Presentation: Linear Invariant Generation Using Non-Linear Constraint Solving

Ashwin Abraham

IIT Bombay

26th September, 2023

Table of Contents

Abstract

A method is presented for generating linear invariants for transition systems that reduces to a non-linear constraint solving problem.

Transition System

A transition system P is a tuple $\langle V, L, l_0, \Theta, \mathcal{T} \rangle$ where V is a set of variables, L is a set of locations, $l_0 \in L$ is an initial location, Θ is an initial assertion over the variables V, and $\mathcal{T}$ is a set of transitions of the form (l, l', ρ_τ), where ρ_τ is an assertion over $V \cup V'$ that serves as the transition condition. We use V' to distinguish the variables pre and post transition. It is not the case that each location has it's own variable set, but rather that for transitions, we would like to have constraints relating the values of the variables before and after the update. We assume that $V = \{x_1, \ldots x_n\}$ throughout the paper. For any formula ψ over variables V, we denote the formula obtained by replacing the variables in V with the corresponding variables in V' by ψ'.

Transition System

$$
\begin{array}{l}
\textbf{integer } i, j \textbf{ where } i = 2 \wedge j = 0 \\
l_0 : \textbf{while true do} \\
\quad \left[\begin{array}{ll} & i := i + 4 \\ l_1 : & \textbf{or} \\ & (i, j) := (i + 2, j + 1) \end{array} \right]
\end{array}
\qquad
\begin{array}{l}
L = \{l_0, l_1\},\ V = \{i, j\}, \\
\Theta :\ (i = 2 \wedge j = 0),\ \mathcal{T} = \{\tau_0, \tau_1, \tau_2\}, \\
\tau_0 :\ \langle l_0, l_1, \mathit{true} \rangle \\
\tau_1 :\ \langle l_1,\ l_0,\ (i' = i + 4 \wedge j' = j) \rangle \\
\tau_2 :\ \langle l_1,\ l_0,\ (i' = i + 2 \wedge j' = j + 1) \rangle
\end{array}
$$

Fig. 1. A simple program fragment and the corresponding transition system

Draw CFG for this system [on whiteboard]

Control Flow Graph

The CFG of a transition system is a graph whose vertices are the locations of the transition system and whose edges are the transitions between the locations and are labelled by the transition condition. For a path π in the CFG, the conjunction of the transition conditions of each edge in the graph is called the transition condition for the path, ρ_π.
A **cutset** of a transition system is a subset of locations such that every cycle in the CFG passes through the subset. A location in a cutset is called a **cutpoint**. A **basic path** is a path from one cut point to another that does not pass through any other cut points. Any cycle can be broken up into a sequence of basic paths.

Inductive Assertion Maps

Given a transition system P with a cutset C. A mapping $\eta : C \to \mathit{Assertions}$ is called an inductive assertion map iff for all cutpoint $l, l' \in C$ the following hold:

1. For each basic path π from l_0 to l, $\Theta \wedge \rho_\pi \implies \eta(l)$ (Initialization)
2. For each basic path π from l to l', $\eta(l) \wedge \rho_\pi \implies \eta(l')$ (Consecution)

A k-linear inductive assertion map is one where each $\eta(l)$ is k-linear (See the next slide for the definition of k-linearity).

Linear and Non-Linear Constraints

- A linear constraint over V is an inequality of the form $\sum_{i=1}^{n} a_i x_i + b \leq 0$
- If $b = 0$, we call it a homogenous constraint
- A linear assertion over V is a conjunction of linear constraints
- An assertion consisting of k clauses is said to be k-linear
- Linear assertions correspond to polyhedra in $\mathbb{R}^n$
- Given a set of vectors S, $cone(S)$ is the set of finite positive linear combinations of vectors in S. A cone is said to be *finitely generated* if there exists a finite subset S' of S such that $cone(S') = cone(S)$. A fundamental result states that a cone is finitely generated iff it is a polyhedron
- A non linear constraint over V is an inequality of the form $P(x_1, \ldots x_n) \leq 0$, where P is a polynomial. A non-linear assertion is a conjunction of such constraints. The solution set of a non-linear assertion is called a semi-algebraic set.
- We say a transition system P is linear iff Θ and ρ_τ are linear assertions. We deal only with linear transition systems.

