Automated Reasoning I, 2019/20
Re-Exam, Sample Solution

Assignment 1

$$F \;=\; \exists z\, \forall x\, \Big(\big(\exists y\, P(z,y)\big) \leftrightarrow \big(\exists y\, Q(x,y)\big) \Big)$$

Since the equivalence occurs in F with positive polarity, we replace it by a conjunction of two implications.

$$\begin{aligned} \exists z\, \forall x\, \Big(\Big(\big(\exists y\, P(z,y)\big) \rightarrow \big(\exists y\, Q(x,y)\big) \Big) \\ \wedge \Big(\big(\exists y\, Q(x,y)\big) \rightarrow \big(\exists y\, P(z,y)\big) \Big) \Big) \end{aligned}$$

Then we replace the implications by disjunctions:

$$\begin{aligned} \exists z\, \forall x\, \Big(\Big(\neg\big(\exists y\, P(z,y)\big) \vee \big(\exists y\, Q(x,y)\big) \Big) \\ \wedge \Big(\neg\big(\exists y\, Q(x,y)\big) \vee \big(\exists y\, P(z,y)\big) \Big) \Big) \end{aligned}$$

After pushing the negations inward, we obtain the negation normal form

$$\begin{aligned} \exists z\, \forall x\, \Big(\Big(\big(\forall y\, \neg P(z,y)\big) \vee \big(\exists y\, Q(x,y)\big) \Big) \\ \wedge \Big(\big(\forall y\, \neg Q(x,y)\big) \vee \big(\exists y\, P(z,y)\big) \Big) \Big) \end{aligned}$$

We can now use miniscoping. The quantifier $\forall x$ can be pushed inward

$$\begin{aligned} \exists z\, \Big(\Big(\big(\forall y\, \neg P(z,y)\big) \vee \big(\forall x\, \exists y\, Q(x,y)\big) \Big) \\ \wedge \Big(\big(\forall x\, \forall y\, \neg Q(x,y)\big) \vee \big(\exists y\, P(z,y)\big) \Big) \Big) \end{aligned}$$

but the quantifier $\exists z$ cannot. Variable renaming yields

$$\begin{aligned} \exists z\, \Big(\Big(\big(\forall y\, \neg P(z,y)\big) \vee \big(\forall x\, \exists y'\, Q(x,y')\big) \Big) \\ \wedge \Big(\big(\forall x'\, \forall y''\, \neg Q(x',y'')\big) \vee \big(\exists y'''\, P(z,y''')\big) \Big) \Big) \end{aligned}$$

After Skolemization (starting with the outermost existential quantifier), we obtain

$$\begin{aligned} \Big(\big(\forall y\, \neg P(c,y)\big) \vee \big(\forall x\, Q(x,f(x))\big) \Big) \\ \wedge \Big(\big(\forall x'\, \forall y''\, \neg Q(x',y'')\big) \vee P(c,c') \Big) \end{aligned}$$

Finally we push the remaining quantifiers outward:

$$\begin{aligned} \forall y\, \forall x\, \forall x'\, \forall y'' \Big(\big(\neg P(c,y) \vee Q(x,f(x))\big) \\ \wedge \big(\neg Q(x',y'') \vee P(c,c')\big) \Big) \end{aligned}$$

The resulting formula in CNF is equisatisfiable to F, but not equivalent, since the Skolemization step does not yield an equivalent formula.

Grading scheme: -2 points per error.

Assignment 2

Part (a)

$$\begin{aligned} &P(f(x), f(x)) && (1)\\ &P(g(x), g(x)) && (2)\\ &P(h(x), h(x)) \;\vee\; P(h(y), h(b)) && (3)\\ &\neg P(f(x), y) \;\vee\; \neg P(x, y) \;\vee\; \neg P(y, g(x)) && (4)\\ &\neg P(x, y) \;\vee\; \neg P(b, c) && (5) \end{aligned}$$

The second literal of clause (4) is *not* maximal, since it is strictly smaller than the first literal of (4) in the given ordering. All other literals in the clauses (1)–(5) are maximal in their clauses.

Grading scheme: -1 point per error.

