CSC 438, solutions for HW2

Fall 2019

1. To show that $B \equiv \forall x (fx = x)$ is not a logical consequence of $A \equiv \forall x (gx = x)$, we will define a structure $M$ such that $M \models \forall x (gx = x)$ but $M \not\models \forall x (fx = x)$. Understanding the meaning of these sentences is helpful in defining the right structure. (Of course, if $A \not\models B$, then $\{A, \neg B\}$ is satisfiable and using the proof of completeness theorem we can define a suitable structure, but such a structures would be very complicated and proving things about it will be hard, so it is much simpler to understand the meaning of these sentences and use this information to build a simpler structure.)

$A$ means that $f$ is left inverse of $g$ and $B$ means that $f$ is right inverse of $g$, so we are looking for two functions $f$ and $g$ such that $f$ is left inverse of $g$ but not right inverse of $g$. Having a left inverse is equivalent to being injective (one-to-one) and having a right inverse is equivalent to being surjective (onto), therefore we want a injective but not surjective function, like successor function over natural numbers. Now we can define our structure. Let universe of $M$ be the set of natural numbers $\mathbb{N}$, $f(x) = \begin{cases} x - 1 & 1 \leq x \\ 0 & 0 = x \end{cases}$, and $g(x) = x + 1$. Since $fgx = f(x + 1) = x$, we have $M \models \forall x (gx = x)$. On the other hand $gf0 = g0 = 1$ and therefore $M \not\models \forall x (fx = x)$ as required.

2. We want to define a sentence $A$ in the language $L = \{ , R, =\}$ such the set $\{ |M| : M \models A \text{ and } M \text{ has a finite universe} \}$ (which is called finite spectrum of $A$, $\text{spec}(A)$) is the set of even numbers. The hint states that we can think of $R$ as a pairing relation. In these kind of problems thinking of $R$ as an equivalence relation is also helpful. We want to define $R$ to be an equivalence relation such that each equivalence class has exactly two elements (therefore it can be thought of a pairing relation.) To make $R$ an equivalence relation, we need to have $\forall x (xRx)$, $\forall xy (xRy \supset yRx)$, $\forall xyz (xRy \land yRz \supset xRz)$. We need to add the fact that each equivalence class has exactly two elements, i.e., for each element $x$, there is exactly one other element $y$ such that $xRy$: $\forall x, \exists! y (xRy \land x \neq y)$. Let $A$ be conjunction of these four sentences.

To prove that $\text{spec}(A)$ is the set of even numbers we have to show two claims: first, the size of every finite model of $A$ is even; second, any even number is size of some finite model of $A$.

Let $M$ be a finite model of $A$. It is easy to see that $R^M$ is an equivalence relation over $M$. Equivalence classes partition $M$. It only remains to show that each equivalence
class has exactly two elements. Let \([a]\) be an equivalence class of \(R^M\), where \(a \in M\). By \(A\), there is a unique element \(b \in [a]\) such that \(b \neq a\). Therefore \([a]\) has exactly two elements. Since each class has exactly two elements, the size of \(M\) is twice the number of equivalence classes, which is an even number. This proves that \(\text{spec}(A) \subseteq \text{Even}\).

Let \(n\) be an arbitrary positive number. Let the universe of \(M\) be \([0, 2n - 1] = \{0, 1, 2, ..., 2n - 1\}\) and \(aR^{M}b\) iff \(\lfloor \frac{a}{2} \rfloor = \lfloor \frac{b}{2} \rfloor\). It is easy to check that \(M\) is a finite model of \(A\) which has size \(2n\). Therefore \(\text{Even} \subseteq \text{spec}(A)\).

3. The language consists of unary function \(s\), a unary relation \(R\) and a binary relation \(<\). We start with trying to understand what \(A\), \(B\), \(C\) and \(D = \forall x \exists y (x < y \land R y)\) mean. \(C\) simply states that \(<\) is a transitive relation.

If \(s\) was successor, \(<\) an order, then \(A\) would mean successor of \(x\) is larger than \(x\), and \(B\) would mean either \(R\) holds for \(x\) or for its successor, like being an even number. Then \(D\) would mean that for any number, there is an even number larger than that. This interpretation satisfy the sequent, i.e., in this structure (where we interpret the universe to be natural numbers, \(s\) as successor, \(<\) as order, and \(R\) as being even) which satisfies \(A\), \(B\), and \(C\), \(D\) is also satisfied. This interpretation is helpful in understanding why this sequent is true (and therefore provable) and will guide us in proving it, but giving a structure in which the sequent is true is not enough and we need to prove it to show that it is true in all structures. Let’s understand why it is true in this structure. Let \(x\) be an arbitrary number. Then \(x < sx\) by \(A\), and \(sx < ssx\) again by \(A\) and therefore by \(C\) we know that \(x < ssx\).

