

Set Constraints

Loose Ends

- Solving recursive simultaneous equations for over- and under-approximations

Terms

- A finite set of *constructors*
 $a, b, c, f, g, h \in C$
- Each constructor c has an *arity* $a(c)$
 - Can be 0
- Terms
 - $T = \{ f(t_1, \dots, t_{a(f)}) \mid f \in C, t_i \in T \}$

Definition of Set Constraints

Syntax

$$\bigwedge_i E_{i1} \subseteq E_{i2}$$

$$E ::= 0 \mid \\ E_1 \cap E_2 \mid \\ E_1 \cup E_2 \mid \\ \neg E \mid \\ c(E_1, \dots, E_{a(c)}) \mid \\ \nu$$

Semantics

$$\{ \theta \mid \bigwedge_i \theta(E_{i1}) \subseteq \theta(E_{i2}) \}$$

$$\theta(0) = \emptyset \\ \theta(E_1 \cap E_2) = \theta(E_1) \cap \theta(E_2) \\ \theta(E_1 \cup E_2) = \theta(E_1) \cup \theta(E_2) \\ \theta(\neg E) = 1 - \theta(E) \\ \theta(c(t_1, \dots, t_n)) = \{ c(t_1, \dots, t_n) \mid t_i \in \theta(E_i) \} \\ \theta(\nu) = \nu$$

Why Should We Care?

- Most of the theories we have looked (and will look at) are special cases of set constraints
 - With no encoding required
- Exception: Integer constraints

SAT

$$\bigwedge_i E_{i1} \subseteq E_{i2}$$

$$E \subseteq 0$$

$$E ::= 0 \mid \\ E_1 \cap E_2 \mid \\ E_1 \cup E_2 \mid \\ \neg E \mid \\ c(E_1, \dots, E_{a(c)}) \mid \\ \nu$$

$$E ::= 0 \mid \\ E_1 \cap E_2 \mid \\ E_1 \cup E_2 \mid \\ \neg E \mid \\ \nu$$

Dataflow Analysis

$$\bigwedge_i E_{i1} \subseteq E_{i2}$$

$$\bigwedge_i v_i = E_i$$

$$E := 0 \mid$$

$$E_1 \cap E_2 \mid$$

$$E_1 \cup E_2 \mid$$

$$\neg E \mid$$

$$c(E_1, \dots, E_{a(c)}) \mid$$

$$v$$

$$E := 0 \mid$$

$$E_1 \cap E_2 \mid$$

$$E_1 \cup E_2 \mid$$

$$c \mid$$

$$v$$

Profs. Aiken & Dill CS357 Lecture 16

7

Type Equations

$$\bigwedge_i E_{i1} \subseteq E_{i2}$$

$$\bigwedge_i E_{i1} = E_{i2}$$

$$E := 0 \mid$$

$$E_1 \cap E_2 \mid$$

$$E_1 \cup E_2 \mid$$

$$\neg E \mid$$

$$c(E_1, \dots, E_{a(c)}) \mid$$

$$v$$

$$E := c(E_1, \dots, E_{a(c)}) \mid$$

$$v$$

Profs. Aiken & Dill CS357 Lecture 16

8

Points-To Analysis, OCFA, Receiver-Class Analysis

$$\bigwedge_i E_{i1} \subseteq E_{i2}$$

$$\bigwedge_i E_{i1} \subseteq E_{i2}$$

$$E := 0 \mid$$

$$E_1 \cap E_2 \mid$$

$$E_1 \cup E_2 \mid$$

$$\neg E \mid$$

$$c(E_1, \dots, E_{a(c)}) \mid$$

$$v$$

$$E := c(E_1, \dots, E_{a(c)}) \mid$$

$$v$$

Profs. Aiken & Dill CS357 Lecture 16

9

Others

- Context-free reachability
- Datalog
 - How much?
- Equations on finite automata
 - And tree automata
- A significant chunk of logic
 - More on this later

Profs. Aiken & Dill CS357 Lecture 16

10

Historical Context

- Algorithms were first developed for each of these domains separately
- Then people started mixing and matching
 - '70's & '80's
- But this led to problems
 - Especially with anything resembling negation
 - Restricted algorithms, heuristics, and punts

Profs. Aiken & Dill CS357 Lecture 16

11

Context

- Gradually the definition of set constraints emerged
 - Or rather, several related definitions emerged
- Open problems resolved in the early '90's

Profs. Aiken & Dill CS357 Lecture 16

12

Examples

$x = \text{cons}(y,x) \cup \text{nil}$

Examples

$x = \text{tree}(x,x) \cup y$

Examples

$x = \text{cons}(y,\text{cons}(z,x)) \cup \text{nil}$

More Examples

$x \subseteq \text{cons}(y,x) \cup \text{nil}$

More Examples

$x = \text{cons}(y,\neg x) \cup \text{nil}$

More Examples

$x = [\text{cons}(y,\text{cons}(y,x)) \cap \text{cons}(y,\text{cons}(y,\text{cons}(y,x)))] \cup \text{nil}$

More Examples

$$x \subseteq \neg x$$

$$x = \neg x$$

Deriving a Decision Procedure

- Where is the finite structure?
- Not in the terms, or solution sets
- Observation
 - There is a fixed set of variables

