Model checking recursive programs with numeric data types

Matthew Hague and Anthony Widjaja To

Oxford University Computing Laboratory

We give a direct encoding of the reachability problem of PCo into existential Presburger arithmetic used in our implementation. This note accompanies a conference submission by the authors [1]

For this presentation we use variables for the nonterminals of the generated CFGS, in our implementation we replace these variables with the sums given by the formula $Count\,(\boldsymbol{\mathcal{P}}_1, \ldots, \boldsymbol{\mathcal{P}}_h, \boldsymbol{\mathcal{N}}_1, \ldots, \boldsymbol{\mathcal{N}}_h, \boldsymbol{\mathcal{T}})$ given below.

Recall, we can solve the reachability problem in NP by building a pushdown system without counters, but with a "mode" vector. This vector contains a value for each counter of the system indicating whether the value is unchanged (i.e. zero), is in the increase phase, is in the decrease phase, or has (been guessed to have) re-reached zero. That is, a vector of values from 0 to 3 respectively. For this proof, we allow up to r reversals, and so our values range from 0 to $r_{max} = 2 + r + \lceil r/2 \rceil$. That is, the counter cycles through a zero phase (0), an increment phase (+), a decrement phase (−), an (optional) zero phase (0), an increment phase (+), &c. For example, for zero reversals each counter may go through phases 0 and +, for one reversal $0, +, -, 0$, two reversals $0, +, -, 0, +$, three $0, +, -, 0, +, -, 0$, &c.

The key observation in the proof is that, although there are an exponential number of vectors, only a linear number of them may be used during a particular run. This is because, starting from $(0, 0, \ldots, 0)$, the value of the components only increases. Hence, we can guess the sequence $\boldsymbol{m}_1, \ldots, \boldsymbol{m}_h$ of vectors used during a successful run, where h is the maximum number of vector changes. We will refer to each $\boldsymbol{m}_i$ as a *mode*. We do not assume that all runs cycle through h modes (they may go through fewer).

Once the mode sequence has been guessed, we can construct a polynomially sized pushdown system that outputs its counter operations. We then construct the existential Presburger formula representing the Parikh image of this system, and check whether a run exists where the number of increments equals the number of decrements for each counter.

This shows that the problem is in NP; however, we do not know how to perform the guessing phase efficiently. To solve this problem we show how to encode the guessing of the mode sequence into the existential Presburger formula. Efficient Presburger solvers can then be used.

1 Context-Free Grammars

We first build a family of context-free grammars representing the pushdown system. These grammars contain information that enables the Presburger formula to assert facts about its correctness. We assume that the transition relation is of the form $(Q \times \Gamma \times Const_X) \times \mathsf{ACT} \times (Q \times \Gamma^{\leq 2} \times \{-1, 0, 1\}^n)$. That is, we only increase the stack height by at most one at each step. This is a standard assumption and is not a real restriction.

We build a grammar $\mathcal{G}_i$ for each of the h modes. We illustrate the form of the grammars with an example. Let $((p, a_1, c), \alpha, (p', a_2a_3, \boldsymbol{v}))$ be a pushdown rule. Let $\boldsymbol{v}_i$ be the terminals in $\boldsymbol{v}$ augmented with an i subscript. This allows us to know which mode counter actions occur in. The equivalent rules in $\mathcal{G}_i$ will be of the form

$$N^{i,j}_{p,a_1,q} \rightarrow \alpha \boldsymbol{v}_i N^{i,k}_{p',a_2,q'} N^{k,j}_{q',a_3,q}$$

for each $q' \in Q$ and $i \leq k \leq j \leq h$. Informally, a non-terminal $N^{i,j}_{p,a_1,q}$ means that, from a configuration with control state p and top-of-stack character a_1, we can eventually pop the a_1 character and reach control state q. Furthermore, the run will start in mode i and end in mode j. Hence, the right-hand side of the rule, first makes the action and counter operations visible, then requires that the pushed a_2 character is popped to some control state q', using modes i to k, and the a_3 character is popped to q, using the remaining modes k to j.