Now by \(B\), we have either \(Rsx\) or \(Rssx\), in both cases we have found a \(y\) such that \(x < y\) and \(Ry\). (Try to understand the meaning of the sequent in more general setting and see why it is true.) Now that we understand why the sentence is true, it is time to prove the sequent. Note that we have used \(A\) and \(x < sx\) twice in our argument.
We want to have a structure such that $A$ holds where $A$ is a finite sequence $a_1, a_2, ..., a_n$ which is not possible. Therefore $B \neq A_{n+1}$.

5. We want to have a structure such that $A = \forall x \forall y \forall z((x < y \land y < z) \supset x < z)$ is not true, i.e., its negation $\exists x \exists y \exists z(x < y \land y < z \land \neg x < z)$ is true. No one element structure will satisfy this sentence since in such a structure $a < b$ holds either for all elements or for none. So we need at least two elements in the universe. Let $M$ has universe with two distinct elements $a$ and $b$, and let $\prec_M$ be inequality relation on $M$, i.e., $a \prec_M b$ iff $a \neq b$. Then $a \neq b \land b \neq a \land \neg a \neq a$ is true in $M$, and therefore $M$ is the required structure. ($x$ and $z$ are set to $a$, and $y$ is set to $b$.)

6. (a) Let $M$ be a finite structure explicitly given to us, (i.e, we can list its elements, compute $f^M(\overline{a})$ where $f$ is a function in the language, and check if $R^M(\overline{a})$ holds where $R$ is a relation symbol in the language and $\overline{a}$ is a finite sequence in $M$, and $A$ a sentence with parameters from $M$.

The following recursive algorithm (based on the definition of a term) $[t]_M$ computes the value of closed term $t$ in the language with parameters from $M$:

- $t = \text{syn } a$ for some $a \in M$, return $a$,
- $t = \text{syn } f(t_1, ..., t_n)$, return $f^M([t_1]_M, ..., [t_n]_M)$.
Since \( t \) is a closed term, there is no variable in \( t \).

The following recursive algorithm \([A]_M\) checks if \( M \) satisfies \( A \):

\[
A = \text{syn } R(t_1, \ldots, t_n), \text{ return true if } R^{M}(\![t_1]_M, \ldots, [t_n]_M) \text{ is true, otherwise return false,}
\]

\[
A = \text{syn } \neg B, \text{ return } (\text{not } [B]_M),
\]

\[
A = \text{syn } B_1 \land B_2, \text{ return } ([B_1]_M \text{ and } [B_2]_M),
\]

\[
A = \text{syn } B_1 \lor B_2, \text{ return } ([B_1]_M \text{ or } [B_2]_M),
\]

\[
A = \text{syn } \forall x(B(x)), \text{ return true iff for all } a \in M, [B(a)]_M \text{ is true,}
\]

\[
A = \text{syn } \exists x(B(x)), \text{ return true iff for some } a \in M, [B(a)]_M \text{ is true.}
\]

"for all \( a \in M' \)" and "for some \( a \in M' \)" can be implemented by for loops ranging over elements of \( M \) using the fact that we can list members of \( M \).

To check if a formula \( \alpha \) has a model with exactly \( k \) elements, we list all possible structures \( M \) with \( k \) elements in the language of \( \alpha \) and use the above algorithm to check if \( \forall \alpha \) (the universal closure of \( \alpha \), the sentence obtained by putting universal quantifiers in front of \( \alpha \) for all variables appearing free in it) is true in \( M \), and return true if one of these structures satisfy \( \forall \alpha \). There are only finitely many such structures. If the language has functions \( f_1, \ldots, f_n \) where \( f_i \) has arity \( n_i \) and relations \( R_1, \ldots, R_m \) where \( R_i \) has arity \( m_i \), then we can fix the universe to be \( \{1, \ldots, k\} \) and there are \( k^{kn} \) possible interpretations for \( f_i^M \) and \( 2^{km} \) possible interpretations for \( R_i^M \). Therefore we can list all of them and check if the formula is satisfied in one of them in finite time.

