Set Constraints, Pattern Match Analysis, and SMT

Joseph Eremondi
University of British Columbia
TFP 2019
Introduction
Why We Care: Pattern Match Analysis

Motivation: Pattern Match Analysis

numNodes tree =
  case tree of
  Leaf data -> 1
  Branch [child] ->
    1+numNodes child
  Branch (h:t) ->
    1+foldMap numNodes t

Safe if tree is never (Branch [])

Model set of values tree can safely take
Why We Care: Pattern Match Analysis

Pattern Match Analysis (ctd.)

Return contains 1 iff

\[
\text{tree contains Leaf } _{-}
\]

\[
\text{numNodes } \text{tree} = \\
\text{case tree of} \\
\text{Leaf data} \rightarrow 1 \\
\text{Branch [child]} \rightarrow 1 + \text{numNodes child} \\
\text{Branch (h:t)} \rightarrow 1 + \text{foldMap numNodes t}
\]

Reachable iff >2 children

Result set defined recursively

Model set of values \textbf{numNodes} can return
Reasoning Backwards

```haskell
  case numNodes tree of
    1 ->
      let (Leaf data) = tree
      in ...
    _ -> ...

Know this is safe: numNodes returns 1, so input is Leaf

All can be modeled with (unrestricted) Set Constraints!
What We Need

• Need to reason about:
  – Sets of values
  – Constructor application and projection

• Must support:
  – Variables
  – Negative constraints
  – Boolean combinations (implication)

... Set Constraints!
Anatomy of Set Constraints

\[ X \subseteq f(Y) \land Y \not\subseteq f^{-1}(Z) \]

Variables to solve for

Positive constraint

Constructor application

Negative constraint

Projection
Solve all the set constraints!
Set Constraints are the Monadic Class

Leo Bachmair  Harald Ganzinger  Uwe Waldmann

Abstract

We investigate the relationship between set constraints and the monadic class of first-order formulas and show that set constraints are essentially equivalent to the monadic class. From this equivalence we can infer that the satisfiability problem for set constraints is complete for NEXPTIME. More precisely, we prove that this problem has a lower bound of NTIME(n^c/\log n), for some c > 0, as a lower bound for the problem. The relationship between set constraints and the monadic class allows us to extend decision and complexity results for certain practically useful constraints by diagonalization (i.e., by projections). By extensions of set constraints, in particular “negative” set constraints, we end up at the monadic class with projections and subterm equality tests.

The satisfiability of monadic formulas can be reduced to the satisfiability of monadic formulas via length order n^2, conversely, the satisfiability of monadic formulas can be reduced to the satisfiability of set constraints via length order n^2/\log n. As a consequence, the satisfiability of set constraints is complete for NEXPTIME, a result that was left open in (Aiken and Wimmers 1992). More precisely, we establish NTIME(n^c/\log n), for some c > 0, as a lower bound for the problem. The relationship between set constraints and the monadic class allows us to extend decision and complexity results for certain practically useful constraints by diagonalization (i.e., by projections). By extensions of set constraints, in particular “negative” set constraints, we end up at the monadic class with projections and subterm equality tests.

Solve all the set constraints?
Theory vs. Practice

Set Constraints
- Negative Constraints
  - With Projections

Boolean Combos

Solved In Practice
✔
Solving Set Constraints
Monadic First Order Logic

$x \in S \iff P_S(x)$

- Predicate for each sub-expr of constraint (à la Tseitin)
- Takes exactly one argument

Semantics of sets as first-order formulas e.g.

$$\forall x. P_{-S}(x) \iff \neg P_S(x)$$
Set Constraint As Formula

\begin{align*}
S & \subseteq T \\
\iff \\
\forall x. \ P_S(x) & \implies P_T(x) \\
S & \not\subseteq T \\
\iff \\
\exists x. \ P_S(x) & \land \neg P_T(x)
\end{align*}
Finite Model Property

\[ n \text{-predicate formula holds} \iff \text{formula has model with } \leq 2^n \text{ elems} \]

✔ Decidable... by trying \(2^{2^n}\) models ✗
Deciding Monadic Logic

• Objects in model?
  – Eq. Classes of Predicate values

• Model is:
  – Subset of possible Eq. classes

• Idea: model domain as a function
  – Type $\mathbb{B}^n \to \mathbb{B}$
  – Filters equivalence classes
Domain $D \subseteq \mathbb{B}^n$

0110011 ... 01001

Bit $i$ set iff $P_i(x)$ holds for $x$
Narrowing the Domain

Uninterpreted function

\( \text{inDomain} : \mathbb{B}^n \rightarrow \mathbb{B} \)

Qualify formulas e.g.

\[ S \subseteq T \]

\[ \forall x \in \mathbb{B}^n. \text{inDomain}(x) \]

\[ \implies (P_S(x) \implies P_T(x)) \]
Solving the Constraints

Solver finds:

- impl. for \texttt{inDomain} : \mathbb{B}^n \rightarrow \mathbb{B}
- impl. for each constructor
  
  \[ f : (\mathbb{B}^n)^k \rightarrow \mathbb{B}^n \]
Results
Implementation

Translate to SMT
• written in Haskell
• Solved with Z3

Pattern Match Analysis
• Modified Elm compiler
• Emits set constraints
The Experiment

• Compile the rtfeldman(elm-css) library
• Measure slowdown:
  – Exhaustiveness checking (default)
  – Pattern-match analysis
Results

- **Bad news**
  - Min slowdown factor 13 (273 ms)
  - Max slowdown factor 468753 (30 minutes)

- **Good news**
  - Worst case had domain $\subseteq \mathcal{B}_{50}^{50}$
  - $2^{50}$ ns $\approx$ 13 days $>$ 30 minutes
Limitations

• It’s slow
  – But not heat-death of the universe slow
• Error messages not great
  – Need UNSAT core of set constraints
Open Questions

• Is Z3 the best option?
  – CVC4 more configurable?
  – QBF or OBDD?
• What causes specific case to blow-up?
  – Large number of projections?
• What heuristics can help with the speedup?
Set constraints can be solved in practice... but require a bit of patience!

Preprint link at eremondi.com