We define the set $\mathcal{T} = \{\ \alpha, \boldsymbol{v} \mid \alpha \in \mathsf{ACT}, \boldsymbol{v} \in \{-1, 0, 1\}^n\ \}$ of terminals and the following families of non-terminals

$$\mathcal{N}_i = \left\{\ N^{i,j}_{p,a,q} \mid i \leq j \leq h, p \in Q, q \in Q, a \in \Gamma\ \right\}$$

and production rules

$$\begin{array}{rl}\mathcal{P}_i = & \left\{\ N^{i,j}_{p,a_1,q} \rightarrow \alpha \boldsymbol{v}_{i'} N^{i',k}_{p',a_2,q'} N^{k,j}_{q',a_3,q} \ \middle|\ \begin{array}{c} ((p, a_1, c), \alpha, (p', a_2a_3, \boldsymbol{v})) \in \delta, \\ i \leq i' \leq k \leq j \leq h, q' \in Q, \\ i' = i \text{ or } i' = i + 1 \end{array}\right\} \cup \\ & \left\{\ N^{i,j}_{p,a_1,q} \rightarrow \alpha \boldsymbol{v}_{i'} N^{i',j}_{p',a_2,q} \ \middle|\ \begin{array}{c} ((p, a_1, c), \alpha, (p', a_2, \boldsymbol{v})) \in \delta, \\ i, i' \leq j \leq h, i' = i \text{ or } i' = i + 1 \end{array}\right\} \cup \\ & \left\{\ N^{i,i'}_{p,a_1,q} \rightarrow \alpha \boldsymbol{v}_{i'} \mid ((p, a_1, c), \alpha, (q, \epsilon, \boldsymbol{v})) \in \delta, i' = i \text{ or } i' = i + 1\ \right\}\ .\end{array}$$

The use of i' in the rules reflects the fact that a rule may change the mode. Finally, the grammar $\mathcal{G}_i$ represents the run of the system for a particular mode i and is defined as follows.

$$\mathcal{G}_i = \left(\mathcal{N}_i, \left(\mathcal{T} \cup \bigcup_{i < j \leq h} \mathcal{N}_j\right), \mathcal{P}_i\right)\ .$$

In particular, a grammar $\mathcal{G}_i$ has non-terminals $\mathcal{N}_i$ since all moves must belong to the current mode. The non-terminals include all $\mathcal{N}_j$ for $i < j \leq h$ since a run may induce commitments to complete the run in a later mode. The commitments are thus outputted by grammar $\mathcal{G}_i$, but treated as non-terminals in grammar $\mathcal{G}_j$, for some j.

2 The Existential Presburger Formula

We show how to construct an existential Presburger formula that has a solution iff the PCo has an accepting run. Let $\mathcal{T} = \{\ t_1, \ldots, t_n\ \}$, $\boldsymbol{\mathcal{T}} = \hat{t}_1, \ldots, \hat{t}_n$ and let $\phi\left(\boldsymbol{\mathcal{T}}\right)$ be the user supplied constraint on the terminals.

Let $\mathcal{N}_i = \{\ n^i_1, \ldots, n^i_{x_i}\ \}$, $\mathcal{P}_i = \{\ r^i_1, \ldots, r^i_{y_i}\ \}$ for some x_i and y_i and let $\boldsymbol{\mathcal{N}}_i = (\hat{n}^i_1, \ldots, \hat{n}^i_{x_i})$ and $\boldsymbol{\mathcal{P}}_i = (\hat{r}^i_1, \ldots, \hat{r}^i_{y_i})$. From Verma *et al.* [2], we can build a linear-sized formula $\mathit{Run}_i\left(\boldsymbol{\mathcal{N}}_i, \boldsymbol{\mathcal{P}}_i\right)$ which holds when there is a fully expanded derivation forest in the grammar $\mathcal{G}_i$ rooted by $\hat{n}^i_j$ copies of non-terminal n^i_j for each non-terminal. Moreover, the derivation forest uses each production rule r^i_j a total of $\hat{r}^i_j$ times.

We can also construct straightforwardly a formula

$$\mathit{Count}\left(\boldsymbol{\mathcal{P}}_1, \ldots, \boldsymbol{\mathcal{P}}_h, \boldsymbol{\mathcal{N}}_1, \ldots, \boldsymbol{\mathcal{N}}_h, \boldsymbol{\mathcal{T}}\right)$$

that holds when the number of non-terminals and terminals counted by $\boldsymbol{\mathcal{N}}_1, \ldots, \boldsymbol{\mathcal{N}}_h$ and $\boldsymbol{\mathcal{T}}$ matches the number outputted by firing the production rules $\boldsymbol{\mathcal{P}}_1, \ldots, \boldsymbol{\mathcal{P}}_h$ times. Note that a production rule in $\mathcal{P}_i$ does not output non-terminals in $\mathcal{N}_i$, only non-terminals in $\mathcal{N}_j$ for $j > i$.

