A Simplex variant:

Transform the satisfiability problem into the form

$$A\vec{x} = \vec{0}$$
$$\vec{l} \leq \vec{x} \leq \vec{u}$$

(where l_i may be $-\infty$ and u_i may be $+\infty$).

Relation to optimization problem is obscured.

But: More efficient if one needs an incremental decision procedure, where inequations may be added and retracted (Dutertre and de Moura 2006).

1.5 Non-linear Real Arithmetic

Tarski (1951): Quantifier elimination is possible for *non-linear* real arithmetic (or more generally, for real-closed fields). His algorithm had non-elementary complexity, however.

An improved algorithm by Collins (1975) (with further improvements by Hong) has doubly exponential complexity: Cylindrical algebraic decomposition (CAD).

Implementation: QEPCAD.

Cylindrical Algebraic Decomposition

Given: First-order formula over atoms of the form $f_i(\vec{x}) \sim 0$, where the f_i are polynomials over variables $\vec{x}$.

Goal: Decompose $\mathbb{R}^n$ into a finite number of regions such that all polynomials have invariant sign on every region X:

$$\begin{aligned}\forall i\ (\ &\forall \vec{x} \in X.\ f_i(\vec{x}) < 0 \\ \vee\ &\forall \vec{x} \in X.\ f_i(\vec{x}) = 0 \\ \vee\ &\forall \vec{x} \in X.\ f_i(\vec{x}) > 0\ )\end{aligned}$$

Note: Implementation needs exact arithmetic using algebraic numbers (i. e., zeroes of univariate polynomials with integer coefficients).

1.6 Real Arithmetic incl. Transcendental Functions

Real arithmetic with exp/log: decidability unknown.

Real arithmetic with trigonometric functions: undecidable

The following formula holds exactly if $x \in \mathbb{Z}$:

$$\exists y\ (\sin(y) = 0 \wedge 3 < y \wedge y < 4 \wedge \sin(x \cdot y) = 0)$$

(note that necessarily $y = \pi$).

Consequence: Peano arithmetic (which is undecidable) can be encoded in real arithmetic with trigonometric functions.

However, real arithmetic with transcendental functions is decidable for formulas that are *stable under perturbations*, i. e., whose truth value does not change if numeric constants are modified by some sufficiently small ε.

Example:

Stable under perturbations: $\exists x\ x^2 \leq 5$

Not stable under perturbations: $\exists x\ x^2 \leq 0$
(Formula is true, but if we subtract an arbitrarily small $\varepsilon > 0$ from the right-hand side, it becomes false.)

Unsatisfactory from a mathematical point of view, but sufficient for engineering applications (where stability under perturbations is necessary anyhow).

Approach:

Interval arithmetic + interval bisection if necessary (Ratschan).

Sound for general formulas; complete for formulas that are stable under perturbations; may loop forever if the formula is not stable under perturbations.

1.7 Linear Integer Arithmetic

Linear integer arithmetic = Presburger arithmetic.

Decidable (Presburger, 1929), but quantifier elimination is only possible if additional divisibility operators are present:

$\exists x\ (y = 2x)$ is equivalent to $\text{divides}(2, y)$ but not to any quantifier-free formula over the base signature.

Cooper (1972): Quantifier elimination procedure, triple exponential for arbitrarily quantified formulas.

The Omega Test

Omega test (Pugh, 1991): variant of Fourier–Motzkin for conjunctions of (in-)equations in linear integer arithmetic.

Idea:

- Perform easy transformations, e. g.:
  $3x + 6y \leq 8 \;\mapsto\; 3x + 6y \leq 6 \;\mapsto\; x + 2y \leq 2$
  $3x + 6y = 8 \;\mapsto\; \bot$
  (since $3x + 6y$ must be divisible by 3).
- Eliminate equations
  (easy, if one coefficient is 1; tricky otherwise).
- If only inequations are left:
  no real solutions $\rightarrow$ unsatisfiable for $\mathbb{Z}$
  "sufficiently many" real solutions $\rightarrow$ satisfiable for $\mathbb{Z}$
  otherwise: branch

What does "sufficiently many" mean?

Consider inequations $ax \leq s$ and $bx \geq t$ with $a, b \in \mathbb{N}^{>0}$ and polynomials s, t.

If these inequations have real solutions, the interval of solutions ranges from $\frac{1}{b}t$ to $\frac{1}{a}s$.

The longest possible interval of this kind that does not contain any integer number ranges from $i + \frac{1}{b}$ to $i + 1 - \frac{1}{a}$ for some $i \in \mathbb{Z}$; it has the length $1 - \frac{1}{a} - \frac{1}{b}$.

Consequence:

If $\frac{1}{a}s > \frac{1}{b}t + (1 - \frac{1}{a} - \frac{1}{b})$, or equivalently, $bs \geq at + ab - a - b + 1$ is satisfiable, then the original problem must have integer solutions.

It remains to consider the case that $bs \geq at$ is satisfiable (hence there are real solutions) but $bs \geq at + ab - a - b + 1$ is not (hence the interval of real solutions need not contain an integer).