Closed (Hyper) Graphs

$$x = c(\neg x) \cup b$$

Second Example

$$f(x) \subseteq x \cup y$$

$$y \subseteq 0$$

Third Example

$$c(x,y) \subseteq x$$

$$b \subseteq x$$

$$b \subseteq y$$

Decidability Plan

- Rewrite the constraints so that the hypergraph is explicit
- Check whether there is a subset N of the nodes of the hypergraph that is *closed*
 - For every constructor c , for all $\{a_1, \dots, a_{a(c)}\} \subseteq N$ there exists $a_0 \in N$ such that $c(a_1, \dots, a_{a(c)}) \subseteq a_0$

The Goal

- Get the constraints into the form

$$f(a_1, \dots, a_n) \subseteq a_{n+1} \cup \dots \cup a_m$$

Flattening

- Flatten expressions
 - Replace each subexpression $f(e_1, \dots, e_n)$ by a fresh variable x
 - Add equation $x = f(e_1, \dots, e_n)$
- Replace $x = e$ by $x \subseteq e$ and $e \subseteq x$

Move Everything to the Left-Hand Side

Replace

$$x \subseteq y$$

by

$$x \cap \neg y \subseteq 0$$

Switch to Atoms

- Replace each variable and 1 ($\neg 0$) by a disjunction of *atoms*
 - Conjunctions in which all variables appear once, either positively or negatively
- If variables are x and y , atoms are
 - $x \cap y$
 - $x \cap \neg y$
 - $\neg x \cap y$
 - $\neg x \cap \neg y$
- x is replaced by
 - $(x \cap y) \cup (x \cap \neg y)$

Repeat Until Closure

- Rewrite boolean expressions in DNF
- $f(\dots, x \cup y, \dots) = f(\dots, x, \dots) \cup f(\dots, y, \dots)$
- $f(\dots, 0, \dots) = 0$
- $\neg f(a_1, \dots, a_n) = \bigcup_{g \neq f} g(1, \dots, 1) \cup f(\neg a_1, 1, \dots, 1) \cup \dots \cup f(1, \dots, 1, \neg a_n)$
- $X \cap 0 = 0$
- $f(\dots) \cap g(\dots) = 0$ if $f \neq g$
- $a_i \cap a_j = 0$ if $i \neq j$
- $f(a_1, \dots, a_n) \cap f(a'_1, \dots, a'_n) = f(a_1 \cap a'_1, \dots, a_n \cap a'_n)$
- $X \cup Y \subseteq Z = X \subseteq Z$ and $Y \subseteq Z$

Where Are We?

- All constraints are now of the form

$$a_0 \cap f(a_1, \dots, a_n) \subseteq 0$$

or

$$a \subseteq 0$$

- This tells us
 - Where edges can *not* be
 - Which nodes can *not* have terms

Switching Sides

Replace all constraints of the form

$$b_0 \cap f(a_1, \dots, a_n) \subseteq 0$$

...

$$b_m \cap f(a_1, \dots, a_n) \subseteq 0$$

by

$$f(a_1, \dots, a_n) \subseteq \neg(b_0 \cup \dots \cup b_m)$$

Examples Revisited

$$x = c(\neg x) \cup b$$

Examples Revisited

$$f(x) \subseteq x \cup y$$

$$y \subseteq 0$$

Examples Revisited

$$c(x, y) \subseteq x$$

$$b \subseteq x$$

$$b \subseteq y$$

Testing Closure

- We still need to check whether the hypergraph has a closed subset

- Algorithm

- If $a \subseteq 0$, for each $f(a_1, \dots, a_n) \subseteq b_1 \cup \dots \cup b_m$
 - if $a = a_i$, drop the constraint
 - o.w., replace a on the right-hand side by 0
- If $f(a_1, \dots, a_n) \subseteq 0$
 - Add $a_i \subseteq 0$ for some i

Examples Revisited

$$f(x) \subseteq x \cup y$$

$$y \subseteq 0$$

Examples Revisited

$$f(x) \subseteq x \cap y$$
$$y \subseteq 0$$

Examples Revisited

$$f(x) \subseteq x \cap y$$
$$y \subseteq 0$$
$$a \subseteq x$$

Relationship to Logic

- Set constraints are closely related to the *monadic class*
- Provides an alternative proof of decidability
 - And complexity

Set Constraints in Logic

- The monadic class consists of
 - First order formulas
 - No function symbols
 - Unary predicates only

Set Constraints in the Monadic Class

- $P_E(x)$ means "x is in E"
 - One predicate for each subexpression
- $E \subseteq F = P_E(x) \supset P_F(x)$
- $P_0(x) = \text{false}$
- $P_{E \cup F}(x) = P_E(x) \vee P_F(x)$
- $P_{E \cap F}(x) = P_E(x) \wedge P_F(x)$
- $P_{\neg E}(x) = \neg P_E(x)$
- $P_{f(E_1, \dots, E_n)}(f(x_1, \dots, x_n)) = P_{E_1}(x_1) \wedge \dots \wedge P_{E_n}(x_n)$
- $P_{g(\dots)}(f(x_1, \dots, x_n)) = \text{false}$ if $g \neq f$

Properties

- The monadic class is decidable
 - Loewenheim (1912)
- And complete for NEXPTIME
 - Lewis (1980)
- There is also a simple encoding of the monadic class in set constraints