1.3 Linear Rational Arithmetic

There are several ways to define *linear rational arithmetic.*

We need at least the following signature: $\Sigma = (\{0/0, 1/0, +/2\}, \{</2\})$ and the predefined binary predicate $\approx$.

The equational part of linear rational arithmetic is described by the theory of *divisible torsion-free abelian groups*:

$$\forall x, y, z\,(x + (y + z) \approx (x + (y + z))) \qquad \text{(associativity)}$$

$$\forall x, y\,(x + y \approx y + x) \qquad \text{(commutativity)}$$

$$\forall x\,(x + 0 \approx x) \qquad \text{(identity)}$$

$$\forall x\,\exists y\,(x + y \approx 0) \qquad \text{(inverse)}$$

$$\text{For all } n \geq 1\text{: } \forall x\,(\underbrace{x + \cdots + x}_{n \text{ times}} \approx 0 \rightarrow x \approx 0) \qquad \text{(torsion-freeness)}$$

$$\text{For all } n \geq 1\text{: } \forall x\,\exists y\,(\underbrace{y + \cdots + y}_{n \text{ times}} \approx x) \qquad \text{(divisibility)}$$

$$\neg\, 1 \approx 0 \qquad \text{(non-triviality)}$$

Note: Quantification over natural numbers is not part of our language. We really need infinitely many axioms for torsion-freeness and divisibility.

By adding the axioms of a compatible strict total ordering, we define *ordered divisible abelian groups:*

$$\forall x\,(\neg\, x < x) \qquad \text{(irreflexivity)}$$

$$\forall x, y, z\,(x < y \wedge y < z \rightarrow x < z) \qquad \text{(transitivity)}$$

$$\forall x, y\,(x < y \vee y < x \vee x \approx y) \qquad \text{(totality)}$$

$$\forall x, y, z\,(x < y \rightarrow x + z < y + z) \qquad \text{(compatibility)}$$

$$0 < 1 \qquad \text{(non-triviality)}$$

Note: The second non-triviality axiom renders the first one superfluous. Moreover, as soon as we add the axioms of compatible strict total orderings, torsion-freeness can be omitted. Every ordered divisible abelian group is obviously torsion-free.

In fact the converse holds: Every torsion-free abelian group can be ordered (F.-W. Levi 1913).

Examples: $\mathbb{Q}$, $\mathbb{R}$, $\mathbb{Q}^n$, $\mathbb{R}^n$, ...

The signature can be extended by further symbols:

$\leq/2$, $>/2$, $\geq/2$, $\not\approx/2$: defined using $<$ and $\approx$

$-/1$: Skolem function for inverse axiom

$-/2$: defined using $+/2$ and $-/1$

$\text{div}_n/1$: Skolem functions for divisibility axiom for all $n \geq 1$.

$\text{mult}_n/1$: defined by $\forall x\,(\text{mult}_n(x) \approx \underbrace{x + \cdots + x}_{n \text{ times}})$ for all $n \geq 1$.

$\text{mult}_q/1$: defined using mult_n, div_n, $-$ for all $q \in \mathbb{Q}$.

(We usually write $q \cdot t$ or qt instead of $\text{mult}_q(t)$.)

$q/0$ (for $q \in \mathbb{Q}$): defined by $q \approx q \cdot 1$.

Note: Every formula using the additional symbols is ODAG-equivalent to a formula over the base signature.

When $\cdot$ is considered as a binary operator, (ordered) divisible torsion-free abelian groups correspond to (ordered) rational vector spaces.

Fourier-Motzkin Quantifier Elimination

Linear rational arithmetic permits *quantifier elimination:* every formula $\exists x\, F$ or $\forall x\, F$ in linear rational arithmetic can be converted into an equivalent formula without the variable x.

The method was discovered in 1826 by J. Fourier and re-discovered by T. Motzkin in 1936.

Observation: Every literal over the variables $x, y_1, \dots, y_n$ can be converted into an ODAG-equivalent atom $x \sim t[\vec{y}]$ or $0 \sim t[\vec{y}]$, where $\sim \in \{<, >, \leq, \geq, \approx, \not\approx\}$ and $t[\vec{y}]$ has the form $\sum_i q_i \cdot y_i + q_0$.

