**DECISION PROCEDURE**

Language: +, -, <, 1 multiplication by rational coefficients

We want to decide whether

∃y1... ∃yn(L1 ^ … ^ Ln) is satisfiable in the structure of real numbers

Fourier-Motzkin eliminates all quantifiers (keeping logical equivalence over the structure of the reals) until we get a ground constraints that evaluates to true or false

**General instructions**

* remove negative literals

t ≠ u t<u v t > u

* rewrite positive literals by isolating x as a first member, so can to get only literals of the form

x = t x > t x < t

where t is a linear polynomial without x

* remove x if is has a “definition”

x = t ^ A A (t/x)

* now collect terms “above” and “below” x so that the formula we are handling is of the kind (α) ^ (β) ^ (γ)

(α) x < t1 ^ … ^ x < tn

(β) u1 < x ^ … ^ um < x

(γ) this is a formula where x does not occur (or 0=0 if x occur always)

* if (α) is empty, the result of eliminating x is (γ)
* if (β) is empty, the result of eliminating x is (γ)
* if (α) and (β) are not empty, the result is

(γ) ^ (ui < ti) that is, all possible inequalities obtained picking ui from (β) and ti from (γ)

**Example 1**

3y + x < 0 ^ x+6-2y > 0 ^ y > 0 ^ x <1

Rewriting to isolate x, we get

x < -3y ^ x > 2y-6 ^ y>0 ^ x<1

Eliminating of x gives

-6 + 2y < 1 ^ -6+2y < -3y ^ y>0

Rewriting to isolate y, we get

y<7/2 ^ y<6/5 ^ y>0

Eliminating y, we get

0<7/2 ^ 0<6/5

This is true, hence the answer is SAT

Linear arithmetic over real and integers

There are algorithm (Fourier-monzkin) for real and Cooper for integers (more complicated)

Non linear arithmetic

* real → algorithm Collins
* Integers → no algorithm! This problem is undecided (ten hilbert problem)

**Integer difference logic**

As a further example of decision procedure, let us, show how to check satisfiability in Z ( the integers!) of literals of the form

x - y ≤ n x≤ n x-y = n

x - y /≤ n x/≤ n x-y ≠ n

We first reduce to literals of the kind x-y ≤ n - In fact

* rewrite x ≤ n as x-0 ≤n
* rewrite x-y = n as x-u ≤n ^ y-x≤-n
* rewrite x-y/≤ n as x-y>n x-y ≥ n+1 finally y-x≤-n-1
* rewrite x /≤ n as x >n x ≥n+1 and finally 0-x ≤-n-1
* rewrite x-y ≠ n as x-y <n v x-y > n and finally as x-y ≤n-1

[this last case requires a case-split complicity blows up!]

**Characteristic**

Set with 3 elements

Consider the subset of 2 elements

Associate value 1 to both element

And 0 to the external one

This is the sort Bool of z3

Xs(a) = 1 iff a € S

From set i get to the characteristics function and vice versa

S = {a | Xs (a) = 1}

D P unary predicate

(Declare-fun P(D) Bool) you are declaring a characteristic function

*Loves* a person love an animal

Loves(xperson, yanimal)

(Declare-fun loves (Person Animal) Bool) returns a boolean

In z3 no predicates, bu only functions (sort and functions)

But among the sort we have the boolean, we can e code the relation with predicates as a function with Bool

Ex1

(Declare-constant t11 Int)

…

Assert all conjunction

(Check sat)

(Get value (t11 t12…))

T12 ≥ t11+2 is write as

(>= t12 (+ t11 2))

Ex2

(Declare-constant numberoffrenchfries int)

(Assert (>= numberoffrenchfries 0))

Assert numberoffrenchfires\*2.75+……….

= 15.05