Logic and Reasoning in Computer Science SS 2025 — Exercise Sheet 3

Instructions for project submissions: You are requested to solve at least one project problem and submit your project solutions by **June 6, 2025**. Project problems are explicitly stated as *Project Problem 3.1* and *Project Problem 3.2*.

Upload the files for your solved project(s) in a zip archive. Use the following naming and formatting for the project you solved:

- `proj1.smt2` : Your encoding of Project Problem 3.1 in `SMT-LIB2` syntax.
- `proj1.pdf` : A short interpretation/analysis of your input and the output of the SMT solver Z3 when running it on `proj1.smt2`.
- `proj2.py` : Your solution to Project Problem 3.2. Please follow the instructions given in Project Problem 3.2.
- `proj2.pdf` : A short document explaining your solution. Please follow the instructions given in Project Problem 3.2.

Please **follow the file naming convention**, and submit your project solutions as instructed, by June 6, 2025. **Please double-check your file names, only correctly named files are graded.** Please also ensure that you use the correct file extensions, e.g. only `.smt2` and not `.smt2.txt`, as your operating system might hide file extensions.

Exercise Problem 3.1. Consider the following $\mathcal{T}_E$-formula:

$$x = y \ \lor \ y = z \ \lor \ x = z,$$

where x, y, z are variables. *Describe the class* of all $\mathcal{T}_E$- interpretations that make this formula valid.

Solution. A $\mathcal{T}_E$-interpretation makes this formula valid if and only if the domain of the sort of x contains at most two elements.

Exercise Problem 3.2. Consider the following $\mathcal{T}_E$-formula:

$$x = y \ \rightarrow \ (x = a \ \lor \ x = b),$$

where x, y are variables and a, b are constants. *Describe the class* of $\mathcal{T}_E$ interpretations that make this formula valid.

Solution:
Note, that the given formula is equivalent to $x = a \lor x = b$.
The class of interpretations that make this formula valid consists of all interpretations I that interpret the sort of x, a, b wih the same domain and this domain contains either:

(i) only one element, in this case $I(a) = I(b)$.

or

(ii) two elements, in this case $I(a) \neq I(b)$.

Exercise Problem 3.3. For each formula below, *decide whether it is satisfiable or not* by relying on the decision procedure of $\mathcal{T}_E$. Provide also the intermediate steps of the decision procedure.

(a) $p(x) \wedge f(f(x)) = x \wedge f(f(f(x))) = x \wedge \neg p(f(x))$

Solution.
We first transform the formula into the below given equivalent formula without predicates other than $=$. To this end, we introduce the fresh constant t and fresh function f_p, and write:

$$f_p(x) = t \wedge f(f(x)) = x \wedge f(f(f(x))) = x \wedge f_p(f(x)) \neq t$$

From the subterm set of this formula, the initial partition is:

$$\{ \{t\}, \{x\}, \{f(x)\}, \{f(f(x))\}, \{f(f(f(x)))\}, \{f_p(x)\}, \{f_p(f(x))\} \}.$$

According to the literal $f_p(x) = t$, merge $\{f_p(x)\}$ and $\{t\}$ to form the partition:

$$\{ \{t, f_p(x)\}, \{x\}, \{f(x)\}, \{f(f(x))\}, \{f(f(f(x)))\}, \{f_p(f(x))\} \}.$$

From the second literal $f(f(x)) = x$, merge $\{f(f(x))\}$ and $\{x\}$. From the congruence class $\{f(f(x)), x\}$, deduce the following congruence propagation:

$$f(f(x))\, R\, x \implies f(f(f(x)))\, R\, f(x).$$

Thus the final partition of this iteration is:

$$\{ \{t, f_p(x)\}, \{x, f(f(x))\}, \{f(x), f(f(f(x)))\}, \{f_p(f(x))\} \}.$$

From the third literal $f(f(f(x))) = x$, merge $\{f(x), f(f(f(x)))\}$ and $\{x, f(f(x))\}$ to form the congruence class

$$\{ \{t, f_p(x)\}, \{x, f(x), f(f(x)), f(f(f(x)))\}, \{f_p(f(x))\} \}.$$

Propagating the congruence:

$$x\, R\, f(x) \implies f_p(x)\, R\, f_p(f(x))$$

yields the partition

$$\{ \{t, f_p(x), f_p(f(x))\}, \{x, f(x), f(f(x)), f(f(f(x)))\} \},$$

which represents the congruence closure of the subterm set.
We thus have $f_p(f(x))\, R\, t$, while our input formula asserts that $f_p(f(x)) \neq t$. Hence, the considered formula is $\mathcal{T}_E$-unsatisfiable.

