Presburger Arithmetic With Unary Predicates is Π_1^1 Complete*

Joseph Y. Halpern
IBM Almaden Research Center
San Jose, CA 95120
halpern@almaden.ibm.com

Abstract: We give a simple proof characterizing the complexity of Presburger arithmetic augmented with additional predicates. We show that Presburger arithmetic with additional predicates is Π_1^1 complete. Adding one unary predicate is enough to get Π_1^1 hardness, while adding more predicates (of any arity) does not make the complexity any worse.

*This paper is essentially identical to one that appears in *Journal of Symbolic Logic* **56**, 1991, pp. 637–642.

1 Introduction

Presburger arithmetic, the theory of the natural numbers with addition, was shown to be decidable in 1929 by Presburger, using quantifier elimination (see [End72] for a proof). Fischer and Rabin showed that it was actually decidable in double-exponential time [FR74]; a more precise characterization of its complexity was given by Berman [Ber80]. Adding unary predicates to the language makes it significantly more expressive. For example, with a unary predicate P, the following formula expresses the fact that $P(n)$ holds if and only if n is a perfect square. It uses the fact that the difference between consecutive squares keeps increasing by 2, so that $(k+2)^2 - (k+1)^2 = (k+1)^2 - k^2 + 2$.

$$
\begin{array}{l}
P(\mathbf{0}) \wedge P(\mathbf{1}) \wedge \forall n \exists m [m > n \wedge P(m)] \wedge \\
\quad \forall n_1, n_2, n_3 [P(n_1) \wedge P(n_2) \wedge P(n_3) \wedge n_1 < n_2 < n_3 \wedge \\
\qquad \forall m [n_1 < m < n_2 \vee n_2 < m < n_3 \Rightarrow \neg P(m)] \Rightarrow n_3 - n_2 = n_2 - n_1 + 2].
\end{array}
$$

Once we can express perfect squares, it is not hard to show that we can also express multiplication (since $2mn = (m+n)^2 - m^2 - n^2$). That is, if we had a ternary predicate Q in the language, we could force $Q(m_1, m_2, m_3)$ to hold iff $m_1 = m_2 \times m_3$. Thus, with a ternary predicate, we can easily get undecidability.

Given this observation, it is perhaps not surprising that the additional expressive power we can gain with unary predicates also comes at a cost. Presburger arithmetic with unary predicates was shown to be undecidable in [GS74]. In this paper, we completely characterize the complexity of Presburger arithmetic augmented by additional functions and predicates. We show that if we add even one unary predicate, then the validity problem for the resulting language is Π_1^1 complete (i.e., the set of formulas in the resulting language that are valid when interpreted over the natural numbers is a Π_1^1 complete set). However, adding more function and predicate symbols does not make things any worse; the validity problem remains in Π_1^1 no matter how many function and predicate symbols we add to the language.

In the next section we provide all the necessary definitions to make this paper self contained. We give the upper and lower bound proofs in Section 3.

We remark that Alur and Henzinger recently obtained an independent proof that Presburger arithmetic augmented by unary predicates is Π_1^1-complete, although they did not show that it sufficed for the lower bound to have only one unary predicate [AH89].[1] On the other hand, their proof shows that we do not even need the full power of addition to obtain Π_1^1-hardness; it suffices to have multiplication by two. We also remark that the result of this paper is used in [AH94] to prove undecidability of a first-order logic for reasoning about probability.

[1]Originally, they only had an undecidability result. Once they learned about the result of Harel, Pnueli, and Stavi described in Theorem 3.2 below through conversations with Moshe Vardi and the author, they were able to use their techniques to prove Π_1^1-completeness.

2 Definitions

Let $\mathcal{L}$, the language of Presburger arithmetic, be the first-order language with equality, with non-logical symbols $\{\mathbf{0}, \mathbf{1}, +\}$. If Φ is a collection of (uninterpreted) function and predicate symbols, then $\mathcal{L}(\Phi)$ is the result of augmenting $\mathcal{L}$ with the function and predicate symbols in Φ. Presburger arithmetic is intended to be interpreted over the natural numbers, where $\mathbf{0}$ and $\mathbf{1}$ are to be interpreted as 0 and 1, respectively, and $+$ is interpreted as addition. Notice that $x \leq y$ is definable as $\exists z(y = x + z)$; henceforth we proceed as if $\leq$ is in the language. We also take $x - y = z$ to be an abbreviation for $x = y + z$. Finally, we take $\mathbf{k}$ to be an abbreviation for $\mathbf{1} + \cdots + \mathbf{1}$ (k times); other similar abbreviations are also used in the paper.

As mentioned in the introduction, the validity problem for Presburger arithmetic, that is, the validity problem for the language $\mathcal{L}(\emptyset)$, is decidable, while the validity problem for $\mathcal{L}(\Phi)$, for Φ containing at least one unary predicate, is known to be undecidable. (We remark that if Φ contains only constant symbols, then it is easy to show that the validity problem for $\mathcal{L}(\Phi)$ remains decidable; adding fresh constant symbols to the language does not make the complexity any worse.)

We now briefly review the definition of Π^1_1; the interested reader should see [Rog67] for more details. Formulas of *second-order arithmetic with set variables* consist of formulas of first-order arithmetic (that is, in the language with constant symbols $\mathbf{0}$ and $\mathbf{1}$, together with the function symbols $+$ and $\times$) augmented with expressions of the form $x \in X$, where x is a number variable and X is a set variable, together with quantification over set variables and number variables. A *sentence* is a formula with no free variables. Second-order arithmetic with set variables is a very powerful language. For example, the following (true) sentence of the language expresses the law of mathematical induction over the natural numbers:

$$\forall X(\mathbf{0} \in X \wedge \forall x((x \in X \supset x + \mathbf{1} \in X) \supset \forall x(x \in X)))$$

A Π^1_1 formula (resp. Σ^1_1 formula) of second-order arithmetic with set variables is one of the form $\forall X_1 \ldots \forall X_n \varphi$ (resp. $\exists X_1 \ldots \exists X_n \varphi$), where φ is a formula of second-order arithmetic with set variables that has no quantification over set variables. A set A of natural numbers is in Π^1_1 (resp. Σ^1_1) if there is a Π^1_1 formula (resp. Σ^1_1 sentence) $\psi(x)$ with one free number variable x and no free set variables such that $a \in A$ iff $\psi(a)$ holds. Π^1_1 hardness and completeness are defined in the obvious way (the reduction is via one-one recursive functions). It is well-known that Π^1_1-hard sets are not recursively enumerable (see [Rog67]).

3 Upper and lower bound proofs

In this section we prove the Π^1_1 completeness result.

Theorem 3.1:

(a) *The validity problem for $\mathcal{L}(\Phi)$ is in Π^1_1 for all choices of Φ.*

(b) *If Φ contains at least one unary predicate, then the validity problem for $\mathcal{L}(\Phi)$ is Π^1_1 hard.*

Proof: The upper bound proof is almost immediate from the definition of Π^1_1. Suppose for ease of exposition that Φ consists only of predicates. (This is without loss of generality, since we can always replace a k-ary function by a $(k+1)$-ary predicate.) Given a formula φ in $\mathcal{L}(\Phi)$ with predicate symbols $P_1, \ldots, P_m$, we can translate it to a formula of arithmetic with set variables $X_1, \ldots, X_m$ by simply replacing subformulas of φ of the form $P_i(x_1, \ldots, x_k)$ by $\langle x_1, \ldots, x_k \rangle \in X_i$, where $\langle x_1, \ldots, x_k \rangle$ is a recursive encoding of a sequence as a natural number (see [Rog67] for examples of such encodings); we assume that the encoding is such that a sequence of length k can be distinguished from one of length k' for $k' \neq k$. Let φ' be the result of this translation. Clearly φ is valid iff $\forall X_1 \ldots X_m \varphi'$ is true. This shows that the validity problem for $\mathcal{L}(\Phi)$ is in Π^1_1.

For the lower bound, we prove that the satisfiability problem for $\mathcal{L}(\Phi)$ is Σ^1_1-hard. We need the following result, due to Harel, Pnueli, and Stavi [HPS83]. We say that a nondeterministic Turing machine $\mathbf{A}$ is *recurrent* if, when started on the empty tape, $\mathbf{A}$ has an infinite computation that reenters its start state infinitely often. Let $\mathbf{A}_0, \mathbf{A}_1, \mathbf{A}_2, \ldots$ be a recursive enumeration of the nondeterministic Turing machines with one tape, infinite to the right.

Theorem 3.2: *([HPS83]) The set $\{n | \mathbf{A}_n$ is recurrent$\}$ is Σ^1_1 complete.*

Given a Turing machine $\mathbf{A}$, we now show how to effectively construct a formula $\varphi_{\mathbf{A}}$ in $\mathcal{L}(\{P\})$, where P is a unary predicate, such that $\varphi_{\mathbf{A}}$ is satisfiable iff $\mathbf{A}$ is recurrent. Once we do this, it will follow from Theorem 3.2 that the satisfiability problem for Presburger arithmetic augmented with one unary predicate is Σ^1_1 hard.