Part (b) When the second literal in (5) is selected, we get the following three $\mathrm{Res}^{\succ}_{sel}$ inferences:

Resolution (1) literal 1, (4) literal 1
(after renaming x in (4) to x'):
mgu $\{x' \mapsto x,\, y \mapsto f(x)\}$,
conclusion $\neg P(x, f(x)) \;\vee\; \neg P(f(x), g(x))$.

Resolution (2) literal 1, (4) literal 3
(after renaming x in (4) to x'):
mgu $\{x' \mapsto x,\, y \mapsto g(x)\}$,
conclusion $\neg P(f(x), g(x)) \;\vee\; \neg P(x, g(x))$.

Factorization (3) literals 1 and 2:
mgu $\{x \mapsto b,\, y \mapsto b\}$,
conclusion $P(h(b), h(b))$.

Grading scheme: 3 + 4 + 3 points for three inferences.

Assignment 3

The statement holds. Proof: Assume that there is a variable $x \in X$ such that $[x] \neq \{x\}$. Since $x \in [x]$, this means that $[x]$ must contain some term t different from x. Therefore $E \vdash x \approx t$, and by Birkhoff's Theorem, this implies $x \leftrightarrow_E^* t$. Since t is different from x, we have $x \leftrightarrow_E^+ t$, and therefore $x \leftrightarrow_E t' \leftrightarrow_E^* t$ for some term t'. Consequently, $x \rightarrow_E t'$ or $t' \rightarrow_E x$. So some subterm of x must be equal to either $s\sigma$ or $s'\sigma$ for some equation $s \approx s'$ in E. This is impossible, though, since neither s nor s' is a variable.

An alternative proof uses induction over the derivation tree for $E \vdash t \approx t'$ to show that no statemenn $E \vdash x \approx t$ with $t \neq x$ can be derived.

Assignment 4

The relation $\succ$ is irreflexive, transitive, and well-founded. It is not compatible with contexts, since

$$f(b) \succ b,$$

but not

$$g(h(h(b)), f(b)) \succ g(h(h(b)), b).$$

It is also not stable under substitutions, since

$$g(x, h(h(h(b)))) \succ g(h(h(x)), h(h(b))),$$

but not

$$g(f(f(c)), h(h(h(b)))) \\ \succ g(h(h(f(f(c)))), h(h(b))).$$

Assignment 5

Part (a) There are many possible Knuth-Bendix orderings $\succ$ such that $\rightarrow_R \subseteq \succ$. One possibility: $w(h) = 5$, $w(f) = 3$, $w(g) = w(b) = w(c) = w(x) = 1$; in this case the precedence does not matter.

Part (b) There are two critical pairs:

Critical pair between (1) and (1):

$$\langle h(f(c,c), b),\ f(h(c,b), f(c,c))\rangle$$

Both terms are in normal form, therefore not joinable.

Critical pair between (1) and (2):

$$\langle h(b,b),\ g(f(b,c))\rangle$$

$h(b,b)$ can be rewritten to $g(f(b,c))$ using (3), therefore joinable.

Grading scheme: 5 points if a critical pair was computed correctly; 2 points if it was detected correctly but computed incorrectly.

Assignment 6

To give a recursive definition for F_n, we need an auxiliary formula G_n over $\{P_1, \dots, P_n\}$ such that $\mathcal{A}(G_n) = 1$ if and only if $\mathcal{A}$ maps all propositional variables $P_1, \dots, P_n$ to 0. Then we have

$$\begin{aligned} F_0 &\mathrel{|}\!\models\!\mathrel{|} \bot \\ G_0 &\mathrel{|}\!\models\!\mathrel{|} \top \\ F_n &\mathrel{|}\!\models\!\mathrel{|} \text{if } P_n \text{ then } G_{n-1} \text{ else } F_{n-1} \\ G_n &\mathrel{|}\!\models\!\mathrel{|} \text{if } P_n \text{ then } \bot \text{ else } G_{n-1} \end{aligned}$$

for $n \geq 1$. (The if-then-else construct can be encoded using the usual boolean connectives as shown in the lecture notes.)

The recursive definition can be translated directly into a reduced OBDD: The OBDD has $2n + 1$ nodes: one node labelled with P_n (corresponding to the formula F_n), two nodes labelled with P_i for every $i \in \{1, \dots, n-1\}$ (corresponding to F_i and G_i), and two leaf nodes: