J. Hoenicke
A. Nutz

12.01.2016
submit until 19.01.2016, 14:15

Tutorials for Decision Procedures
Exercise sheet 10

Exercise 1: Correctness of DP for $T_{\mathsf{A}}^{\mathbb{Z}}$

Let I be an interpretation. Prove for $F[\bar{i}] : expr \leq expr$ that $I \models F[\bar{i}] \;\rightarrow\; F[\bar{t}]$, where $\bar{i} = (i_1, \ldots, i_n)$ and $\bar{t}$ is the vector $\bar{t} = (t_1, \ldots, t_n) \in \mathcal{I}^n$ with $\alpha_I[t_k] = proj_{\mathcal{I}}(\alpha_I[i_k])$ (in the notation of the book $\bar{t} = \mathsf{proj}_I(\bar{i})$). The expression $expr$ is either a universal variable i_k or a $pexpr$. Note that $\mathcal{I}$ contains all $pexpr$ and that

$$proj_{\mathcal{I}}(v) = \begin{cases} max\{\alpha_I[t] \mid t \in \mathcal{I} \;\wedge\; \alpha_I[t] \leq v\} & \text{if for some } t \in \mathcal{I}\text{: } \alpha_I[t] \leq v \\ min\{\alpha_I[t] \mid t \in \mathcal{I}\} & \text{otherwise} \end{cases}$$

Exercise 2: Nelson-Oppen

Apply the deterministic version of Nelson-Oppen to the following $T_{\mathsf{E}} \cup T_{\mathbb{Q}}$-formulae:

(a) $x + y = z \;\wedge\; f(z) = x + y \;\wedge\; f(f(x + y)) \neq z$.

(b) $g(x{+}y, z) = f(g(x, y)) \wedge x{+}z = y \wedge z \geq 0 \wedge x \geq y \wedge g(x, x) = z \wedge f(z) \neq g(2x, 0)$

Exercise 3: DPLL(T)

In the last lecture we presented the CDCL algorithm in the form of the six rules Decide, Propagate , Conflict , Explain, Learn, Backtrack.
In the lecture on propositional logic we presented the same algorithm as a functional program (printed below).
Which lines of the functional code correspond to which of the six rules? (There may not always be an exact correspondence, in such cases please add a short explanation.)

```
let rec DPLL =
  let PROP U =
    let ℓ = CHOOSE U ∩ unassigned in
    val[ℓ] := ⊤
    let C = DPLL in
    if (C = satisfiable)
      satisfiable
    else
      val[ℓ] := undef
      if (ℓ̄ ∉ C) C
      else U \ {ℓ} ∪ C \ {ℓ̄}
  if conflictclauses ≠ ∅
    CHOOSE conflictclauses
  else if unitclauses ≠ ∅
    PROP (CHOOSE unitclauses)
  else if coreclauses ≠ ∅
     let ℓ = CHOOSE (⋃coreclauses) ∩ unassigned in
     val[ℓ] := ⊤
     let C = DPLL in
     if (C = satisfiable) satisfiable
     else
        val[ℓ] := undef
        if (ℓ̄ ∉ C) C
        else LEARN C; PROP C
  else satisfiable
```