Suppose $\mathbf{A}$ uses tape alphabet Γ and has state space Q. We use the special symbol b to denote the blank symbol and \$ to separate between consecutive IDs (*i*nstantaneous *d*escriptions of the Turing machine), where $b, \$ \notin (\Gamma \cup Q)$. Let CD (for *cell descriptor*) be $\Gamma \cup (\Gamma \times Q) \cup \{b, \$\}$. (As usual, we use a pair $(\gamma, q) \in \Gamma \times Q$ to denote that $\mathbf{A}$ is in state q with its head reading symbol γ.) We first assume we have many unary predicates, one predicate P_c corresponding to each $c \in CD$, and then show how to reduce to one. A computation looks like a sequence of IDs separated by \$'s: $\$\mathrm{ID}_1\$\mathrm{ID}_2\$\mathrm{ID}_3\$\ldots$, where ID_i is in turn a finite sequence of cell descriptors $c_1 c_2 \ldots c_{k_i}$. We can encode this computation by forcing $P_c(n)$ to be true iff the symbol on the n^{th} cell of the computation is c. We do this using the following formulas $\varphi_1, \ldots, \varphi_5$:

- The formula φ_1 guarantees for all n, $P_c(n)$ holds for exactly one $c \in CD$:

$$\varphi_1 =_{\text{def}} \forall n (\bigvee_{c,d \in CD} (P_c(n) \wedge \bigwedge_{d \neq c} \neg P_d(n)))$$

- The formula φ_2 guarantees that the distance between consecutive \$'s always increases by one:

$$\begin{aligned}\varphi_2 =_{\mathrm{def}} \;& \forall n_1, n_2((n_1 < n_2 \wedge P_\$(n_1) \wedge P_\$(n_2) \wedge \forall m((n_1 < m < n_2) \Rightarrow \neg P_\$(m))) \Rightarrow \\ & \exists m'((m' - n_2 = n_2 - n_1 + \mathbf{1}) \wedge P_\$(m') \wedge \forall m''((n_2 < m'' < m') \Rightarrow \neg P_\$(m''))))\end{aligned}$$

- The formula φ_3 guarantees that the machine "starts right", on a blank tape in the start state q_0:

$$\varphi_3 =_{\mathrm{def}} P_\$(0) \wedge P_{(b,q_0)}(1) \wedge P_\$(2)$$

- We next want to say that successive steps of the computation proceed according to the transition rules of the Turing machine. It is well known that we can characterize a Turing machine by giving a function which, given three consecutive cells in an ID, describes the set of possible corresponding three cells in the next ID. Thus, given the Turing machine $\mathbf{A}$ and $i, j, k \in CD$, there is a function N such that $N(i,j,k) = \{\langle c,d,e\rangle |$ if $\langle i,j,k\rangle$ describes three consecutive cells in a given ID then $\langle c,d,e\rangle$ is a possible description of the corresponding cells in the next ID$\}$. Note that $N(i,j,k)$ is a finite set for each triple (i,j,k). We can talk about corresponding cells in two consecutive IDs by using the distance between consecutive \$'s as a yardstick.

$$\begin{aligned}\varphi_4 =_{\mathrm{def}} \;& \forall n, \ell[\textstyle\bigwedge_{i,j,k \in CD, j \neq \$}(P_i(n) \wedge P_j(n{+}\mathbf{1}) \wedge P_k(n{+}\mathbf{2}) \wedge \\ & \quad \exists m_1, m_2[(m_1 \leq n < m_2) \wedge (m_2 = m_1 + \ell) \wedge P_\$(m_1) \wedge P_\$(m_2) \wedge \\ & \quad \forall m'(m_1 < m' < m_2 \Rightarrow \neg P_\$(m'))]) \Rightarrow \\ & \quad \textstyle\bigvee_{\langle c,d,e\rangle \in N(i,j,k)}(P_c(n+\ell) \wedge P_d(n{+}\ell{+}\mathbf{1}) \wedge P_e(n{+}\ell{+}\mathbf{2}))]\end{aligned}$$

- Finally, we need to say that $\mathbf{A}$ returns to the start state infinitely often. This is the job of φ_5:

$$\varphi_5 =_{\mathrm{def}} \forall n \exists m(m > n \wedge (\bigvee_{\gamma \in \Gamma} P_{(\gamma,q_0)}(m))).$$

Let $\varphi_{\mathbf{A}} = \varphi_1 \wedge \ldots \wedge \varphi_5$. We leave it to the reader to check that $\varphi_{\mathbf{A}}$ is satisfiable iff there is a recurrent computation of $\mathbf{A}$. From Theorem 3.2, it follows that the satisfiability problem for $\mathcal{L}(\Phi)$ where Φ contains an infinite collection of unary predicates is Σ_1^1 hard, and hence the validity problem is Π_1^1 hard.

