Constraint Solving	WS 2022/2023	LVA 703304
EXAM 1		February 3, 2023

$\boxed{1}$ (a) *calculation + explanation*

$$\varphi := \exists x.\exists y.\ (3x + 2y < 2 \wedge -x + 5y > 1)$$

gather y on one side

$$\equiv \exists x.\exists y.\ (2y < 2 - 3x \wedge 5y > 1 + x)$$

multiply to get $10y$ everywhere

$$\equiv \exists x.\exists y.\ (10y < 10 - 15x \wedge 10y > 2 + 2x)$$

switch to y'

$$\equiv \exists x.\exists y'.\ (y' < 10 - 15x \wedge y' > 2 + 2x \wedge 10 | y')$$

elimination of y': $B = \{2 + 2x\}$, $\delta = 10$, left infinite projection simplifies to $\bot$

$$\equiv \exists x. \bigvee_{j=1}^{10} (2 + 2x + j < 10 - 15x \wedge 2 + 2x + j > 2 + 2x \wedge 10 | 2 + 2x + j)$$

further simplification (not required)

$$\equiv \exists x. \bigvee_{j \in \{2,4,6,8,10\}} (17x + j < 8 \wedge 10 | 2 + 2x + j)$$

(b) *calculation + explanation*

The formula can be written as

$$3x + 2y < 2$$
$$-x + 5y > 1$$

and the elimination of strict inequalities yields

$$3x + 2y \leq 2 - \delta$$
$$-x + 5y \geq 1 + \delta$$

So we get the following initial tableau and bounds and an initial assignment where everything becomes 0:

tableau	x	y	bounds	assignment	x	y	s	t
s	3	2	$s \leq 2 - \delta$		0	0	0	0
t	-1	5	$t \geq 1 + \delta$					

There is a violation for t. Both x and y are suitable, but Bland's rule will select x. Pivoting of t and x results in:

tableau	t	y	bounds	assignment	x	y	s	t
s	-3	17	$s \leq 2 - \delta$		0	0	0	0
x	-1	5	$t \geq 1 + \delta$					

Updating the assignment $t := 1 + \delta$ results in:

tableau	t	y	bounds	assignment	x	y	s	t
s	-3	17	$s \leq 2 - \delta$		$-1 - \delta$	0	$-3 - 3\delta$	$1 + \delta$
x	-1	5	$t \geq 1 + \delta$					

Since no bound is violated, no further iterations of the main loop are required.

$\boxed{2}$ (a) *algorithm bb-mixed*$(\mathcal{V}_\mathbb{Z}, \mathcal{V}_\mathbb{Q}, \varphi)$

- invoke simplex on φ
- if the output is unsat, return unsat
- otherwise, let v be the solution of simplex
- if $v(x) \in \mathbb{Z}$ for all $x \in \mathcal{V}_\mathbb{Z}$ then return solution v
- otherwise, choose some $x \in \mathcal{V}_\mathbb{Z}$ such that $c = v(x) \notin \mathbb{Z}$
- if *bb-mixed*$(\mathcal{V}_\mathbb{Z}, \mathcal{V}_\mathbb{Q}, \varphi \cup \{x \leq \lfloor c \rfloor\})$ returns a solution v' then return v'
- otherwise, return *bb-mixed*$(\mathcal{V}_\mathbb{Z}, \mathcal{V}_\mathbb{Q}, \varphi \cup \{x \geq \lceil c \rceil\})$

(b.i) *upper bound*

The small solution is obtain by starting in $hull(H)$ and adding at most n vectors $\vec{c}_i$ of C with coefficients $\lambda_i < 1$. Hence, an upper bound is $(n+1) \cdot u$.

(b.ii) *answer and explanation*

The same construction is still possible:
one starts with an arbitrary solution $\vec{x} \in \mathbb{Z}^k \times \mathbb{Q}^{n-k}$, i.e., $\vec{x} = \vec{v} + \sum_{i=1}^{n} \lambda_i \vec{c}_i$ where $\vec{v} \in hull(H)$ and each $\vec{c}_i \in C$ and $\lambda_i \geq 0$.
If all λ_i are below 1, then the small vector is obtained. Otherwise, there is some $\lambda_i \geq 1$. Then one can change λ_i to $\lambda_i - 1$ and gets the solution $\vec{x} - \vec{c}_i$ that is contained in $\mathbb{Z}^k \times \mathbb{Q}^{n-k}$, since C consists of integral vectors. Repeating this adaptation of the solution will finally result in the desired small solution.

