(6.21) Prove or provide a counterexample: Consider the two inequations $x > y + 2$ and $2x < 4y + 3$ and the inequation $2y + 4 < 4y + 3$ obtained via elimination of x. Then all integer solutions of $2y+4 < 4y+3$ can be extended to integer solutions for x.

(6.22)$*$ Prove $a \mid dx + c$ iff $a \mid x + c \wedge d \mid x$. **ToDo:** This does not make sense

6.2.5 Branch and Bound for LIA

Let N be a conjunction of linear arithmetic constraints over $\{\leq, <, \not\approx, >, \geq\}$ with variables $x_1, \ldots, x_n$ where all coefficients are integers. Then LRA $\models \exists x_1, \ldots, x_n.N$ is typically efficiently decidable by the simplex algorithm, see also Section 6.2.2, whereas LIA $\models \exists x_1, \ldots, x_n.N$ is an NP-complete problem [74]. **ToDo:** other refs papa and poly simplex

Membership in NP results from the observation that the variables of N can be a priori bounded. Let a be the maximal absolute value of a coefficient in N then LIA $\models \exists x_1, \ldots, x_n.N$ iff there is an assignment β such that LIA$(\beta) \models N$ and for all variables x_i, $-n(|N|a)^{2|N|+1} \leq \beta(x_i) \leq n(|N|a)^{2|N|+1}$. Now guessing an assignment β and checking whether it satisfies N can be done in polynomial time, because the potential values for the x_i can be encoded logarithmically.

Lemma 6.2.14 (LIA Satisfiability is NP-Complete)**.** Let N be a conjunction of linear arithmetic constraints then LIA $\models \exists x_1, \ldots, x_n.N$ is NP-complete.

Proof. Membership in NP follows from the singly exponential bounds for each variable for a satisfying assignment.

On the other hand NP-hardness follows from a coding of 3-SAT, Section 2.14. Each propositional variable becomes an integer variable. The integer variables are a priori bounded to the values 0, 1. Clauses are translated into inequations. For example the clause set

$$N = \{\neg P \vee Q \vee R,\ P \vee \neg Q \vee R\}$$

is translated into the inequations

$$\begin{aligned} N_{\text{LIA}} = \quad & \{(1 - x_P) + x_Q + x_R \geq 1,\ x_P + (1 - x_Q) + x_R \geq 1, \\ & 0 \leq x_P, x_Q, x_R \leq 1\} \end{aligned}$$

where the latter inequation on x_P, x_Q, x_R is an abbreviation for the respective single inequations. It is not difficult to check that N is satisfiable iff N_{LIA} has a solution. □

So a naive procedure for checking LIA $\models \exists x_1, \ldots, x_n.N$ could simply test all values for the x_i. Such a procedure can already be significantly improved by the following observation: if LIA $\models \exists x_1, \ldots, x_n.N$ then LRA $\models \exists x_1, \ldots, x_n.N$. As a consequence, if there is no rational solution, i.e., LRA $\not\models \exists x_1, \ldots, x_n.N$,

then there is also no integer solution, i.e., LIA $\not\models \exists x_1, \ldots, x_n.N$. Furthermore, if by accident a rational solution β, LRA(β) $\models N$ is also an integer solution, it can be reused. If β assigns to some variable x_i a non-integer value d, then N has an integer solution iff $N \wedge x_i \geq \lceil d \rceil$ or $N \wedge x_i \leq \lfloor d \rfloor$. Both problems rule out the solution $\beta(x_i) = d$, so they can again be tested by relaxation with respect to LRA. Putting all this together results in the classical branch and bound approach to LIA solving.

The simple LIA branch and bound calculus is very similar to DPLL, Section 2.8. A LIABB problem state is a pair $(M; N)$ where M a sequence of partly annotated simple bounds $x_i \leq d$, $d \in \mathbb{Z}$, and N is a set of inequations, vars$(N) = \{x_1, \ldots, x_n\}$. Let a be the maximal absolute value of a coefficient in N, $c = n(|N|a)^{2|N|+1}$, then the following LIABB states can be distinguished:

$(B; N)$	is the start state for some set N of inequations, where $B = -c \leq x_1, x_1 \leq c, \ldots, -c \leq x_n, x_n \leq c$.
$(M; N)$	is a final state, if there is a unique β, LIA(β) $\models M \wedge N$
$(M; N)$	is a final state, if there is no β, LIA(β) $\models N$

Given a state (M, N), a simple bound $x \circ d$, $d \in \mathbb{Z}$, is called *undefined* in M, if there exists a valuation β, LIA(β) $\models M$ and LIA(β) $\not\models x \circ d$. The rules Propagate, Decide, and Backtrack constitute the LIABB calculus.

