Decision Procedure for a Restriction of Nonlinear Integer Programming
Matthew J. McGill

Background
- Linear Integer Programming (ILP) is a powerful modeling tool; its decision procedure is NP-Hard.
- Nonlinear Integer Programming (NLP) is, in a sense, too powerful; no decision procedure exists (see Hilbert’s 10th problem).
- The study of various NLP restrictions is an active research area.

A Restricted NLP Class
- Let $Q$ be a system of inequalities having one of these three forms:
  - $a \cdot x_0 \leq b + x_1 + x_2 + \ldots + x_k$
  - $a \cdot x_0 \leq b \cdot x_1 \cdot x_2 \cdot \ldots \cdot x_k$
  - $a \leq x$
  where $a, b, k$ are non-negative integers and variables $x_0 \ldots x_k$ range over positive integers.
- $\text{vars}(Q)$ denotes the set of variables in $Q$. 

A Restricted NLP Class
- An assignment to $Q$ is a function $\alpha : \text{vars}(Q) \rightarrow \mathbb{Z}^+$ from variables to positive integers.
- $\alpha$ is a solution to $Q$ iff every inequality in $Q$ is true after substituting $\alpha(x)$ for each variable $x$. 
Decision Problem

Given $Q$ and positive integer $n$, decide whether there exists a solution $\alpha$ to $Q$ such that, for all $x \in \text{vars}(Q)$, $\alpha(x) \leq n$.

Without loss of generality, assume that for all $x \in \text{vars}(Q)$, $Q$ contains some inequality $a \cdot x \leq \psi$.

Decision Procedure

1. Inputs: $Q$, $n > 0$
2. For each $x$ in $\text{vars}(Q)$ Do: $\alpha_0(x) := n$  # Initialize bounds
3. $i := 0$
4. Repeat
5. $i := i + 1$
6. For each $x$ in $\text{vars}(Q)$ Do: $\alpha(x) := \alpha_{i-1}(x)$
7. For each $a \cdot x \leq \psi$ in $Q$ Do:
8. $\alpha(x_0) := \min \{ \alpha(x_0), \text{eval}(\psi, \alpha_{i-1}) / a \}$  # (Integer division)
9. For each $a \leq x$ in $Q$ do:
10. If $a > \alpha(x_0)$ Then:  # Check for crossed bounds
11. Return False
12. Until $\alpha_{i-1} = \alpha_i$  # Repeat until fixpoint
13. Return True  # $\alpha_i$ is now a solution to $Q$

Example 1

$Q = \{ 12 \cdot x_1 \leq x_2 \cdot x_3, \\
3 \cdot x_2 \leq x_1 + x_3, \\
5 \cdot x_3 \leq x_1 + x_2 \}$

$n = 100$

Example 2

$Q = \{ 3 \cdot x_1 \leq x_2 + x_3, \\
2 \cdot x_2 \leq x_1, \\
x_3 \leq x_1 \}$

$n = 100$
Sound? Yes.

Complete (i.e. always terminates)? Yes.
Analysis

sound? yes.
complete (i.e. always terminates)? yes.
time polynomial in size of Q?

Analysis

sound? yes.
complete (i.e. always terminates)? yes.
time polynomial in size of Q? yes, but not quite as clear. hints to the proof:
- if a=1 for each inequality (i.e. no division), each iteration fixes some bound.
- each division performed O(log n) times at most for initial upper bound n.

ORM

object role modeling (ORM) is a conceptual modeling language.
modelers use ORM to model an information domain without biasing towards a particular database design.

Example ORM Model

constraints
roles

entity type
fact type
value type

Project
(has

ProjName)
Example ORM Model with Legal Instance

Example ORM Model with Illegal Instance

Example ORM Model with Illegal Instance

ORM Consistency

An ORM model $M$ is consistent if and only if there is a legal instance of $M$ in which each object (fact) type is non-empty.

Problem is NP-Hard, in general.
ORM\textsuperscript{−}

- Smaragdakis et al. [1] identify ORM\textsuperscript{−}, a strict subset of ORM for which consistency can be checked in polynomial time.
- Smaragdakis et al. [1] reduce ORM\textsuperscript{−} consistency problem to the restricted NLP class we just examined.
- Given an ORM\textsuperscript{−} model $M$, produce a system $Q$ that has a solution iff $M$ is consistent.

**Reduction**

- Add a variable representing the cardinality of each object type, fact type, and role.

![Diagram](image)

**Reduction**

- Add inequalities relating object types, roles, and fact types.

\[
\begin{align*}
    x_{\text{proj}} &\leq x_{r1} \\
    x_{\text{name}} &\leq x_{r2} \\
    x_{r1} &\leq x_{\text{has}} \\
    x_{r2} &\leq x_{\text{has}} \\
    x_{\text{has}} &\leq x_{r1} \cdot x_{r2}
\end{align*}
\]

**Reduction**

- Add inequalities that encode each constraint.

\[
\begin{align*}
    x_{\text{proj}} &\leq x_{r1} \\
    x_{\text{name}} &\leq x_{r2} \\
    x_{r1} &\leq x_{\text{has}} \\
    x_{r2} &\leq x_{\text{has}} \\
    x_{\text{has}} &\leq x_{r1} \cdot x_{r2}
\end{align*}
\]
Reduction

- Add inequalities that require each object/fact type to be non-empty.

Extending ORM-

- We conducted a study of ORM models at LogicBlox, inc.
- Found that developers rely heavily on objectification and, to a lesser extent, external uniqueness constraints, features not in ORM⁻.
- We extended the ORM⁻ consistency algorithm to support both features.

Questions?