Farkas' Lemma

Let S be an m-linear assertion over variables $\{x_1, \ldots x_n\}$ of the form

$$\sum_{j=1}^{n} a_{ij} x_j + b_i \leq 0, i \in \{1, \ldots m\}$$

If S is satisfiable and ψ is some linear constraint of the form

$$\sum_{j=1}^{n} c_j x_j + d \leq 0$$

Then we have $S \implies \psi$ if and only if there exist non-negative $\lambda_0, \lambda_1, \ldots \lambda_m$ such that

$$\psi \equiv -\lambda_0 + \sum_{i=1}^{m} \lambda_i S_i \leq 0$$

where S_i is $\sum_{j=1}^{n} a_{ij} x_j + b$. Furthermore, S is unsatisfiable if and only if the inequality $1 \leq 0$ can be obtained this way.

Farkas' Lemma

We use a tabular format to represent applications of Farkas' Lemma.

$$
\begin{array}{c|l}
\lambda_0 & -1 \leq 0 \\
\lambda_1 & a_{11}x_1 + \cdots + a_{1n}x_n + b_1 \leq 0 \\
\vdots & \vdots \qquad\qquad \vdots \quad \vdots \\
\lambda_m & a_{m1}x_1 + \cdots + a_{mn}x_n + b_m \leq 0 \\
\hline
 & c_1x_1 + \cdots + c_nx_n + d \leq 0 \leftarrow \psi \\
 & 1 \leq 0 \leftarrow \mathit{false}
\end{array}
\quad S
$$

The antecedents are placed above the line and the consequences below. For each column, the sum of the column entries above the line, with the appropriate multipliers, must be equal to the entry below the line. If a row corresponds to an inequality, the corresponding multiplier is required to be non-negative. This requirement is dropped for rows corresponding to equalities.

Generating Invariants

The approach to generating invariants is based on Farkas' Lemma. Given a transition system P, we write down the initiation and consecution conditions for each cutpoint l, and apply Farkas' Lemma on the conditions to generate the constraints. We parametrize each clause of each inductive assertion by the coefficients of the clause. We end up applying Farkas' Lemma on all basic paths from l_0 to l, on all basic paths from l to l', where l and l' are cutpoints, and for each clause of $\eta(l)$ and $\eta(l')$.
The condition that Farkas' Lemma gives us is a disjunction, corresponding to 2 possibilities - either the system is satisfiable or it is not. Paths for which the system is not satisfiable are known as disabled paths, and those for which it is are known as enabled paths. Therefore at the end, we get an existentially quantified CNF formula where each clause is a disjunction of two assertions.
If there are a initialization paths and b consecution paths, the number of clauses we get would be $ak + bk^2$. It is found that for $k > 1$, solving this becomes too computationally intensive, except for the simplest systems.

Generating Invariants

$$
\begin{array}{l}
\textbf{integer } i, j \textbf{ where } i = 2 \wedge j = 0 \\
l_0 : \textbf{while true do} \\
\quad \left[\begin{array}{ll} & i := i + 4 \\ l_1 : & \textbf{or} \\ & (i, j) := (i + 2, j + 1) \end{array} \right]
\end{array}
\qquad
\begin{array}{l}
L = \{l_0, l_1\},\ V = \{i, j\}, \\
\Theta :\ (i = 2 \wedge j = 0),\ \mathcal{T} = \{\tau_0, \tau_1, \tau_2\}, \\
\tau_0 :\ \langle l_0, l_1, \mathit{true} \rangle \\
\tau_1 :\ \langle l_1,\ l_0,\ (i' = i + 4 \wedge j' = j) \rangle \\
\tau_2 :\ \langle l_1,\ l_0,\ (i' = i + 2 \wedge j' = j + 1) \rangle
\end{array}
$$

Fig. 1. A simple program fragment and the corresponding transition system

Generate invariants for this system [on whiteboard]

Soundness and Completeness

The solutions generated by our method correspond precisely to the set of k-linear inductive assertion maps, ie our method is both sound and complete. This follows as Farkas' Lemma is sound and complete.

Solving Non-Linear Constraints

Non-Linear constraints are generated by the enabled consecution paths, as for an enabled basic path from $\eta(l)$ to $\eta(l')$, we will be multiplying the (unknown) coefficients of $\eta(l)$ by another unknown.
The Linear constraints can be solved by techniques such as Fourier-Motzkin elimination. For Non-Linear constraints, the method used is *Cylindrical Algebraic Decomposition*, introduced by Collins. Another approach is to use Linear Programming and delay solving non-linear constraints until they can be linearized or at least simplified.
Many heuristics have also been developed to simplify solving these constraints.
A tool known as `REDLOG`, that incorporates many of these techniques and heuristics, has been used to compute the solutions.