).

Referát č. 13 (Simulace mezi různými variantami Turingových strojů 2)
Představte si Turingův stroj pracující na "čtverečkované rovině" (místo lineární pásky). Vstupní slovo je zapsáno na začátku v jednom řádku, čtecí hlava stojí na jeho začátku (ostatní buňky=čverečky obsahují prázdný znak). Obor hodnot přechodové funkce je nyní rozšířen tak, že možné pohyby hlavy jsou Left, Right, Up, Down.
Stručně a srozumitelně popište, jak je možné simulovat tento "rovinný" stroj klasickým "lineárním" strojem. (Nápověda. Musíte tedy popsat, jak bude mít lineární stroj uložen na pásce obsah oné roviny; stačí mít v každém okamžiku zachycen jen obdélník obsahující všechna políčka roviny, která simulovaný stroj dosud navštívil. Pak musíte popsat, jak bude simulující stroj provádět analogii konkrétních instrukcí simulovaného.)

Referát č. 14
V definici modelu RAM v základním studijním materiálu je uvedena hodnota operandu $*i$ jako číslo uložené na adrese, jež je dána součtem čísla i a čísla uloženého v indexregistru.
Jinou užívanou možností nepřímé adresace je použití operandu $\#i$, jehož hodnota je chápána jako číslo uložené na adrese, která je uložena v buňce s adresou i. (I pro tuto možnost se často užívá syntaxe $*i$, my zde pro přehlednější rozlišení používáme $\#i$.)
Ukažte, jak lze RAM-program v jedné variantě simulovat RAM-programem v druhé variantě a naopak. Popište tedy přirozený překlad programu užívající typ $*i$ na program užívající typ $\#i$ a naopak. (Ilustrujte na jednoduchém příkladu.)

Referát č. 15 (Rozhodnutelnost a nerozhodnutelnost)
Uvažujme problém

NÁZEV: UHP (*Uniform Halting Problem*)

VSTUP: Turingův stroj M.

OTÁZKA: Zastaví se M na každý vstup?

Zjistěte, zda tento problém je rozhodnutelný či nerozhodnutelný, a své zjištění prokažte. V případě rozhodnutelnosti problému ukažte algoritmus, který jej řeší; v případě nerozhodnutelnosti můžete vyjít z nerozhodnutelnosti problému zastavení a ukázat příslušnou převeditelnost. (Můžete např. vyjít ze stručné zmínky v textu
`http://www.cs.vsb.cz/jancar/TEORET-INF/teoret-inf.pdf`.)

Referát č. 16 (Převeditelnost mezi problémy)
Demonstrujte myšlenku převeditelnosti IPKP (iniciálního Postova korespondenčního problému) na PKP (Postův korespondenční problém). (Můžete vyjít např. z příslušné animace, zvolte si ale jiný příklad, na němž myšlenku srozumitelně předvedete a vysvětlíte.)

Referát č. 17 (Polynomiální převeditelnost)
Jedna z animací k předmětu ukazuje polynomiální převeditelnost problému nezávislé množiny na problém hamiltonovského cyklu.
Je to technicky netriviální konstrukce (která vyšla z Referátu 6 na `http://www.cs.vsb.cz/jancar/VYCSLOZ/vycsloz.htm`).
Prostudujte ji a prezentujte na co nejjednodušším konkrétním případu, který ještě umožňuje rozumnou demonstraci hlavní myšlenky.

Referát č. 18 (Savitchova věta)
Popište konstrukci v důkazu Savitchovy věty. (K nastudování můžete např. využít podklad k referátu č. 8 na `http://www.cs.vsb.cz/jancar/VYCSLOZ/vycsloz.htm`.)
Můžete se ovšem omezit na tento speciální případ:

> Je-li problém P rozhodován nedeterministickým Turingovým strojem s prostorovou složitostí n, pak je také rozhodován deterministickým Turingovým strojem s polynomiální prostorovou složitostí.

