DANSAS 2011, Odense

(using)

Complexity Analysis of Monotonicity-Constraint Transition System

Amir M. Ben-Amram

Joint work with Michael Vainer,
Tel-Aviv Academic College
Problem Definition

INPUT: program $p$

QUESTIONS:

Does $p$ run in polynomial time (etc)?

Does $p$ run in linear/quadratic/cubic... time?

etc...

There are various motivations
Abstract and Conquer

Realistic programming languages are Turing-complete
Properties of interest are undecidable

We study simple models that can be used as an abstraction of real programs.

Not a new idea (models include: finite automata, $\omega$-automata, pushdown automata, call graphs, size-change graphs, Petri nets, counter programs...)

Theoretical goals: develop understanding and insights
Solve problems in the model, or prove the limits of decidability. Also: classify complexity of problems.
Practical application:

source program → abstraction → model

FRONT END

analysis

BACK END
Constraint Transition Systems (CTS)

Program = a state-transition system (with variables) = finite control-flow graph + transition constraints

mixing lists

\[
\text{mix}(x,y) = \\
\text{let } x' = y, \ y' = \text{tl } x \ \text{in} \\
x :: \text{mix}(x', y')
\]

...
Types of Transition Systems

• Defined by what kind of constraints may be used (and their domain)

• monotonicity constraints: conjunction of (in)equalities

• domain:
M.C.s + well-founded data = Size-Change Termination

Fundamental result: [Lee, Jones, B 2001]
- Termination is decidable (PSPACE-complete).

Practical relevance:
- Representation of different kinds of source programs
  - works best with computation on lists, terms, etc.
Monotonicity Constraints (cont.)

• Codish, Lagoon, Stuckey 2005 + recent works:
  Integer setting
  (instead of well-founded one) \((MC, \mathbb{Z})\)-CTS

\[ x > x' \land y \leq y' \land x \geq y \]

Fundamental result: [B 2011]
• Termination is decidable (PSPACE-complete).
Monotonicity Constraints for Termination

- functional programs: LJB 2001, MV 2006, ...
- imperative programs: A 2006, SMP 2009, CGBFG 2011, ...
- HO functional programs: SJ 2005
- Term Rewriting Systems: TG 2005, CGBFG 2011, ...

The common point:
infinite computation $\Rightarrow$ infinite transition-sequence.
CTS in Complexity Analysis

- Albert et al., ESOP 2007 (Analysing Java Bytecode)
- Alias et al., SAS 2010 (Analysing C programs)

**Common point:**

- **Affine constraints over integers**
  \[ x' = 2y \land y' \leq x - 1 \]
- **Termination of abstract program undecidable**
(MC, Z)-CTS is a special case of (Aff, Z)-CTS

- affine constraints can be “relaxed” to MCs - front ends can be reused

- both works use the concept of ranking functions.

- for (MC, Z)-CTS, we have a complete characterization and synthesis of ranking functions.
Our results: complexity analysis in $(MC, \mathbb{Z})$-CTS

- Decision problems:
  
  **Termination**: there are no infinite transition sequences (from an initial state)

  **Bounded termination**: there is a function $B$ such that the length of any transition sequence is bounded by $B(x^{\text{init}}, y^{\text{init}}, ...)$

\[
\begin{align*}
x > x' & \land y \leq y' & \land x \geq y \\
\text{bound: } x^{\text{init}} - y^{\text{init}}
\end{align*}
\]
Polynomial Termination: bounded termination, where $B$ is a polynomial.

More refined problems: $B$ is of degree $k$ ...
Bounded Termination

**THM** The bounded termination problem is decidable for \((MC, \mathbb{Z})\)-CTS (PSPACE complete).

**Proof components:**
- a decision algorithm
- a soundness proof
- a completeness proof
- PSPACE upper and lower bounds

**Not a component:**
- ranking functions
Algorithm 1

For a program point $\ell$, variable $X$ is called \textit{bounded} at $\ell$ if:

$$\forall \text{ reachable state } (\ell, \ldots, X, \ldots) : \exists i, j \ x_i^{\text{init}} \leq X \leq x_i^{\text{init}}$$

1. Identify bounded variables

2. Check whether the program terminates if only bounded variables are considered.

\textit{soundness} of conclusion: state-space is bounded...
\textit{completeness:} hmm...
input a, b
if (*) then x := b;  y := *
else  y := b;  x := *

while ( x>a \land y>a ) who's bounded here?
  x--; y --
Algorithm 2

1. Ensure that the transition system is *stable*

2. Perform Algorithm 1.

Stabilization is a form of control-flow refinement
After stabilization

input $a$, $b$
if (*) then $x := b$; $y := *$
else $y := b$; $x := *$

while ($x > a$ \&\& $y > a$)
$x --$; $y --$

$x \leq b$
$y \leq b$
Algorithm 2

1. Ensure that the transition system is *stable*

2. Perform Algorithm 1.

**soundness** of conclusion: state-space is bounded...

**completeness**: a counter-example can be constructed.
Polynomial Termination

**THM** A transition system is bounded-terminating IFF It is polynomially terminating.

- Falls out easily from the decision algorithm (suppose that $0 \leq X, Y, Z \leq n \ldots$)
Finer distinctions

If there is a polynomial bound, can we get the precise degree? (assume univariate polynomials)

THM The precise degree of the bounding polynomial is computable.
Approach 1:
a global ranking function.

⇒ g.r.f. expression can be exponential

Approach 2:
“closure based”

⇒ proves PSPACE complexity of the decision problem - is there a bound of degree k
Back to concrete programs

source program \rightarrow abstraction \rightarrow model

FRONT END \rightarrow BACK END
Abstracting a program as a CTS

• imperative programs: transition = basic block

• Suppose that there are no procedure calls (“flat” imperative program)

• basic block = $O(1)$ time
  polynomially bounded CTS $\Rightarrow$ polynomial time

• In fact it suffices that there is no recursion
Abstracting a program as a CTS

• functional programs: transition = function call

Length of transition sequence = length of call chain = depth of call stack

We obtain a polynomial bound on the stack depth
⇒ exponential bound on time
⇒ and polynomial space, if heap space cannot explode
Conclusion

• CTS abstraction is useful for complexity analysis

• \((MC, \mathbb{Z})\)-CTS is simple enough to obtain decision procedures; expressive enough to be useful
  (based on: experience with termination; a skim of complexity analysis examples; [GSZH 11])

• Extensional completeness

• Future challenges: scalable algorithms, multivariate bounds, improving precision beyond \(MC\) s