IZLO – Projekt 1: SAT solvery

Úvod

V projektu uvažujeme problém studenta, který se snaží naplánovat pořadí, ve kterém si zapíše předměty během studia. Předměty však mezi sebou mají řadu prerekvizit a není jasné zda vůbec existuje způsob jak si všechny zvolené předměty zapsat tak, aby byly všechny prerekvizity splněny. Problém budeme řešit pomocí převodu na splnitelnost ve výrokové logice a následného využití SAT solveru.

Zadání

Instancí problému je číslo $P > 0$ udávající počet zapsaných předmětů, číslo $S > 0$ udávající počet semestrů a množina prerekvizit mezi předměty. Jak předměty, tak semestry jsou reprezentovány pomocí čísel v rozsahu $0, \ldots, P - 1$ pro předměty a $0, \ldots, S - 1$ pro semestry. Prerekvizita je dvojice (a, b) udávající, že pro zapsání předmětu b v nějakém semestru i je potřeba, aby předmět a byl zapsán v semestru j, takovém, že $j < i$. Dále je potřeba zajistit, že každý předmět je zapsán právě v jednom semestru.

Vaším cílem je vytvořit program, který pro instanci tohoto problému vygeneruje formuli, která je splnitelná právě tehdy, když problém má řešení. Navíc musí platit, že každý model vygenerované formule splňuje tři níže uvedené podmínky. K dispozici již máte kostru (k dispozici níže), která se stará o zpracování vstupu a vygenerování formule ve formátu DIMACS (popis např. zde). Vaším úkolem je doplnit kód tří funkcí v souboru `code/add_conditions.c`, které se starají o generování následujících podmínek:

1. **Každý předmět je zapsán alespoň jednou.** Generování této formule je potřeba doplnit do funkce `each_subject_enrolled_at_least_once(...)`.
2. **Každý předmět je zapsán nejvýše jednou.** Generování této formule je potřeba doplnit do funkce `each_subject_enrolled_at_most_once(...)`.
3. **Splnění prerekvizit.** Formuli zajišťující, že všechny prerekvizity jsou splněny, je potřeba vygenerovat ve funkci `add_prerequisities_to_formula(...)`. Pole obsahující prerekvizity je jedním z parametrů této funkce. Prerekvizita je struktura obsahující položky `earlier_subject` a `later_subject`. Pro splnění prerekvizity je potřeba, aby byl `earlier_subject` zapsán dříve než `later_subject`.

Výše zmíněné funkce jsou jediné části kódu, které mají být modifikovány.

Kostra

Kostra projektu je ke stažení zde

Formát vstupu

Instance problému je popsána v následujícím textovém formátu. Na první řádce souboru se nachází hlavička obsahující přesně tři přirozená čísla udávající postupně celkový počet předmětů, celkový počet semestrů a celkový počet prerekvizit. Následuje seznam prerekvizit, kde se na každém řádku nachází dvojice čísel (v rozsahu vymezeném hlavičkou):

```
<pocet predmetu> <pocet semestru> <pocet prerekvizit>

<earlier_1> <later_1>
<earlier_2> <later_2>
...
```

kde `<earlier_x> <later_x>` značí prerekvizitu $(\text{earlier}_x, \text{later}_x)$. Konkrétní příklad vstupu je následující soubor:

```
3 2 3

0 1
1 2
2 0
```

Hlavička udává, že je potřeba zapsat 3 předměty ve 2 semestrech. Dále musí platit, že předmět `0` je potřeba zapsat dříve než předmět `1`, předmět `1` dříve než předmět `2` a předmět `2` dříve než předmět `0`. Tento vstup nemá řešení jelikož prerekvizity mezi předměty tvoří cyklus.