In particular, the formula we require is

$$\begin{array}{c}\bigwedge\limits_{i>1,j} \left(\hat{n}^i_j = \sum_{k<i,j'} \hat{r}^k_{j'} \cdot \mathit{Occurences}\left(n^i_j, r^k_{j'}\right)\right) \wedge \\ \bigwedge\limits_{j} \left(\hat{t}_j = \sum_{i,j'} \hat{r}^j_i \cdot \mathit{Occurences}\left(t_j, r^i_j\right)\right)\end{array}$$

where $\mathit{Occurences}\left(x, r^i_j\right)$ denotes the number of occurrences of x in the right-hand side of the rule r^i_j. Note that we do not put any conditions on the values of $\boldsymbol{\mathcal{N}}_1$. This is because these are supplied in the main formula.

Let us assume that p_0 is the initial control state, a_0 the initial stack character and f the final control state. Then let $\mathit{Start}\left(\boldsymbol{\mathcal{N}}_1\right)$ be the formula asserting of $\boldsymbol{\mathcal{N}}_1$ that one and only one of $N^{1,j}_{(p_0,a_0,f)}$ for $1 \leq j \leq h$ is one, while all others are zero. The formula we require is of the form

$$\begin{aligned}\mathit{HasRun} = {} & \exists \boldsymbol{m}_1, \ldots, \boldsymbol{m}_h, \\ & \quad \exists \boldsymbol{\mathcal{P}}_1, \ldots, \boldsymbol{\mathcal{P}}_h, \\ & \quad\quad \exists \boldsymbol{\mathcal{N}}_1, \ldots, \boldsymbol{\mathcal{N}}_h, \boldsymbol{\mathcal{T}}. \\ & \quad\quad\quad \mathit{Start}\left(\boldsymbol{\mathcal{N}}_1\right) \wedge \mathit{Sequence}\left(\boldsymbol{m}_1, \ldots, \boldsymbol{m}_h\right) \wedge \\ & \quad\quad\quad\quad \mathit{Count}\left(\boldsymbol{\mathcal{P}}_1, \ldots, \boldsymbol{\mathcal{P}}_h, \boldsymbol{\mathcal{N}}_1, \ldots, \boldsymbol{\mathcal{N}}_h, \boldsymbol{\mathcal{T}}\right) \wedge \\ & \quad\quad\quad\quad\quad \bigwedge_i \left(\mathit{Run}_i\left(\boldsymbol{\mathcal{N}}_i, \boldsymbol{\mathcal{P}}_i\right) \wedge \mathit{Valid}_i\left(\boldsymbol{m}_i, \boldsymbol{\mathcal{P}}_i\right)\right) \wedge \\ & \quad\quad\quad\quad\quad\quad \bigwedge_{1 \leq i < h} \mathit{Connects}_i\left(\boldsymbol{m}_i, \boldsymbol{\mathcal{P}}_i, \boldsymbol{m}_{i+1}\right) \wedge \\ & \quad\quad\quad\quad\quad\quad\quad \bigwedge_{1 \leq i \leq h} \mathit{GoodCounts}_i\left(\boldsymbol{m}_i, \boldsymbol{\mathcal{T}}\right) \wedge \\ & \quad\quad\quad\quad\quad\quad\quad\quad \phi\left(\boldsymbol{\mathcal{T}}\right)\end{aligned}$$

where $\mathit{Sequence}\left(\boldsymbol{m}_1, \ldots, \boldsymbol{m}_h\right)$ asserts that the guessed sequence of modes is a valid sequence, $\mathit{Valid}_i\left(\boldsymbol{m}_i, \boldsymbol{\mathcal{P}}_i\right)$ ensures the run given by $\mathit{Run}_i\left(\boldsymbol{\mathcal{N}}_i, \boldsymbol{\mathcal{P}}_i\right)$ respects the mode $\boldsymbol{m}_i$, and $\mathit{Connects}_i\left(\boldsymbol{m}_i, \boldsymbol{\mathcal{P}}_i, \boldsymbol{m}_{i+1}\right)$ asserts that the fired rules adequately connect sequential modes. Finally, $\mathit{GoodCounts}_i\left(\boldsymbol{m}_i, \boldsymbol{\mathcal{T}}\right)$ asserts that the counter values are correct with respect to the mode. We define these formulas below.