Máte tedy vysvětlit, jak lze k tzv. lineárně omezenému automatu (linear bounded automaton) M,

> tj. k nedeterministickému Turingovu stroji M, který při výpočtu na vstupním w, $|w| = n$, nenavštíví jiná políčka než ta, na nichž je zapsán vstup (a má tedy prostorovou složitost n)

navrhnout (deterministický) algoritmus A, který pro zadané w zjistí, zda M má přijímající výpočet pro w (tedy zda $w \in L(M)$). Algoritmu A přitom musí stačit polynomiálně omezená paměť.
(Připomenutí. Počet konfigurací délky n stroje M je omezen hodnotou c^n, kde konstantu c lze snadno spočítat z velikosti (stavové množiny a abecedy) stroje M. Délka nejkratšího přijímajícího výpočtu M nad w, $|w| = n$, (pokud takový existuje) je tedy také omezena oním c^n.)
(Bylo by dobré ukázat, že ta prostorová složitost A se dá omezit kvadraticky, je v $O(n^2)$, a naznačit, proč A lze přímočaře implementovat deterministickým Turingovým strojem s prostorovou složitostí $O(n^2)$.)

Referát č. 19 (Problém QBF; Quantified Boolean Formulas)
Uvažujme problém

> *Název*: QBF (*problém pravdivosti kvantifikovaných booleovských formulí*)
>
> *Vstup*: formule $(\exists x_1)(\forall x_2)(\exists x_3)(\forall x_4)\ldots(\exists x_{2n-1})(\forall x_{2n})\mathcal{F}(x_1, x_2, \ldots, x_{2n})$, kde $\mathcal{F}(x_1, x_2, \ldots, x_{2n})$ je booleovská formule v konjunktivní normální formě.
> *Otázka*: je daná formule pravdivá ?

Navrhněte algoritmus, který řeší problém QBF a má prostorovou složitost omezenou polynomem. (Tím ukážete, že QBF je v PSPACE.)
Návod. Řekneme, že formule $\mathcal{F}(x_1, x_2, \ldots, x_{2n})$ je OK pro posloupnost booleovských hodnot $b_1, b_2, \ldots, b_i$, kde $0 \leq i \leq 2n$, jestliže

> buď $i = 2n$ a $\mathcal{F}(b_1, b_2, \ldots, b_{2n}) = true$,
>
> nebo $i < 2n$, i je liché a $\mathcal{F}$ je OK jak pro $b_1, b_2, \ldots, b_i, true$, tak pro $b_1, b_2, \ldots, b_i, false$,
>
> nebo $i < 2n$, i je sudé a $\mathcal{F}$ je OK pro alespoň jednu z posloupností $b_1, b_2, \ldots, b_i, true$ a $b_1, b_2, \ldots, b_i, false$.

Ověřte nejprve, že formule $(\exists x_1)(\forall x_2)(\exists x_3)(\forall x_4)\ldots(\exists x_{2n-1})(\forall x_{2n})\mathcal{F}(x_1, x_2, \ldots, x_{2n})$ je pravdivá právě tehdy, když $\mathcal{F}$ je OK pro prázdnou posloupnost.
Pak sestavte kýžený algoritmus (a prokažte, že jeho prostorová [tedy paměťová] složitost je polynomiální).

Referát č. 20 (Oblázková hra v PSPACE)
Uvažujme problém, jehož instancí je orientovaný graf s vybraným vrcholem v a dále k 'oblázků'. Můžeme v jakémkoli pořadí provádět následující elementární kroky:

- na vrchol x můžeme položit oblázek, pokud v daný okamžik leží oblázky na všech vrcholech, z nichž vede hrana do x,
- oblázek položený na vrchol můžeme odebrat (a znovu použít později).

Otázkou je, zda existuje posloupnost kroků, při níž položíme oblázek na zadaný vrchol v.
Prokažte, že problém je v PSPACE.
(Jednou z motivací problému je problém přidělování paměti při výpočtu; stačí daný počet registrů k provedení určeného výpočtu ?)