Generování formulí

Při řešení máte k dispozici proměnné ve tvaru $x_{p,s}$ kde $0 \leq p < P$ je index předmětu a $0 \leq s < S$ je index semestru. Sémantika proměnných je následující:

$$x_{p,s} = \begin{cases} 0 \Leftrightarrow & \text{předmět } p \text{ není zapsaný v semestru } s, \\ 1 \Leftrightarrow & \text{předmět } p \text{ je zapsaný v semestru } s. \end{cases}$$

Jelikož formát DIMCAS pracuje s proměnnými indexovanými přirozenými čísly, proměnná $x_{p,s}$ je převedena na promměnnou x_i, kde $i = p \cdot S + s + 1$. S těmito proměnnými však nebudete pracovat přímo, ale pomocí funkcí popsaných v následujícím odstavci. Nicméně způsob reprezentace proměnných se může hodit, pokud si budete chtít projít vygenerovaný DIMACS soubor (ve vygenerovaném souboru jsou na řádcích začínajících symbolem `c` komentáře, kde je kopie vstupního problému a popis mapování čísel proměnných na předměty a semestry).

Všechny formule ze zadání je potřeba vygenerovat v konjunktivní normální formě (CNF), která je standardním vstupem SAT solverů. Pro reprezentaci formule jsou již v kostře projektu vytvořeny potřebné struktury. Pro manipulaci s těmito strukturami jsou k dispozici následující funkce:

- `Clause *create_new_clause(unsigned num_of_subjects, unsigned num_of_semesters)` – Vytvoří novou klauzuli. Parametry funkce jsou celkový počet předmětů a celkový počet semestrů.
- `void add_clause_to_formula(Clause *c, CNF *f)` – Vloží klauzuli `c` do formule `f`.
- `void add_literal(Clause *c, bool is_positive, int p, int s)` – Vloží do klauzule `c` literál $x_{p,s}$ (pokud `is_positive = true`), nebo $\neg x_{p,s}$ (pokud `is_positive = false`).

Ukázku použití těchto funkcí lze nalézt v souboru `code/add_conditions.c` ve funkci `example_condition`. Tato funkce generuje následující formuli:

$$\bigwedge_{0 \leq p < P} \bigwedge_{0 \leq s < S} (x_{p,s} \vee \neg x_{p,s})$$

Testování

Referenčním SAT solverem je MiniSat[1]. Na linuxových systémech založených na Debianu lze tento nástroj nainstalovat jednoduše příkazem `apt-get install minisat`, na Windows lze použít stejný postup ve WSL[2], na macOS lze použít `brew install minisat`. Pro vyzkoušení lze také využít online rozhraní (pak ale nelze využít automatické testy níže).

K ověření základní funkčnosti vašeho řešení je možné použít dva přiložené skripty. Tyto skripty vyžadují `python3` a nainstalovaný MiniSat dostupný v `PATH`. Po přeložení vašeho řešení je možné využít skript `run.sh`, který spustí vaše řešení nad vstupním souborem a následně spustí SAT solver nad vygenerovanou formulí. V případě, že vygenerovaná formule je splnitelná, skript vypíše model jako lidsky čitelné řešení problému (rozdělení předmětů mezi semestry) a zkontroluje jestli model splňuje podmínky specifikované zadáním. V případě, že ne, je vypsána první porušená podmínka. Příkazem `make test` je možné ověřit základní funkcionalitu vašeho řešení na jednoduchých vstupech. Tyto vstupy lze nalézt ve složkách `tests/sat` a `tests/unsat`.

Další pokyny a doporučení

- Potřebné formule jdou vytvořit téměř přímo v CNF, nemělo by být potřeba aplikovat distributivní zákony pro úpravu do CNF. Před samotným programováním si zkuste formule nejprve napsat na papír.
- Odevzdáváte pouze soubor `add_conditions.c` do IS VUT.
- Své řešení vypracujte samostatně. Odevzdané projekty budou kontrolovány proti plagiátorství, za nějž se považuje i sdílení vlastního řešení s jinými lidmi.
- Případné dotazy směřujte do fóra “Diskuze k projektům”.

---

1. http://minisat.se/↩
2. https://ubuntu.com/wsl↩