Propagate $(M; N) \Rightarrow_{\text{LIABB}} (M, x \circ d; N)$

provided there is a valuation β, LRA(β) $\models M \wedge N$, LIA $\models \forall x_1, \ldots, x_n.[(M \wedge N) \rightarrow x \circ d]$, $d \in \mathbb{Z}$, and $x \circ d$ is undefined in M

Decide $(M; N) \Rightarrow_{\text{LIABB}} (M, x \circ e^d; N)$

provided $x \circ e$ is undefined in M, LRA(β) $\models M \wedge N$, $\beta(x) = d$ and either ($\circ = \leq$ and $e = \lfloor d \rfloor$) or ($\circ = \geq$ and $e = \lfloor d \rfloor + 1$)

Backtrack $(M_1, x \circ_1 e_1^d, M_2; N) \Rightarrow_{\text{LIABB}} (M_1, x \circ_2 e_2; N)$

provided there is no valuation β, LRA(β) $\models (M_1 \wedge x \circ_1 e_1 \wedge M_2 \wedge N)$ and there is no $y \circ' e'^{d'}$ in M_2 and if $\circ_1 = \leq$, then $\circ_2 = \geq$ and $e_2 = \lfloor d \rfloor + 1$; if $\circ_1 = \geq$, then $\circ_2 = \leq$ and $e_2 = \lfloor d \rfloor$

The notions of a trail, a propagated literal, or a decided literal carry over from DPLL. Similar to DPLL, the rules Propagate and Decide can only be applied finitely often.

Note that rule Decide can be restricted to decisions on rationals that are not integeters, i.e., the condition $d \notin \mathbb{Z}$ can be added without changing any properties of the calculus. In case β is actually an integer solution the restriction definitely improves the performance. In case β contains non-integer assignemtns it is a heuristic.

Lemma 6.2.15 (LIABB Propagate and Decide)**.** Let $(B, N) \Rightarrow^*_{\text{LIABB}} (M, N)$ be a LIABB derivation. Then from (M, N) there only finitely many applications of Propagate and Decide possible.

Proof. For each variable x_i there is an integer upper and lower bound in M. So there are only finitely many different β with $\text{LIA}(\beta) \models M$. Both Propagate and Decide rule out at least one assignment β. If $M \wedge N$ becomes LRA unsolvable, none of the rules is applicable anymore. If M has no LIA solution, none the rules is applicable anymore. □

Theorem 6.2.16 (LIABB Terminates)**.** Any derivation $(B, N) \Rightarrow^*_{\text{LIABB}} \ldots$ is finite.

Similar to DPLL, Propagate is not needed for soundness or completeness. In contrast to DPLL, it is not a priori wise to give Propagate priority over Decide. Consider the following set of linear inequations

$$N = \{2x_1 - x_2 \leq 0,\ x_2 - x_1 \leq 0\},$$

where $n = |N| = a = 2$, hence $c = 2048$. So the start state is $(B; N)$, where $B = -2048 \leq x_1 \leq 2048, -2048 \leq x_2 \leq 2048$. Obviously, the two inequations have many integer solutions, e.g., $x_1 = x_2 = 0$, $x_1 = x_2 = -1, \ldots$. In particular, for all (integer) solutions, $x_1 \leq 0$. So Propagate can be applied 2047 times on x_1 without making any progress in finding a solution in the following way:

$$\begin{array}{ll} \text{(B;N)} & \Rightarrow^{\text{Propagate}}_{\text{LIABB}} (B, x_1 \leq 2047; N) \\ & \Rightarrow^{\text{Propagate}}_{\text{LIABB}} (B, x_1 \leq 2047, x_1 \leq 2046; N) \\ & \Rightarrow^{\text{Propagate}}_{\text{LIABB}} \ldots \end{array}$$