(b) $(a = b \vee f(a) = b) \wedge f(f(a)) \neq b \wedge f(b) = b$

Solution.
We first name every theory atom by a propositional symbol:

$$\begin{array}{ll} p_1: & a = b \\ p_2: & f(a) = b \\ p_3 & f(f(a)) = b \\ p4: & f(b) = b \end{array}$$

Our input formula is thus represented by the following set S of propositional clauses:

$$\begin{array}{l} p_1 \ \vee p_2 \\ \neg p_3 \\ p4 \end{array}$$

By splitting first on p_1, DPLL returns satisfiability of S and reports the set of literals

$$p_1, \neg p_3, p_4$$

satisfying S.
Satisfiability of the set T of theory literals

$$\begin{array}{l} a = b \\ f(f(a)) \neq b \\ f(b) = b \end{array}$$

corresponding to $p_1, \neg p_3, p_4$ is next checked, using the congruence closure algorithm of $\mathcal{T}_E$. We obtain that T is $\mathcal{T}_E$-unsatisfiable.
We add $\neg p_1 \vee p_3 \vee \neg p_4$ to the initial set S of propositional clauses, and obtain the set S' of propositional clauses:

$$\begin{array}{l} p_1 \ \vee p_2 \\ \neg p_3 \\ p4 \\ \neg p_1 \ \vee \ p_3 \ \vee \ \neg p_4 \end{array}$$

DPLL reports the literals

$$\neg p_1, p2, \neg p_3, p_4$$

satisfying S'.
Satisfiability of the set T' of theory literals

$$\begin{array}{l} a \neq b \\ f(a) = b \\ f(f(a)) \neq b \\ f(b) = b \end{array}$$

corresponding to $\neg p_1, p2, \neg p_3, p_4$ is next checked. We derive that T' is $\mathcal{T}_E$-unsatisfiable.
We add $p_1 \vee \neg p_2 \vee p_3 \vee \neg p_4$ to the set S' of propositional clauses, and obtain the set S'' of propositional clauses:

$$\begin{array}{l} p_1 \ \vee p_2 \\ \neg p_3 \\ p4 \\ \neg p_1 \ \vee \ p_3 \ \vee \ \neg p_4 \\ p_1 \ \vee \ \neg p_2 \vee p_3 \ \vee \ \neg p_4 \end{array}$$

DPLL reports the unsatisfiability of S'', which concludes the $\mathcal{T}_E$-unsatisfiability of our input formula.

Exercise Problem 3.4. *Find a model* of the following formula in the $\mathcal{T}_E \cup \mathcal{T}_A \cup \mathcal{T}_{\mathbb{Z}}$:

$$\mathit{read}(A, f(x,y)) = x + y \ \wedge \ \mathit{read}(A, f(y,x)) \neq y + x.$$

Solution.
We consider the domain $\mathbb{Z}$ of integer numbers, and define a model I of the given set of two formulas. We will interpret arrays as functions on $\mathbb{Z} \to \mathbb{Z}$ and define the function $read^I$ as the function application:

$$read(B, z)^I := B(z^I) \quad \text{for all } B, z$$

Obviously, this interpretation satisfies all axioms of the theory of arrays.
Now, define:

$$\begin{array}{ll} x^I := 1 & \\ y^I := 0 & \\ f^I(u^I, v^I) := u^I & \text{for all } u^I, v^I \text{ in } \mathbb{Z} \\ A^I[u^I] := A(u^I) := u^I & \text{for all } u^I \text{ in } \mathbb{Z} \end{array}$$

Then,

$$\begin{array}{l} f(x, y)^I := 1, \\ f(y, x)^I := 0, \\ (x + y)^I := 1, \\ (y + x)^I := 1, \\ read(A, f(x, y))^I := 1, \\ read(A, f(y, x))^I := 0, \end{array}$$

and thus

$$(read(A, f(x, y))^I, (x + y)^I) \in = \quad \text{and} \quad (read(A, f(y, x))^I, (y + x)^I) \notin =$$

We thus conclude that I is a model of the given set of formulas.

Exercise Problem 3.5. *Show unsatisfiability* of the following formula in the $\mathcal{T}_E \cup \mathcal{T}_A \cup \mathcal{T}_\mathbb{Z}$:

$$f(b) = b \ \wedge \ f(f(b)) = 1 \ \wedge \ f(1) + c = d \ \wedge \ read(write(A, c, b), d - 1) \neq f(b).$$

Use reasoning in combination of theories, and provide a level of details.