$\boxed{3}$ (a) *implication graph*

$$\begin{tikzcd}
1 \arrow[r, "(b)"] \arrow[d, "(d)"] & \neg 2 \arrow[r, "(c)"] \arrow[d, "(e)"] & 3 & \\
4 \arrow[r, "(e)"] \arrow[dr, "(g)"] & 5 \arrow[dr, "(i)"] \arrow[drr, "(j)"] & & \\
6 \arrow[r, "(g)"] \arrow[d, "(h)"] & \neg 7 \arrow[r, "(i)"] & \neg 9 \arrow[r, "(j)"'] \arrow[d, "(k)"] & \neg 10 \arrow[d, "(l)"] \\
8 \arrow[urr, "(i)"] & & 11 \arrow[r, "(l)"] & \otimes
\end{tikzcd}$$

(b) *first unique implication point + backjump clause*

There are two unique implication points, the first one is $\neg 9$, and the other is the decision literal 6. The backjump clause of the first UIP is (m): $\neg 5 \vee 9$.

(c) *next configuration*

$$\overset{d}{1}\ \underset{(b)}{\neg 2}\ \underset{(c)}{3}\ \underset{(d)}{4}\ \underset{(e)}{5}\ \underset{(m)}{9}$$

$\boxed{4}$ (a) *chosen theory + formula*

The obvious choice is LIA where each puzzle variable x_i is directly represented by an integer variable. The formula is

$$\underbrace{c = 1x_1 + \ldots + kx_k}_{\varphi_{eq}} \wedge \underbrace{\bigwedge_{i=1}^{k} (0 \leq x_i \wedge x_i \leq 7)}_{\varphi_{LIA}}$$

(b) *chosen theory + formula*

The next obvious choice is LRA where each puzzle variable x_i is represented by a rational variable. To avoid that rational values are taken, we replace the formula φ_{LIA} by φ_{LRA}.

$$\underbrace{\bigwedge_{i=1}^{k} (x_i = 0 \vee x_i = 1 \vee \ldots \vee x_i = 7)}_{\varphi_{LRA}}$$

(c) *chosen theory + formula*

Another obvious choice is bit-vector arithmetic where each puzzle variable x_i is represented by a bit-vector variable. In bit-vector arithmetic we have to specify which kind of encoding we want to use in comparisons (we want unsigned), so we replace the formula φ_{LIA} by φ_{BV}.

$$\underbrace{\bigwedge_{i=1}^{k} x_i \leq_u 7}_{\varphi_{BV}}$$

We also have to take care that no overflows can occur within φ_{eq}. Since the summation can take a value of at most $1 \cdot 7 + 2 \cdot 7 + \ldots + k \cdot 7 = 7 \cdot k \cdot (k+1)/2 \leq 4(k+1)^2$, it suffices to consider a bit-width of at least $2 \cdot \lceil \log_2(4(k+1)) \rceil$. Using only 3 bits (to represent numbers between 0 and 7 which covers digits 0 to 7) is not sufficient.

(d) *algorithm*

Let $\varphi(p)$ be the encoding for a single puzzle constraint, i.e., choose any of the encodings above.
We first check satisfiability of the puzzle by invoking a SMT solver on $\psi := \bigwedge_{i=1}^{n} \varphi(p_i)$.
If the ψ is not satisfiable then return "puzzle has no unique solution."
Otherwise, extract the solution, i.e., we get a concrete assignment $x_1 = v_1, \ldots, x_m = v_m$ for digits $v_1, \ldots, v_m$.
Next check satisfiability of $\chi := \psi \wedge \neg(\bigwedge_{i=1}^{m} x_i = v_i)$.
If χ is satisfiable return "puzzle has no unique solution." and otherwise return the unique solution $x_1 = v_1, \ldots, x_m = v_m$.

$\boxed{5}$

Question	Yes	No
The Nelson–Oppen algorithm is a quantifier elimination algorithm for LRA.	☐	☑
The congruence closure algorithm is used in the context of EUF.	☑	☐
Difference logic constraints can be solved in polynomial time for both $\mathbb{Z}$ and $\mathbb{Q}$.	☑	☐
LRA, LIA and EUF are convex theories.	☐	☑
Array Logic is decidable if both the index theory (including quantifiers) is decidable and the element theory is decidable.	☐	☑