On the other hand an application of Decide with $\beta(x_1) = \beta(x_2) = -\frac{1}{2}$ immediately rules out the above sequence

$$\text{(B;N)} \quad \Rightarrow^{\text{Decide}}_{\text{LIABB}} (B, x_1 \leq -1^{-\frac{1}{2}}; N)$$

and further a Propagate step combining $x_1 \leq -1$ with $x_2 - x_1 \leq 0$

$$\Rightarrow^{\text{Propagate}}_{\text{LIABB}} (B, x_1 \leq -1^{-\frac{1}{2}}, x_2 \leq -1; N)$$

Next Decide is applied twice with $\beta(x_1) = \beta(x_2) = -1$ yielding

$$\begin{array}{l} \Rightarrow^{\text{Decide}}_{\text{LIABB}} (B, x_1 \leq -1^{-\frac{1}{2}}, x_2 \leq -1, x_1 \geq -1^{-1}; N) \\ \Rightarrow^{\text{Decide}}_{\text{LIABB}} (B, x_1 \leq -1^{-\frac{1}{2}}, x_2 \leq -1, x_1 \geq -1^{-1}, x_2 \geq -1^{-1}; N) \end{array}$$

that is in solved form an no more rule is applicable, because M has as its only LIA solution $\beta(x_1) = \beta(x_2) = -1$ that is also a solution for N.

The second example consists of three inequations

$$N = \{3x_2 \leq x_1,\ 6x_2 \geq 1 + x_1,\ 2x_2 \leq 3 - x_1\},$$

where $n = 2$, $|N| = 3$, and $a = 6$, hence $c = 1224440064$. So the start state is $(B; N)$, where $B = -1224440064 \leq x_1 \leq 1224440064, -1224440064 \leq x_2 \leq 1224440064$. An application of Decide with $\beta(x_1) = 2$, $\beta(x_2) = \frac{1}{2}$ and subsequently Propagate yields

$$\begin{array}{ll} \text{(B;N)} & \Rightarrow^{\text{Decide}}_{\text{LIABB}} (B, x_1 \leq 2^2; N) \\ & \Rightarrow^{\text{Propagate}}_{\text{LIABB}} (B, x_1 \leq 2^2, x_2 \leq 1; N) \end{array}$$

and one more application of Decide with the above β

$$\Rightarrow^{\text{Decide}}_{\text{LIABB}} (B, x_1 \leq 2^2, x_2 \leq 1, x_2 \leq 0^{\frac{1}{2}}; N).$$

The above state does not have an LRA solution resulting in the sequence

$$\begin{array}{l} \Rightarrow^{\text{Backtrack}}_{\text{LIABB}} (B, x_1 \leq 2^2, x_2 \leq 1, x_2 \geq 1; N). \\ \Rightarrow^{\text{Backtrack}}_{\text{LIABB}} (B, x_1 \geq 3; N). \\ \Rightarrow^{\text{Propagate}}_{\text{LIABB}} (B, x_1 \geq 3, x_2 \geq 1; N). \end{array}$$

where the Propagate application is the result of combining $x_1 \geq 3$ and $6x_2 \geq 1 + x_1$. No rule is applicable to this state anymore because it does not have a LRA solution. Hence, N does not have a solution in LIA.

Already, the above two simple examples show that the a priori upper bounds are not very useful in practice, because they grow too rapidly. Therefore, many implementations of the LIA branch and bound approach don't consider them at all.

Note that checking LIA $\models \exists x_1, \ldots, x_n.M$ can be done in linear time with respect to M. It just means to compare the largest lower bound with the smallest upper bound for each variable, plus considering some potential inequations. Checking definedness for a particular inequation is also of linear time complexity. However, checking the propagation condition LIA $\models \forall x_1, \ldots, x_n.[(M \wedge N) \rightarrow x \circ d]$ is difficult, in contrast to checking propagation in DPLL. This condition needs to be instantiated by affordable techniques. For example, when applied to the above examples, I restricted Propagate to the propagation of simple bounds out of M with respect to single inequalities from N.

ToDo: Some nice pictures for the meaning of the inequations in 2D