CS 228M: Logic for Computer Science
Quiz II Solutions

Ashwin Abraham

22nd October 2023

Marking Scheme:

- **2 marks** for correctly writing the expression for $x < y$ in Presburger Arithmetic
- **2 marks** for correctly writing the expression for $S(x, y)$ in Presburger Arithmetic
- **2 marks** for correctly writing the expression for $zero(x)$ in Presburger Arithmetic
- **2 marks** for correctly writing the expression for $one(x)$ in Presburger Arithmetic
- **2 marks** for correctly encoding the PA formula $x = y$ into MSO with binary encoding
- **5 marks** for correctly encoding the set of carry bits of binary addition in MSO
- **3 marks** for correctly encoding the bits of the sum in binary addition with MSO
- **2 marks** for restricting quantification only to finite sets, by domain transformation
- **Partial Marking** has been given according to the following rule: For every (non-major) change that has to be made to the expression to make it correct, **1 mark** has been deducted from the total in that part, unless 0 has already been reached
- If an alternative encoding has been used, marks will be given if and only if it is correct and the proof of the decidability of PA-SAT using that encoding has been mentioned in the solution

Question 1

(a) $x < y$ can be written in Presburger Arithmetic as $\exists z\,[x + z = y] \wedge \neg(x = y)$.

$\exists z\,[x + z = y]$ encodes the expression $x \leq y$, as z could be 0, so to remove the case where $x = y$, we add the second clause.

$S(x, y)$ can be written in Presburger Arithmetic as $x < y \wedge \neg\exists z(x < z \wedge z < y)$, where $x < y$ is expressed as mentioned above.

(b) $zero(x)$ can be written as $x + x = x$ or $+(x, x, x)$

$one(x)$ can be written as $\exists z\,[zero(z) \wedge S(z, x)]$, where $zero$ and S are expressed as mentioned above.

There maybe other correct expressions for these predicates, and marks have been awarded to all correct ones.

Some common errors in these two parts include encoding these in MSO instead of PA, forgetting to exclude the possibility that $x = y$ while encoding $x < y$, using 0 and 1 as constants in expressions (0 and 1 are not constants in PA), treating $zero$ and one as functions instead of predicates, and quantifying over x and y (which are free variables).

(c) We follow the encoding mentioned in the hint. Each natural number has a binary encoding. For example, the binary encoding of $17 = 1\times2^0+0\times2^1+0\times2^2+0\times2^3+1\times2^4$ would be 10001. In our encoding, we map each natural number to the set of positions at which a 1 is present, ie if $n = \sum_{i=0}^{\infty} a_i 2^i$, then n is mapped to $\{i : a_i = 1\}$. In terms of n, this is the set $\{i : \lfloor \frac{n}{2^i} \rfloor \equiv 1 \mod 2\}$. For our example of 17, the corresponding set is $\{0, 4\}$. This mapping defines a bijection from $\mathbb{N}$ to the set of **finite** subsets of $\mathbb{N}$. Using this encoding, each PA variable is converted to a **second order** MSO variable. For each PA variable x, we use it's capitalized form, X, to denote the corresponding variable. Note that our MSO is over universe $\mathbb{N}$, and contains the constant 0 and the successor relation S.

We first need to encode the atomic formulae of PA into MSO. The only atomic formulae of PA are $x = y$ and $+(x, y, z)$, ie $x + y = z$.

$x = y$ occurs if and only if the corresponding MSO variables X and Y will also be equal. However, since there is no equality between second order variables in MSO, we instead write $X = Y$ as $\forall u\,[X(u) \iff Y(u)]$.

The encoding of $x+y = z$ is a little more complex. We basically have to encode binary addition. In binary addition, there are a set of *carry bits* that are used to carry out the addition. Recall the truth table for a full-adder:

c_i	0	0	0	0	1	1	1	1
x_i	0	0	1	1	0	0	1	1
y_i	0	1	0	1	0	1	0	1
z_i	0	1	1	0	1	0	0	1
c_{i+1}	0	0	0	1	0	1	1	1

Here x_i, y_i and z_i are the i^{th} bits of x, y and z respectively, where $x + y = z$, and c_i is the i^{th} carry bit. Note that we start with $c_0 = 0$. From the table, we can see that $z_i \iff c_i \oplus x_i \oplus y_i$ and $c_{i+1} \iff (x_i \wedge y_i) \vee (x_i \wedge c_i) \wedge (y_i \vee c_i)$ (ie at least 2 of x_i, y_i and c_i are 1). We encode this into an MSO formula. Let C be the set of positions i at which the carry bit is enabled, ie $c_i = 1$. Our formula becomes:

$$\exists C[\neg C(0) \wedge \forall u, v(S(u, v) \implies [(C(u) \oplus X(u) \oplus Y(u) \iff Z(u)) \\ \wedge([(X(u) \wedge Y(u)) \vee (X(u) \wedge C(u)) \vee (Y(u) \wedge C(u))] \iff C(v))])]$$