Solution.
We show unsatisfiability of the following set of formulas:

$$\begin{array}{l} f(b) = b \\ f(f(b)) = 1 \\ f(1) + c = d \\ read(write(A, c, b), d - 1) \neq f(b) \end{array}$$

We introduce the following new variables[1]:

$$\begin{array}{lll} c_1 & \text{to denote} & f(1) \\ c_2 & \text{to denote} & d - 1 \\ c_3 & \text{to denote} & f(b) \end{array}$$

Then we should satisfy the following set of literals in three theories:

Theory of equality and uninterpreted functions $\mathcal{T}_E$:

[1]For simplicity and efficiency, in the sequel we allow sharing of constants among theories.

$$\begin{aligned} &f(b) = b \\ &f(f(b)) = 1 \\ &f(1) = c_1 \\ &f(b) = c_3 \end{aligned}$$

Theory of linear integer arithmetic $\mathcal{T}_{\mathbb{Z}}$:

$$\begin{aligned} &c_1 + c = d \\ &c_2 = d - 1 \end{aligned}$$

Theory of arrays $\mathcal{T}_A$

$$read(write(A, c, b), c2) \neq c_3$$

Congruence closure applied to the theory of equality and uninterpreted functions gives a single congruence class:

$$\{f(b), b, c_3, 1, f(1), c_1\},$$

which results in the following shared equations

$$b = c_3 = 1 = c_1.$$

These equations and

$$\begin{aligned} &c_1 + c = d \\ &c_2 = d - 1 \end{aligned}$$

yield the consequence $c = c_2$, which will be added by the arithmetic reasoner to the set of shared equations. The set of shared equations becomes

$$\begin{aligned} &b = c_3 = 1 = c_1 \\ &c = c_2 \end{aligned}$$

Then the array reasoner derives unsatisfiability.

Exercise Problem 3.6. Consider the following two formulas

(F1)

$$\begin{aligned} &b - 1 = c + 3 \ \wedge \ f(b) \neq b + 1 \ \wedge \ read(A, f(c+4)) = c + 3 \ \wedge \\ &\big(read(A, f(b)) = b + 2 \vee read(write(A, b+1, f(c+3)), f(c+4)) = c + 5\big) \end{aligned}$$

(F2)

$$\begin{aligned} &b = c + 3 \ \wedge \ f(b) = c - 1 \ \wedge \ read(A, f(c+3)) = b + 1 \ \wedge \\ &\big(read(A, f(b)) = c \vee read(write(A, b, f(c+3)), f(b)) = c + 4\big) \end{aligned}$$

where b, c are constants, f is a unary function symbol, A is an array constant, *read*, *write* are interpreted in the array theory, and $+$, $-$, -1, 1, 2, 3, 4, 5 are interpreted in the standard way over the integers.

For each formula (F1) and (F2) above, *use the Nelson-Oppen decision procedure* in conjunction with

DPLL-based reasoning in the combination of the theories of arrays, uninterpreted functions, and linear integer arithmetic. Use the decision procedures for the theory of arrays and the theory of uninterpreted functions, and use simple mathematical reasoning for deriving new equalities among the constants in the theory of linear integer arithmetic. If the formula is satisfiable, give an interpretation that satisfies the formula.

Solution.
The input formulas are expressed in the combined theories of $\mathcal{T}_E$, $\mathcal{T}_A$, and $\mathcal{T}_Z$. Both formulas contain a clause consisting of two literals. Therefore, to decide the satisfiability of the input formulas, we use DPLL($\mathcal{T}$), where $\mathcal{T}$ is $\mathcal{T}_E \cup \mathcal{T}_A \cup \mathcal{T}_Z$.

Solution for (F1):
We first name every atom by a propositional variable:

$$\begin{array}{ll} p_1: & b-1=c+3 \\ p_2: & f(b)=b+1 \\ p_3: & read(A,f(c+4))=c+3 \\ p_4: & read(A,f(b))=b+2 \\ p_5: & read(write(A,b+1,f(c+3)),f(c+4))=c+5 \end{array}$$

Our input formula is thus represented by the following set S of propositional clauses:

$$\{p_1, \quad \neg p_2, \quad p_3, \quad p_4 \vee p_5\}$$

We apply DPLL on S:

(DPLL-1) DPLL reports satisfiability and returns a set of literals satisfying S: $\{p_1, \neg p_2, p_3, p_4, p_5\}$. The corresponding set of $\mathcal{T}$-theory literals is:

$$\begin{array}{l} b-1=c+3 \\ f(b)\neq b+1 \\ read(A,f(c+4))=c+3 \\ read(A,f(b))=b+2 \\ read(write(A,b+1,f(c+3)),f(c+4))=c+5 \end{array}$$