In other words, we can either eliminate x completely or isolate in on one side of the atom.

Moreover, we can convert every $\not\approx$ atom into an ODAG-equivalent disjunction of two $<$ atoms.

We first consider existentially quantified conjunctions of atoms.

If the conjunction contains an equation $x \approx t[\vec{y}]$, we can eliminate the quantifier $\exists x$ by substitution:

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

is equivalent to

$$F\{x \mapsto t[\vec{y}]\}$$

If x occurs only in inequations, then

$$\begin{aligned}\exists x\, \Big(& \textstyle\bigwedge_i x < s_i(\vec{y}) \ \wedge\ \bigwedge_j x \le t_j(\vec{y}) \\ & \wedge\ \textstyle\bigwedge_k x > u_k(\vec{y}) \ \wedge\ \bigwedge_l x \ge v_l(\vec{y}) \ \wedge\ \bigwedge_m 0 \sim_m w_m(\vec{y})\Big)\end{aligned}$$

is equivalent to

$$\begin{aligned}& \textstyle\bigwedge_i \bigwedge_k s_i(\vec{y}) > u_k(\vec{y}) \ \wedge\ \bigwedge_j \bigwedge_k t_j(\vec{y}) > u_k(\vec{y}) \\ & \quad \wedge\ \textstyle\bigwedge_i \bigwedge_l s_i(\vec{y}) > v_l(\vec{y}) \ \wedge\ \bigwedge_j \bigwedge_l t_j(\vec{y}) \ge v_l(\vec{y}) \\ & \quad \wedge\ \textstyle\bigwedge_m 0 \sim_m w_m(\vec{y}) < 0\end{aligned}$$

Proof: ($\Rightarrow$) by transitivity;
($\Leftarrow$) take $\frac{1}{2}(\min\{s_i, t_j\} + \max\{u_k, v_l\})$ as a witness.

Extension to arbitrary formulas:

Transform into prenex formula;

if innermost quantifier is $\exists$: transform matrix into DNF and move $\exists$ into disjunction;

if innermost quantifier is $\forall$: replace $\forall x\, F$ by $\neg\exists x\, \neg F$, then eliminate $\exists$.

Consequence: every closed formula over the signature of ODAGs is ODAG-equivalent to either $\top$ or $\bot$.

Consequence: ODAGs are a *complete* theory, i. e., every closed formula over the signature of ODAGs is either valid or unsatisfiable w. r. t. ODAGs.

Consequence: every closed formula over the signature of ODAGs holds either in all ODAGs or in no ODAG.

ODAGs are indistinguishable by first-order formulas over the signature of ODAGs.

(These properties do not hold for extended signatures!)

Fourier-Motzkin: Complexity

One FM-step for $\exists$:

formula size grows quadratically, therefore $O(n^2)$ runtime.

m quantifiers $\exists \dots \exists$:

naive implementation needs $O(n^{2^m})$ runtime;
unknown whether optimized implementation with simply exponential runtime is possible.

m quantifiers $\exists\forall\exists\forall \dots \exists$:

CNF/DNF conversion (exponential!) required after each step;
therefore non-elementary runtime.

Loos-Weispfenning Quantifier Elimination

A more efficient way to eliminate quantifiers in linear rational arithmetic was developed by R. Loos and V. Weispfenning (1993).

The method is also known as "test point method" or "virtual substitution method".

For simplicity, we consider only one particular ODAG, namely $\mathbb{Q}$ (as we have seen above, the results are the same for all ODAGs).

Let $F(x, \vec{y})$ be a *positive* boolean combination of linear (in-)equations $x \sim_i s_i(\vec{y})$ and $0 \sim_j s'_j(\vec{y})$ with $\sim_i, \sim_j \in \{\approx, \not\approx, <, \leq, >, \geq\}$, that is, a formula built from linear (in-) equations, $\wedge$ and $\vee$ (but without $\neg$).

Goal: Find a *finite* set T of "test points" so that

$$\exists x\, F(x, \vec{y}) \quad \models\!\!\mid \quad \bigvee_{t \in T} F(x, \vec{y})\, \{x \mapsto t\}$$

In other words: We want to replace the infinite disjunction $\exists x$ by a finite disjunction.