Now that our atomic PA formulae have been encoded in MSO, we just have to encode the production rules that produce arbitrary PA formulae to MSO. Say the PA formulae φ and ψ have corresponding MSO formulae φ' and ψ'. We then have:

- The MSO formulae for $\neg\varphi$ and $\neg\psi$ are $\neg\varphi'$ and $\neg\psi'$ respectively
- The MSO formula for $\varphi \circ \psi$ is $\varphi' \circ \psi'$, for any binary connective $\circ$, such as $\wedge$, $\vee$, $\implies$, etc

The only connectives remaining are $\forall$ and $\exists$. Remember that our encoding maps natural numbers only to **finite** subsets of $\mathbb{N}$, not all subsets of $\mathbb{N}$. Therefore, when we translate a quantification over PA variables $\mathbb{N}$ to a quantification over an MSO variable, we must quantify only over finite sets.

Consider the proposition $finite(X)$ that is true if and only if the set corresponding to the second order variable X is finite. $finite(X)$ can be written as $\exists u \forall v \left[X(v) \implies v < u\right]$ (this is as a subset of $\mathbb{N}$ is finite if and only if it has an upper bound).

Now, if $\varphi(x)$ is PA formula with free variable x and MSO equivalent $\varphi'(X)$, then the MSO equivalent of $\forall x\ \varphi(x)$ is $\forall X \left[finite(X) \implies \varphi'(X)\right]$ and the MSO equivalent of $\exists x\ \varphi(x)$ is $\exists X \left[finite(X) \wedge \varphi'(X)\right]$. This is as we must only quantify over finite subsets of $\mathbb{N}$, which can be done via a domain transformation.[1]

By structural induction, it is easy to show that for any assignment $\alpha : Vars \to \mathbb{N}$, $\alpha \models \varphi \iff \alpha' \models \varphi'$, where φ is any PA formula and φ' is its MSO equivalent, and $\alpha' : Vars \to \{S \subseteq \mathbb{N} : |S| < \infty\}$ (the set of finite subsets of $\mathbb{N}$) such that $\alpha'(x) = \{i : \lfloor \frac{\alpha(x)}{2^i} \rfloor \equiv 1 \mod 2\}$ (ie $\alpha'(x)$ is the encoded version of α').

Now, the satisfiability problem asks that given a PA formula φ, if there exists an assignment α such that $\alpha \models \varphi$. This is equivalent to finding the value of the sentence[2] $\exists x_1 \ldots \exists x_n \varphi$, where $x_1 \ldots x_n$ are the free variables in φ. This is equivalent to finding the value of the corresponding sentence in MSO.

Now, note that this MSO sentence is over a structure with universe $\mathbb{N}$. For an MSO sentence φ over a word structure w, we can decide if $w \models \varphi$ by automata construction. In that case, the universe was finite, and there were relations Q_a determining the alphabet present at a position in a word. In this case, the universe is $\mathbb{N}$, which is infinite. However, we can employ the same automata construction described earlier by changing the acceptance condition to the **Büchi acceptance condition** - ie our automaton is now a Büchi automaton[3]. There is no alphabet associated to the word here, and so the transitions are unlabelled - so it is now more like a directed graph than an automaton. To check if $(\mathbb{N}, 0, S) \models \varphi$, we will just have to check if there exists an infinite walk of this graph. This can be checked by constructing the SCC (strongly connected component) graph of our graph and then checking if there exists a path from a component containing an initial state to a component containing a good state and a self-loop. This can be done by algorithms such as Kosaraju's Algorithm combined with a graph search algorithm such as DFS. Therefore, there exists an algorithm to decide whether $(\mathbb{N}, 0, S) \models \varphi$, for any MSO sentence φ. And since every PA sentence can be converted into an MSO sentence, the same algorithm can be used to decide if $(\mathbb{N}, +) \models \varphi$ for any PA sentence φ. Therefore, the satisfiability problem for Presburger Arithmetic is decidable.

[1] If we ignore the domain transformation, a sentence such as $\exists x, y, z \left[one(y) \wedge x + y = z \wedge zero(z)\right]$ which is false in PA would be true in MSO, and the reverse is also possible.

[2] The value of a sentence φ under a structure $\mathcal{S}$ is 1 if $\mathcal{S} \models \varphi$ and 0 otherwise.

[3] One may object by saying that since our quantification is only over finite sets, $(\mathbb{N}, 0, S) \models \varphi$ iff there exists some finite word w such that $w \models \varphi$. This is however, false. Consider the MSO sentence $\forall x \exists y S(x, y)$. Note that this contains no second order variables, so how they are quantified is irrelevant. This sentence is not satisfied by any finite word but is satisfied by $(\mathbb{N}, 0, S)$ (even assuming quantification is only over finite sets).