In the latter case, $bs \leq at + ab - a - b$ holds, hence for every solution of the original problem:

$t \leq bx \leq \frac{b}{a}s \leq t + (b - 1 - \frac{b}{a})$

and if x is an integer, $t \leq bx \leq t + \lfloor b - 1 - \frac{b}{a} \rfloor$

$\Rightarrow$ Branch non-deterministically:
Add one of the equations $bx = t + i$ for $i \in \{0, \ldots, \lfloor b - 1 - \frac{b}{a} \rfloor\}$.

Alternatively, if $b > a$:
Add one of the equations $ax = s - i$ for $i \in \{0, \ldots, \lfloor a - 1 - \frac{a}{b} \rfloor\}$.

Note: Efficiency depends highly on the size of coefficients. In applications from program verification, there is almost always some variable with a very small coefficient. If all coefficients are large, the branching step gets expensive.

Branch-and-Cut

Alternative approach: Reduce satisfiability problem to optimization problem (like Simplex). ILP, MILP: (mixed) integer linear programming.

Two basic approaches:

Branching: If the simplex algorithm finds a solution with $x = 2.7$, add the inequation $x \leq 2$ or the inequation $x \geq 3$.

Cutting planes: Derive an inequation that holds for all real solutions, then round it to obtain an inequation that holds for all integer solutions, but not for the real solution found previously.

Example:

Given:
$$\begin{aligned} 2x - 3y &\leq 1 \\ 2x + 3y &\leq 5 \\ -5x - 4y &\leq -7 \end{aligned}$$

Simplex finds an extremal solution $x = \frac{3}{2}$, $y = \frac{2}{3}$.

From the first two inequations, we see that $4x \leq 6$, hence $x \leq \frac{3}{2}$. If $x \in \mathbb{Z}$, we conclude $x = \lfloor x \rfloor \leq \lfloor \frac{3}{2} \rfloor = 1$.

$\Rightarrow$ Add the inequation $x \leq 1$, which holds for all integer solutions, but cuts off the solution $(\frac{3}{2}, \frac{2}{3})$.

In practice:

Use both: Alternate between branching and cutting steps.
Better performance than the individual approaches.

1.8 Difference Logic

Difference Logic (DL):

Fragment of linear rational or integer arithmetic.

Formulas: conjunctions of atoms $x - y < c$ or $x - y \leq c$,
$x, y \in X$,
$c \in \mathbb{Q}$ (or $c \in \mathbb{Z}$).

One special variable x_0 whose value is fixed to 0 is permitted;
this allows to express atoms like $x < 3$ in the form $x - x_0 < 3$.

Solving difference logic:

Let F be a conjunction in DL.
For simplicity: only non-strict inequalities.

Define a weighted graph G:

Vertices V: Variables in F.

Edges E: $x - y \leq c \rightsquigarrow$ edge (x, y) with weight c.

Theorem: F is unsatisfiable iff G has a negative cycle.

Can be checked in $O(|V| \cdot |E|)$ using the Bellman-Ford algorithm.

1.9 C-Arithmetic

In languages like C: Bounded integer arithmetic (modulo 2^n), in device drivers also combined with bitwise operations.

Bit-Blasting (encode everything as boolean circuits, use CDCL):

Naive encoding: possible, but often too inefficient.

If combined with over-/underapproximation techniques (Bryant, Kroening, et al.): successful.

1.10 Decision Procedures for Data Structures

There are decision procedures for, e. g.,

Arrays (read, write)

Lists (car, cdr, cons)

Sets or multisets with cardinalities

Bitvectors

Note: There are usually restrictions on quantifications. Unrestricted universal quantification can lead to undecidability.

Literature: Further Decision Procedures

Aaron R. Bradley, Zohar Manna: The Calculus of Computation. Springer, 2007.

Aaron R. Bradley, Zohar Manna, Henny B. Sipma: What's decidable about arrays? Verification, Model Checking, and Abstract Interpretation (VMCAI), LNCS 3855, pp. 427-442, Springer, 2006.

Randal E. Bryant, Daniel Kroening, Joël Ouaknine, Sanjit A. Seshia, Ofer Strichman, Bryan Brady: Deciding bit-vector arithmetic with abstraction. 13th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS'07), LNCS 4424, pp. 358–372, Springer, 2007.

George E. Collins: Quantifier Elimination for Real Closed Fields by Cylindrical Algebraic Decomposition. 2nd. GI Conf. Automata Theory and Formal Languages, LNCS 33, pp. 134–183, Springer, 1975.

D. C. Cooper: Theorem Proving in Arithmetic Without Multiplication. Machine Intelligence, vol. 7, pp. 91–99. American Elsevier, New York, 1972.

George B. Dantzig: Linear Programming and Extensions. Princeton Univ. Press, 1963.

L. V. Kantorovich: Mathematical Methods in the Organization and Planning of Production. Publication House of the Leningrad State University, 1939. Translated in Management Science, 6:366–422, 1960.

