IZLO – Projekt 2: SMT solvery

Úvod

Poté, co si náš známý student z prvního projektu úspěšně naplánoval předměty s využitím SAT solveru, dostal hlad a rozhodl se vyrazit na oběd do menzy. Když dorazil na místo, byl velmi překvapen širokou nabídkou jídel (především různých variant kuřecích plátků) a zmocnila se ho rozhodovací paralýza. Nakonec usoudil, že jediným možným způsobem, jak ideálně vybrat oběd, je aplikovat na tento problém logiku. Jelikož už byl unaven kódováním problémů do výrokové logiky, rozhodl se zvolit expresivnější prvořádovou logiku a využít SMT solver.

Zadání

Vstupem problému je pět jídel reprezentovaných čísly $1, 2, \ldots, 5$, kde každé z jídel má následující parametry nabývající hodnot z přirozených čísel (včetně 0):

- Cena c_i, kterou je potřeba zaplatit za jednu porci daného jídlo.
- Kalorická hodnota k_i jedné porce daného jídla (čím vyšší, tím lepší).
- Počet porcí m_i daného jídla k dispozici.

Cílem je najít vhodnou kombinaci jídel s co největší kalorickou hodnotou, přičemž si student může objednat více porcí jednoho druhu jídla. Tato kombinace musí splňovat následující podmínky:

1. Celková cena obědu je maximálně max_cena.
2. Požadovaný počet porcí každého jídla nepřesahuje maximální počet.
3. Celkový součet kalorií je *maximální*, jinými slovy, neexistuje jiné řešení, které by splňovalo podmínky `1` a `2` a mělo vyšší celkový součet kalorií.

Řešení

Vaším úkolem je do vyznačeného místa v souboru `projekt2.smt2` doplnit formule, které zajistí, že:

1. Každá proměnná n_i je interpretována jako objednaný počet porcí jídla i.
2. Proměnná $best$ je interpretována jako *optimální* kalorická hodnota.

Tyto promměné přitom musí splňovat podmínky uvedené v předchozí sekci.

Při řešení byste si měli vystačit s SMT-LIB příkazy `declare-fun` (pro deklaraci proměnných) a `assert` (pro přidání formulí, které mají být splněny). V těchto formulích si vystačíte s booleovskými spojkami (`and`, `or`, `not`, `=>`, …), kvantifikátory (`forall`, `exists`) a termy v teorii celočíselné aritmetiky. Tato teorie obsahuje klasické funkce nad čísly (`+`, `-`, `*`, `div`, `mod`, `abs`) a predikáty pro porovnání čísel (`=`, `>`, `<`, `<=`, `>=`). Příklady použití lze nalézt níže, případně na stránce předmětu.

Referenčním (a doporučeným) SMT solverem je cvc5[1]. Pro řešení projektu zcela stačí použít jeho online rozhraní. Alternativně lze solver stáhnout jako binárku z githubu, případně je k dispozici na serveru Merlin, kde ji lze použít pomocí příkazu:

```
/usr/local/groups/verifikace/IZLO/cvc5-Linux projekt2.smt2
```

Formát SMT-LIB

Formát SMT-LIB využívá prefixový zápis operátorů. Term `x + y = x * y` je tedy zapsán jako `(= (+ x y) (* x y))`. Při deklaraci proměnných je potřeba uvést její tzv. *sort* (podobně jako typ proměnné například v jazyce C), v tomto projektu budeme pracovat pouze s celočíselnými proměnnými a odpovídajícím sortem `Int`.

```
; Nastavení teorie, ve které má SMT solver pracovat.
; ALL značí všechny teorie podporované solverem
(set-logic ALL)

; Nastavení parametrů SMT solveru
(set-option :produce-models true)

; Deklarace celočíselný konstant jako nulární funkce)
(declare-fun x () Int)
(declare-fun y () Int)
(declare-fun z () Int)

; Deklarace binární funkce na celých číslech
(declare-fun c (Int Int) Int)

; Deklarace binárního predikátu na celých číslech
(declare-fun pred (Int Int) Bool)

; Přidání omezení reprezentující formuli (x * y = 0) → (x = 0 ∨ y = 0)
(assert
  (=>
    (= (* x y) 0)
    (or (= x 0) (= y 0))
  )
)

; Přidání dalšího omezení reprezentující formuli ∀a ∀b (∃c (a + b = c))
(assert
  (forall ((a Int) (b Int))
    (exists ((c Int))
      (= (+ a b) c)
    )
  )
)

; Ověření splnitelnost konjunkce všech omezení
(check-sat)

; Pokud je status sat, vypíše model
(get-model)

; Pokud je status sat, vypíše hodnoty termů x, y a x + y
(get-value (x y (+ x y)))
```

Kostra

Kostra projektu je ke stažení zde. Odevzdejte tento soubor, s řádkem “Zde doplnte vase reseni” nahrazeným Vaším řešením. **Pozor! Nemodifikujte nic jiného, jinak Vám automatické testy zbytečně strhnou body.**

Testování

Soubor s kostrou je doplněn několika testovacími vstupy, které fungují následujícím způsobem. Každý test je rozdělen do dvou částí:

- **a)** Ověří zda pro vstupní parametry existuje řešení a vypíše jej. Očekávaný status je `sat`.
- **b)** Ověří zda pro vstupní parametry neexistuje jiné řešení. Očekáváný status je `unsat`.

Skript samotný nekontroluje správnost výstupů. Jednotlivé testy jsou implementované pomocí příkazu `(check-sat-assuming (formulae))`, který ověří splnitelnost globálních omezení (definovaných pomocí `assert`) v konjunci s omezeními `formulae`. Pro debugování modelů v jednotlivých testech lze využít příkaz `(get-value (terms))`.

Další pokyny a doporučení

- V případě špatného nebo příliš komplikovaného řešení se může stát, že se solver „zasekne“. Při správném řešení by měl solver doběhnout během několika sekund.
- Odevzdáváte pouze soubor `projekt2.smt2` 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. https://github.com/cvc5/cvc5↩