The formula $\mathit{Sequence}\left(\boldsymbol{m}_1, \ldots, \boldsymbol{m}_h\right)$ requires an increasing sequence of vectors starting from $(0, 0, \ldots, 0)$. Let k be the number of counters, and m_i^k denote the kth component of $\boldsymbol{m}_i$. Recall r is the number of reversals. Also define a family of formulas for detecting whether a rule is congruent with a mode. Let

$$\begin{aligned} \mathit{Zero}\,(m) &= \bigvee_{0 \le j \le r} (m = 3j) \\ \mathit{Up}\,(m) &= \bigvee_{0 \le j < r} (m = 3j+1) \\ \mathit{Down}\,(m) &= \bigvee_{0 \le j < r} (m = 3j+2)\ . \end{aligned}$$

We have the following

$$\begin{aligned} \mathit{Sequence}\left(\boldsymbol{m}_1, \ldots, \boldsymbol{m}_h\right) = {} & \bigwedge_{1 \le j \le k} m_1^j = 0 \ \wedge \\ & \bigwedge_{1 \le i \le h} \bigwedge_{1 \le j \le k} m_i^j \le r_{max} \ \wedge \\ & \bigwedge_{1 \le i < h} \bigwedge_{1 \le j \le k} \left(m_{i+1}^j \ge m_i^j\right)\ . \end{aligned}$$

The formula $\mathit{Valid}_i\left(\boldsymbol{m}_i, \boldsymbol{\mathcal{P}}_i\right)$ is more involved. Let $\mathit{change}\,(r)$ be a constant which holds whenever the rule r is of the form $N_{p,a,q}^{i,j} \to \alpha \boldsymbol{v}_{i+1} N_{p',b,q'}^{i+1,k} N_{q',c,q}^{k,j}$, $N_{p,a,q}^{i,j} \to \alpha \boldsymbol{v}_{i+1} N_{p',b,q}^{i+1,j}$ or $N_{p,a,q}^{i,i+1} \to \alpha \boldsymbol{v}_{i+1}$. Furthermore, let $\mathit{inc}\,(r, z)$ be a constant holding when r increments counter z and similarly for $\mathit{dec}\,(r, z)$. We also say $\mathit{incdec}\,(r)$ holds if the rule increments or decrements at least one counter. Finally, let $C_{r_j^i}\left(\boldsymbol{m}\right)$ be the counter constraint associated with the rule r_j^i with all atoms replaced by the appropriate literal $\mathit{Zero}\,(m_z)$ or $\neg \mathit{Zero}\,(m_z)$.

$$\begin{aligned} & \mathit{Applicable}_{r_j^i}\left(m_1, \ldots, m_k\right) = \\ & C_{r_j^i}\left(m_1, \ldots, m_k\right) \\ & \wedge \begin{cases} \left(\begin{array}{l} \displaystyle\bigwedge_{\mathit{inc}\left(r_j^i, z\right)} \left(\mathit{Zero}\,(m_z) \vee \mathit{Down}\,(m_z)\right) \\ \displaystyle \wedge \bigwedge_{\mathit{dec}\left(r_j^i, z\right)} \left(\mathit{Up}\,(m_z) \vee \mathit{Down}\,(m_z)\right) \end{array}\right) & \text{if } \mathit{incdec}\left(r_j^i\right) \text{ and } \mathit{change}\left(r_j^i\right) \\ \left(\begin{array}{l} \displaystyle\bigwedge_{\mathit{inc}\left(r_j^i, z\right)} \mathit{Up}\,(m_z) \\ \displaystyle \wedge \bigwedge_{\mathit{dec}\left(r_j^i, z\right)} \mathit{Down}\,(m_z) \end{array}\right) & \text{if } \mathit{incdec}\left(r_j^i\right) \text{ and } \neg \mathit{change}\left(r_j^i\right) \\ \text{False} & \text{if } \neg \mathit{incdec}\left(r_j^i\right) \text{ and } \mathit{change}\left(r_j^i\right) \\ \text{True} & \text{if } \neg \mathit{incdec}\left(r_j^i\right) \text{ and } \neg \mathit{change}\left(r_j^i\right)\ . \end{cases} \end{aligned}$$

That is, in addition to the counter constraint being satisfied, the rule must be in a mode it can change (where the rule is applicable) if required, and leave it unchanged otherwise.