We introduce the following variables:

a_1	to denote	$b+1$	a_5	to denote	$f(b)$
a_2	to denote	$c+4$	a_6	to denote	$b+2$
a_3	to denote	$f(a_2)$	a_7	to denote	$f(a_4)$
a_4	to denote	$c+3$	a_8	to denote	$c+5$

By separating reasoning in the various theories, we should satisfy the following set of literals:

– in $\mathcal{T}_E$:

$$\begin{array}{l} f(b)\neq a_1 \\ f(a_2)=a_3 \\ f(b)=a_5 \\ f(a_4)=a_7 \end{array}$$

– in $\mathcal{T}_A$:

$$\begin{array}{l} read(A,a_3)=a_4 \\ read(A,a_5)=a_6 \\ read(write(A,a_1,a_7),a_3)=a_8 \end{array}$$

– in $\mathcal{T}_Z$:

$$\begin{aligned}
&b - 1 = c + 3\\
&a_1 = b + 1\\
&a_2 = c + 4\\
&a_4 = c + 3\\
&a_6 = b + 2\\
&a_8 = c + 5
\end{aligned}$$

Using simple arithmetical reasoning of $\mathcal{T}_Z$, from $b - 1 = c + 3$ and $a_2 = c + 4$ we derive the shared equality:

– shared equality among $\mathcal{T}_A$, $\mathcal{T}_E$ and $\mathcal{T}_Z$:

$$a_2 = b$$

Next, we use the decision procedure of the theory $\mathcal{T}_E$ over the above $\mathcal{T}_E$-theory literals and shared equalities, that is over:

$$\begin{aligned}
&f(b) \neq a_1\\
&f(a_2) = a_3\\
&f(b) = a_5\\
&f(a_4) = a_7\\
&a_2 = b
\end{aligned}$$

and derive $a_3 = a_5$ as a new shared equality. That is,

– shared equalities among $\mathcal{T}_A$, $\mathcal{T}_E$ and $\mathcal{T}_Z$:

$$\begin{aligned}
&a_2 = b\\
&a_3 = a_5
\end{aligned}$$

We then use the decision procedure of the theory $\mathcal{T}_A$ over the $\mathcal{T}_A$-theory literals and shared equalities and derive $a_4 = a_6$, yielding

– shared equalities among $\mathcal{T}_A$, $\mathcal{T}_E$ and $\mathcal{T}_Z$:

$$\begin{aligned}
&a_2 = b\\
&a_3 = a_5\\
&a_4 = a_6
\end{aligned}$$

We use simple reasoning over $\mathcal{T}_Z$, $\mathcal{T}_A$-literals and shared equalities, and derive a contradiction (as $b - 1$ cannot equal $b + 2$). Hence, the literals in $\mathcal{T}_Z$ are UNSAT, implying that the *(DPLL-1)* branch is unsatisfiable. DPLL is supplied with $S \cup \{\neg p_1 \vee p_2 \vee \neg p_3 \vee \neg p_4 \vee \neg p_5\}$.

(DPLL-2) The next propositional model is $\{p_1, \neg p_2, p_3, \neg p_4, p_5\}$. Similarly to the *(DPLL-1)* branch, we derive a contradiction. Next, DPLL is run with the following propositional clause set:

$$S \cup \left\{\begin{array}{c} \neg p_1 \vee p_2 \vee \neg p_3 \vee \neg p_4 \vee \neg p_5,\\ \neg p_1 \vee p_2 \vee \neg p_3 \vee p_4 \vee \neg p_5 \end{array}\right\}$$

(DPLL-3) The next propositional model is $\{p_1, \neg p_2, p_3, p_4, \neg p_5\}$. Similarly to the *(DPLL-1)* branch, we derive a contradiction. DPLL is given the following propositional clause set which is unsatisfiable:

$$S \cup \left\{\begin{array}{c} \neg p_1 \vee p_2 \vee \neg p_3 \vee \neg p_4 \vee \neg p_5,\\ \neg p_1 \vee p_2 \vee \neg p_3 \vee p_4 \vee \neg p_5,\\ \neg p_1 \vee p_2 \vee \neg p_3 \vee \neg p_4 \vee p_5 \end{array}\right\}$$

DPLL returns UNSAT, therefore the formula **F1** is unsatisfiable.