Narendra Karmarkar: A New Polynomial Time Algorithm for Linear Programming. Combinatorica, 4(4):373–395, 1984.

Daniel Kroening, Ofer Strichman: Decision Procedures – An Algorithmic Point of View. Springer, 2008.

Mojżesz Presburger: Über der Vollständigkeit eines gewissen Systems der Arithmetik ganzer Zahlen, in welchem die Addition als einzige Operation hervortritt. Comptes Rendus Premier Congrès des Mathématiciens des Pays Slaves, Warsaw, pp. 92-101, 1929.

William Pugh: The Omega Test: a fast and practical integer programming algorithm for dependence analysis. Comm. of the ACM, 35(8):102-114, 1992.

Stefan Ratschan: Approximate Quantified Constraint Solving by Cylindrical Box Decomposition. Reliable Computing, 8(1):21–42, 2002.

Alfred Tarski: A Decision Method for Elementary Algebra and Geometry. Univ. of California Press, Berkeley, 1951.

1.11 Combining Decision Procedures

Problem:

Let $\mathcal{T}_1$ and $\mathcal{T}_2$ be first-order theories over the signatures Σ_1 and Σ_2.

Assume that we have decision procedures for the satisfiability of existentially quantified formulas (or the validity of universally quantified formulas) w. r. t. $\mathcal{T}_1$ and $\mathcal{T}_2$.

Can we combine them to get a decision procedure for the satisfiability of existentially quantified formulas w. r. t. $\mathcal{T}_1 \cup \mathcal{T}_2$?

General assumption:

Σ_1 and Σ_2 are disjoint.

The only symbol shared by $\mathcal{T}_1$ and $\mathcal{T}_2$ is built-in equality.

We consider only conjunctions of literals.

For general formulas, convert to DNF first and consider each conjunction individually.

Abstraction

To be able to use the individual decision procedures, we have to transform the original formula in such a way that each atom contains only symbols of one of the signatures (plus variables).

This process is known as *variable abstraction* or *purification*.

We apply the following rule as long as possible:

$$\frac{\exists \vec{x}\,(F[t])}{\exists \vec{x}, y\,(F[y] \wedge t \approx y)}$$

if the top symbol of t belongs to Σ_i and t occurs in F directly below a Σ_j-symbol or in a (positive or negative) equation $s \approx t$ where the top symbol of s belongs to Σ_j $(i \neq j)$, and if y is a new variable.

It is easy to see that the original and the purified formula are equivalent.

Stable Infiniteness

Problem:

Even if the Σ_1-formula F_1 and the Σ_2-formula F_2 do not share any symbols (not even variables), and if F_1 is $\mathcal{T}_1$-satisfiable and F_2 is $\mathcal{T}_2$-satisfiable, we cannot conclude that $F_1 \wedge F_2$ is $(\mathcal{T}_1 \cup \mathcal{T}_2)$-satisfiable.

Example:

Consider

$\mathcal{T}_1 = \{\forall x, y, z\,(x \approx y \ \vee \ x \approx z \ \vee \ y \approx z)\}$

and

$\mathcal{T}_2 = \{\exists x, y, z\,(x \not\approx y \ \wedge \ x \not\approx z \ \wedge \ y \not\approx z)\}$.

All $\mathcal{T}_1$-models have at most two elements, and all $\mathcal{T}_2$-models have at least three elements.

Since $\mathcal{T}_1 \cup \mathcal{T}_2$ is contradictory, there are no $(\mathcal{T}_1 \cup \mathcal{T}_2)$-satisfiable formulas.

To ensure that $\mathcal{T}_1$-models and $\mathcal{T}_2$-models can be combined to $(\mathcal{T}_1 \cup \mathcal{T}_2)$-models, we require that both $\mathcal{T}_1$ and $\mathcal{T}_2$ are stably infinite.

A first-order theory $\mathcal{T}$ is called *stably infinite*, if every existentially quantified formula that has a $\mathcal{T}$-model has also a $\mathcal{T}$-model with a (countably) infinite universe.

Note: By the Löwenheim–Skolem theorem, "countable" is redundant here.

Shared Variables

Even if $\exists \vec{x}\, F_1$ is $\mathcal{T}_1$-satisfiable and $\exists \vec{x}\, F_2$ is $\mathcal{T}_2$-satisfiable, it can happen that $\exists \vec{x}\,(F_1 \wedge F_2)$ is not $(\mathcal{T}_1 \cup \mathcal{T}_2)$-satisfiable, for instance because the shared variables x and y must be equal in all $\mathcal{T}_1$-models of $\exists \vec{x}\, F_1$ and different in all $\mathcal{T}_2$-models of $\exists \vec{x}\, F_2$.

Example:

Consider

$F_1 = (x + (-y) \approx 0)$,

and

$F_2 = (f(x) \not\approx f(y))$

where $\mathcal{T}_1$ is linear rational arithmetic and $\mathcal{T}_2$ is EUF.

We must exchange information about shared variables to detect the contradiction.