If we keep the values of the variables $\vec{y}$ fixed, then we can consider F as a function $F : x \mapsto F(x, \vec{y})$ from $\mathbb{Q}$ to $\{0, 1\}$.

The value of each of the atoms $s_i(\vec{y}) \sim_i x$ changes only at $s_i(\vec{y})$, and the value of F can only change if the value of one of its atoms changes.

Let $\delta(\vec{y}) = \min\{\, |s_i(\vec{y}) - s_j(\vec{y})| \mid s_i(\vec{y}) \neq s_j(\vec{y})\, \}$

F is a piecewise constant function; more precisely, the set of all x with $F(x, \vec{y}) = 1$ is a finite union of intervals. (The union may be empty, the individual intervals may be finite or infinite and open or closed.)

Moreover, each of the intervals has either length 0 (i. e., it consists of one point), or its length is at least $\delta(\vec{y})$.

If the set of all x for which $F(x, \vec{y})$ is 1 is non-empty, then

(i) $F(x, \vec{y}) = 1$ for all $x \leq r(\vec{y})$ for some $r(\vec{y}) \in \mathbb{Q}$

(ii) or there is some point where the value of $F(x, \vec{y})$ switches from 0 to 1 when we traverse the real axis from $-\infty$ to $+\infty$.

We use this observation to construct a set of test points.

We start with some "sufficiently small" test point $r(\vec{y})$ to take care of case (i).

For case (ii), we observe that $F(x, \vec{y})$ can only switch from 0 to 1 if one of the atoms switches from 0 to 1. (We consider only *positive* boolean combinations of atoms and $\wedge$ and $\vee$ are monotonic w. r. t. truth values.)

$x \leq s_i(\vec{y})$ and $x < s_i(\vec{y})$ do not switch from 0 to 1 when x grows.

$x \geq s_i(\vec{y})$ and $x \approx s_i(\vec{y})$ switch from 0 to 1 at $s_i(\vec{y})$
$\Rightarrow s_i(\vec{y})$ is a test point.

$x > s_i(\vec{y})$ and $x \not\approx s_i(\vec{y})$ switch from 0 to 1 "right after" $s_i(\vec{y})$
$\Rightarrow s_i(\vec{y}) + \varepsilon$ (for some $0 < \varepsilon < \delta(\vec{y})$) is a test point.

If $r(\vec{y})$ is sufficiently small and $0 < \varepsilon < \delta(\vec{y})$, then

$$\begin{aligned} T := \{r(\vec{y})\} &\cup \{\, s_i(\vec{y}) \quad\;\; \mid \sim_i \in \{\geq, =\}\,\} \\ &\cup \{\, s_i(\vec{y}) + \varepsilon \mid \sim_i \in \{>, \neq\}\,\}. \end{aligned}$$

is a set of test points.

Problem:
We don't know how small $r(\vec{y})$ has to be for case (i), and we don't know $\delta(\vec{y})$ for case (ii).

Idea:
We consider the limits for $r \to -\infty$ and for $\varepsilon \searrow 0$, that is, we redefine

$$\begin{aligned} T := \{-\infty\} &\cup \{\, s_i(\vec{y}) \quad\;\; \mid \sim_i \in \{\geq, =\}\,\} \\ &\cup \{\, s_i(\vec{y}) + \varepsilon \mid \sim_i \in \{>, \neq\}\,\}. \end{aligned}$$

How can we eliminate the infinitesimals ∞ and ε when we substitute elements of T for x?