(b) Let \( \alpha \) be a monadic formula with \( n \) predicate symbols. Let \( M \) be a model of \( \alpha \). We define a new structure \( M' \): Each predicate symbol partitions \( M \) to two sets, those satisfying it and those not satisfying it. Define the equivalence relation \( \sim \) over \( M \) by letting \( a \sim b \) iff for all predicate symbols \( P, P^M(a) \text{ iff } P^M(b) \).

In other words, \( a \sim b \) if they are in the same part of partition for all predicate symbols. Let universe of \( M' \) be the equivalence classes of \( \sim \). If \( P \) is a predicate symbol of the language, let \( P^M'([a]) \text{ iff } P^M(a) \), where \( [a] \) is the equivalence class of \( a \). It is easy to check that this structure is well defined, i.e., the definition of interpretations of predicates in \( M' \) is independent of the representative chosen from the equivalence class \( [a] \). \( M' \) has at most \( 2^n \) elements, since there are \( n \) predicates and there are two possibilities for each of them.

Next, we will show that \( \alpha \) is true in \( M \) iff it is true in \( M' \). Actually we will show something stronger, an arbitrary sentence \( \alpha(\overrightarrow{d}) \) with parameters \( \overrightarrow{d} \) from \( M \) is true in \( M \) iff it is true in \( M' \) when those parameters are replaced with their equivalence classes, i.e., \( M \models \alpha(\overrightarrow{d}) \text{ iff } M' \models \alpha([\overrightarrow{a}]) \). We will use induction on the structure of the sentence to prove this fact.

If \( \alpha(\overrightarrow{d}) \) is atomic, then \( \alpha = \text{syn } P(a) \) for some predicate symbol \( P \) and some element \( a \in M \). \( M' \models P([a]) \) iff \( P^{M'}([a]) \) iff (by definition of interpretations of predicates in \( M' \)) \( P^M(a) \) iff \( M \models P(a) \).

If \( \alpha(\overrightarrow{d}) \) = \( \text{syn } \neg \beta(\overrightarrow{d}) \), then \( M' \models \alpha([\overrightarrow{a}]) \text{ iff } M' \not\models \beta([\overrightarrow{a}]) \) iff (by induction hypothesis) \( M \not\models \beta(\overrightarrow{a}) \) iff \( M \models \alpha(\overrightarrow{a}) \).
If $\alpha(\overrightarrow{a}) = \text{sym} \beta_1 \land \beta_2$, then $M' \models \alpha([\overrightarrow{a}])$ iff $M' \models \beta_1([\overrightarrow{a}])$ and $M' \models \beta_2([\overrightarrow{a}])$ iff (by induction hypothesis) $M \models \beta_1(\overrightarrow{a})$ and $M \models \beta_2(\overrightarrow{a})$ iff $M \models \alpha(\overrightarrow{a})$. The case for $\lor$ is similar.

If $\alpha(\overrightarrow{a}) = \text{sym} \forall x \beta(x, \overrightarrow{a})$, then $M' \models \forall x \beta(x, [\overrightarrow{a}])$ iff for all $[b] \in M'$, $M' \models \beta([b], [\overrightarrow{a}])$ iff (by induction hypothesis) for all $b \in M$ $M \models \beta(b, \overrightarrow{a})$ iff $M \models \forall x \beta(x, \overrightarrow{a})$. The case for $\exists x$ is similar. This completes the proof of the lemma.

Let’s assume that $\alpha$ is satisfiable, then there is a structure $M$ s.t. $M \models \alpha$. By the lemma, $M' \models \alpha$, but $M'$ has at most $2^n$ elements. Therefore, if $\alpha$ is satisfiable, then it has a model of size at most $2^n$.

(c) Let’s assume the time required to check $P^M(a)$ for a size $m$ model is $t(m)$ (which we can assume can be done in linear time), and try to get an upper bound on the running time of $[A]_M$. The worst cases in the algorithm is the cases for quantifiers which decrease the size of formula by a constant and make $m$ recursive calls. Therefore the worst case running time is $m^{O(l)} t(m) = m^{O(l)}$. We have to check all possible structures with at most $2^n$ elements. The only thing which matters is which equivalence classes are empty, and since there are $2^n$ possible equivalence classes, where each of them can be empty, we have to check $2^{2^n} - 1$ structures ($-1$ for the empty structure which is not allowed). Therefore we obtain an upper bound of $2^{2^n} (2^n)^{O(l)} = 2^{2^n} 2^{nO(l)} = 2^{2^n + nO(l)}$. 