We are now ready to define $Valid_i\left(\boldsymbol{m}_i, \boldsymbol{\mathcal{P}}_i\right)$. Recall x_i is the number of production rules in $\mathcal{P}_i$.

$$Valid_i\left(\boldsymbol{m}_i, \boldsymbol{\mathcal{P}}_i\right) = \bigwedge_{1 \leq j \leq x_i} \hat{r}^i_j > 0 \Rightarrow Applicable_{r^i_j}\left(\boldsymbol{m}_i\right)$$

Note that it is a consequence of the construction of $\mathcal{G}_i$ that only one mode changing rule can be applied in any derivation.

Next, we define a family of formulas which hold when a rule can connect two modes, causing the change.

$$Connects_{r^i_j}\left(m_1, \ldots, m_k, m'_1, \ldots, m'_k\right) = \\ \begin{cases} \left(\begin{array}{l} \bigwedge\limits_{inc\left(r^i_j, z\right)} \left(\begin{array}{c} \left(Zero\left(m_z\right) \wedge Up\left(m'_z\right)\right) \ \vee \\ \left(Down\left(m_z\right) \wedge Up\left(m'_z\right)\right) \end{array} \right) \wedge \\ \bigwedge\limits_{dec\left(r^i_j\right)} \left(\begin{array}{c} \left(Up\left(m_z\right) \wedge Down\left(m'_z\right)\right) \vee \\ \left(Down\left(m_z\right) \wedge Zero\left(m'_z\right)\right) \vee \\ \left(Up\left(m_z\right) \wedge Zero\left(m'_z\right)\right) \end{array} \right) \end{array} \right) & \text{if } incdec\left(r^i_j\right) \text{ and } change\left(r^i_j\right) \\ \text{False} & \text{otherwise} \end{cases}$$

Note that the second disjunct of the second rule represents a non-deterministic guess that the counter has re-reached zero. Finally, we have

$$Connects_i\left(\boldsymbol{m}_i, \boldsymbol{\mathcal{P}}_i, \boldsymbol{m}_{i+1}\right) = \left(\bigwedge_{\substack{1 \leq j \leq x_i \\ change\left(r^i_j\right)}} \hat{r}^i_j > 0 \Rightarrow Connects_{r^i_j}\left(\boldsymbol{m}_i, \boldsymbol{m}_{i+1}\right) \right) \wedge \\ \left(\bigwedge_{\substack{1 \leq j \leq x_i \\ change\left(r^i_j\right)}} \hat{r}^i_j = 0 \Rightarrow \boldsymbol{m}_i = \boldsymbol{m}_{i+1} \right) .$$

Finally, $GoodCounts_i\left(\boldsymbol{m}_i, \boldsymbol{\mathcal{T}}\right)$ is defined as follows. Let $IncDec^z_{=,i}\left(\boldsymbol{\mathcal{T}}\right)$ assert, for the zth counter, the number of increments equals the number of decrements in all modes less than or equal to i. Similarly define $IncDec^z_{>,i}\left(\boldsymbol{\mathcal{T}}\right)$. Then we have

$$GoodCounts_i\left(\boldsymbol{m}_i, \boldsymbol{\mathcal{T}}\right) = \bigvee_z \left(\begin{array}{c} \left(Zero\left(m_z\right) \Rightarrow IncDec^z_{>,i}\left(\mathcal{T}\right)\right) \wedge \\ \left(Zero\left(m_z\right) \Rightarrow IncDec^z_{=,i}\left(\mathcal{T}\right)\right) \end{array} \right)$$

References

1. M. Hague and A. W. To. Model checking recursive programs with numeric data types. Submitted.
2. Kumar Neeraj Verma, Helmut Seidl, and Thomas Schwentick. On the complexity of equational horn clauses. In Robert Nieuwenhuis, editor, *CADE*, volume 3632 of *Lecture Notes in Computer Science*, pages 337–352. Springer, 2005.