Virtual substitution:

$$\begin{aligned}
(x < s(\vec{y}))\,\{x \mapsto -\infty\} &:= \lim_{r\to-\infty}(r < s(\vec{y})) = \top\\
(x \leq s(\vec{y}))\,\{x \mapsto -\infty\} &:= \lim_{r\to-\infty}(r \leq s(\vec{y})) = \top\\
(x > s(\vec{y}))\,\{x \mapsto -\infty\} &:= \lim_{r\to-\infty}(r > s(\vec{y})) = \bot\\
(x \geq s(\vec{y}))\,\{x \mapsto -\infty\} &:= \lim_{r\to-\infty}(r \geq s(\vec{y})) = \bot\\
(x \approx s(\vec{y}))\,\{x \mapsto -\infty\} &:= \lim_{r\to-\infty}(r \approx s(\vec{y})) = \bot\\
(x \not\approx s(\vec{y}))\,\{x \mapsto -\infty\} &:= \lim_{r\to-\infty}(r \not\approx s(\vec{y})) = \top
\end{aligned}$$

$$\begin{aligned}
(x < s(\vec{y}))\,\{x \mapsto u+\varepsilon\} &:= \lim_{\varepsilon\searrow 0}(u+\varepsilon < s(\vec{y})) = (u < s(\vec{y}))\\
(x \leq s(\vec{y}))\,\{x \mapsto u+\varepsilon\} &:= \lim_{\varepsilon\searrow 0}(u+\varepsilon \leq s(\vec{y})) = (u < s(\vec{y}))\\
(x > s(\vec{y}))\,\{x \mapsto u+\varepsilon\} &:= \lim_{\varepsilon\searrow 0}(u+\varepsilon > s(\vec{y})) = (u \geq s(\vec{y}))\\
(x \geq s(\vec{y}))\,\{x \mapsto u+\varepsilon\} &:= \lim_{\varepsilon\searrow 0}(u+\varepsilon \geq s(\vec{y})) = (u \geq s(\vec{y}))\\
(x \approx s(\vec{y}))\,\{x \mapsto u+\varepsilon\} &:= \lim_{\varepsilon\searrow 0}(u+\varepsilon \approx s(\vec{y})) = \bot\\
(x \not\approx s(\vec{y}))\,\{x \mapsto u+\varepsilon\} &:= \lim_{\varepsilon\searrow 0}(u+\varepsilon \not\approx s(\vec{y})) = \top
\end{aligned}$$

We have traversed the real axis from $-\infty$ to $+\infty$. Alternatively, we can traverse it from $+\infty$ to $-\infty$. In this case, the test points are

$$\begin{aligned}
T' := \{+\infty\} &\cup \{\, s_i(\vec{y}) \mid \sim_i \in \{\leq, =\}\,\}\\
&\cup \{\, s_i(\vec{y}) - \varepsilon \mid \sim_i \in \{<, \neq\}\,\}.
\end{aligned}$$

Infinitesimals are eliminated in a similar way as before.

In practice: Compute both T and T' and take the smaller set.

For a universally quantified formulas $\forall x\, F$, we replace it by $\neg\exists x\, \neg F$, push inner negation downwards, and then continue as before.

Note that there is no CNF/DNF transformation required. Loos-Weispfenning quantifier elimination works on arbitrary positive formulas.

Loos-Weispfenning: Complexity

One LW-step for $\exists$ or $\forall$:

as the number of test points is at most half of the number of atoms, the formula size grows quadratically; therefore $O(n^2)$ runtime.

Multiple quantifiers of the same kind:

$$\begin{aligned}
& \exists x_2\, \exists x_1.\ F(x_1, x_2, \vec{y}) \\
\rightsquigarrow\quad & \exists x_2.\ \big(\textstyle\bigvee_{t_1 \in T_1} F(x_1, x_2, \vec{y})\,\{x_1 \mapsto t_1\}\big) \\
\rightsquigarrow\quad & \textstyle\bigvee_{t_1 \in T_1} (\exists x_2.\ F(x_1, x_2, \vec{y})\,\{x_1 \mapsto t_1\}) \\
\rightsquigarrow\quad & \textstyle\bigvee_{t_1 \in T_1} \bigvee_{t_2 \in T_2} (F(x_1, x_2, \vec{y})\,\{x_1 \mapsto t_1\}\,\{x_2 \mapsto t_2\})
\end{aligned}$$

m quantifiers $\exists \dots \exists$ or $\forall \dots \forall$:

formula size is multiplied by n in each step, therefore $O(n^{m+1})$ runtime.

m quantifiers $\exists\forall\exists\forall \dots \exists$:

doubly exponential runtime.

Note: The formula resulting from a LW-step is usually highly redundant; so an efficient implementation must make heavy use of simplification techniques.