We conclude by briefly sketching how we can encode the computation of $\mathbf{A}$ given only one unary predicate, rather than a collection of them. Suppose the proof above actually uses k predicates to encode the computation of $\mathbf{A}$. For simplicity, call these predicates $P_1, \ldots, P_k$. We can think of the infinite sequence of cell descriptors that describes a computation of $\mathbf{A}$ as an infinite word w written in a language with k symbols: $P_1, \ldots, P_k$. Corresponding to w we consider an infinite word w' in a language with only two symbols, 0 and 1; both w and w' are intended to encode the same computation. Conceptually, we think of w' as being partitioned into blocks of size $k+4$. The last 4 symbols of each such block always contain the string 0110; this string marks the end of the block. The first k symbols of the block contain exactly one 1, all the rest being 0s. Intuitively, symbol i in the m^{th} block of w' is 1 iff the m^{th} symbol in w is P_i.

If we use the truth and falsity of $P(n)$ to denote that the n^{th} symbol is 1 or 0, respectively, then the following formula says that w' is divided up into blocks of size $k+4$ in the appropriate way:

$$\begin{array}{l}\neg P(\mathbf{k}) \wedge P(\mathbf{k}+\mathbf{1}) \wedge P(\mathbf{k}+\mathbf{2}) \wedge \neg P(\mathbf{k}+\mathbf{3}) \wedge \\ \quad \forall n[(P(n) \wedge P(n{+}\mathbf{1})) \Leftrightarrow (P(n{+}\mathbf{k}+\mathbf{4}) \wedge P(n{+}\mathbf{k}+\mathbf{5}))] \wedge \\ \quad\quad \forall n[P(n+\mathbf{k}+\mathbf{1}) \wedge P(n+\mathbf{k}+\mathbf{2}) \Rightarrow \\ \quad\quad\quad \exists m[(n \leq m < n+\mathbf{k}) \wedge P(m) \wedge \forall m'[((n \leq m' < n+\mathbf{k}) \wedge m' \neq m) \Rightarrow \neg P(m')]]]\end{array}$$

It is now straightforward to translate $\varphi_2, \ldots, \varphi_5$ to hold with respect to this encoding of the computation. (There is no need to translate φ_1; we have already forced it to be the case that precisely one predicate holds for each block.) We leave details to the reader. This completes our proof. ∎

We remark that a minor extension to this proof shows that arithmetic (with + and $\times$) augmented by unary predicates is also Π_1^1 complete. Clearly, our Π_1^1 lower bound applies. The upper bound follows by the same argument.

Acknowledgements: I'd like to thank Martín Abadi and Moshe Vardi for discussions on this proof. I'd also like to thank Herbert Enderton, Jeanne Ferrante, Bill Gasarch, Yuri Gurevich, Harry Lewis, Albert Meyer, Larry Stockmeyer, and Moshe Vardi for help in my efforts to track down previous undecidability results on Presburger arithmetic. I'd particularly like to thank Mark Pleszkoch, who dug up the [GS74] result.

References

[AH89] R. Alur and T. A. Henzinger. A really temporal logic. In *Proc. 30th IEEE Symp. on Foundations of Computer Science*, pages 164–169, 1989.

[AH94] M. Abadi and J.Y. Halpern. Decidability and expressiveness for first-order logics of probability. *Information and Computation*, 112(1):1–36, 1994.

[Ber80] L. Berman. The complexity of logical theories. *Theoretical Computer Science*, 11:71–77, 1980.

[End72] H. B. Enderton. *A Mathematical Introduction to Logic.* Academic Press, New York, 1972.

[FR74] M. J. Fischer and M. O. Rabin. Super-exponential complexity of Presburger arithmetic. In R. M. Karp, editor, *Complexity of Computation, SIAM-AMS Proceedings, Vol. 7*, pages 27–42, 1974.

[GS74] S. Garfunkel and J. H. Schmerl. The undecidability of theories of groupoids with an extra predicate. *Proc. AMS*, 42(1):286–289, 1974.

[HPS83] D. Harel, A. Pnueli, and J. Stavi. Propositional dynamic logic of nonregular programs. *Journal of Computer and System Sciences*, 26(2):222–243, 1983.

[Rog67] H. Rogers, Jr. *Theory of Recursive Functions and Effective Computability.* McGraw-Hill, New York, 1967.