Solution for (F2):
We first name every atom by a propositional variable:

$$\begin{array}{ll} p_1: & b = c+3 \\ p_2: & f(b) = c-1 \\ p_3: & read(A, f(c+3)) = b+1 \\ p_4: & read(A, f(b)) = c \\ p_5: & read(write(A, b, f(c+3)), f(b)) = c+4 \end{array}$$

Our input formula is thus represented by the following set S of propositional clauses:

$$\{p_1, \quad p_2, \quad p_3, \quad p_4 \vee p_5\}$$

We apply DPLL on S:

(DPLL-1) DPLL reports satisfiability and returns a set of literals satisfying S: $\{p_1, p_2, p_3, p_4, p_5\}$. The corresponding set of $\mathcal{T}$-theory literals is:

$$\begin{array}{l} b = c+3 \\ f(b) = c-1 \\ read(A, f(c+3)) = b+1 \\ read(A, f(b)) = c \\ read(write(A, b, f(c+3)), f(b)) = c+4 \end{array}$$

We introduce the following variables:

a_1	to denote	$c+3$	a_4	to denote	$b+1$
a_2	to denote	$c-1$	a_5	to denote	$f(b)$
a_3	to denote	$f(a_1)$	a_6	to denote	$c+4$

By separating reasoning in the various theories, we should satisfy the following set of literals:

– in $\mathcal{T}_E$:

$$\begin{array}{l} f(b) = a_2 \\ f(a_1) = a_3 \\ f(b) = a_5 \end{array}$$

– in $\mathcal{T}_A$:

$$\begin{array}{l} read(A, a_3) = a_4 \\ read(A, a_5) = c \\ read(write(A, b, a_3), a_5) = a_6 \end{array}$$

– in $\mathcal{T}_Z$:

$$\begin{array}{l} b = c+3 \\ a_1 = c+3 \\ a_2 = c-1 \\ a_4 = b+1 \\ a_6 = c+4 \end{array}$$

First we derive the shared equalities in $\mathcal{T}_Z$:

– shared equalities among $\mathcal{T}_A$, $\mathcal{T}_E$ and $\mathcal{T}_Z$:

$$\begin{array}{l} a_1 = b \\ a_4 = a_6 \end{array}$$

Next, we use the decision procedure of the theory $\mathcal{T}_E$ over the above $\mathcal{T}_E$-theory literals and shared equalities, that is over:

$$\begin{aligned} &f(b) = a_2 \\ &f(a_1) = a_3 \\ &f(b) = a_5 \\ &a_1 = b \\ &a_4 = a_6 \end{aligned}$$

and derive the following shared equalities:

– shared equalities among $\mathcal{T}_A$, $\mathcal{T}_E$ and $\mathcal{T}_Z$ (represented by equivalence classes from now on for simplicity):

$$[a_1, b], \quad [a_4, a_6], \quad [a_2, a_3, a_5]$$

Since neither a_3 nor a_5 is present in the literals of $\mathcal{T}_Z$, what remains is considering the theory $\mathcal{T}_A$ over the $\mathcal{T}_A$-theory literals and shared equalities. Since the $\mathcal{T}_A$ literals contain $write$ symbols, we branch on the read-over-write operations. First, we get the shared equalities $b = a_5$ and $a_3 = a_6$ by $read(write(A, b, a_3), a_5) = a_6$:

– shared equalities among $\mathcal{T}_A$, $\mathcal{T}_E$ and $\mathcal{T}_Z$:

$$[a_1, b], \quad [a_4, a_6], \quad [a_2, a_3, a_5], \quad [b, a_5], \quad [a_3, a_6]$$

Reasoning over $\mathcal{T}_E$ results in the new shared equalities:

– shared equalities among $\mathcal{T}_A$, $\mathcal{T}_E$ and $\mathcal{T}_Z$:

$$[b, a_1, a_2, a_3, a_4, a_5, a_6]$$

Next, reasoning over $\mathcal{T}_Z$ results in a contradiction as we obtain $c = c + 4$. We backtrack to the branching point in $\mathcal{T}_A$ to get $b \neq a_5$ and $read(A, a_5) = a_6$. Moreover, since the current $\mathcal{T}_A$-theory literals do not contain $write$ operations, we can replace write over A with the fresh uninterpreted function f_A and get the following $\mathcal{T}_E$ equivalence classes:

$$\begin{gathered} [a_1, b], \quad [a_4, a_6], \quad [a_2, a_3, a_5], \quad [f_A(a_3), a_4], \quad [f_A(a_5), c], \\ [f_A(a_5), a_6], \quad [f(b), a_2], \quad [f(a_1), a_3], \quad [f(b), a_5] \end{gathered}$$

From this via $\mathcal{T}_E$ reasoning, we get the equivalence classes:

$$[a_1, b], \quad [a_2, a_3, a_5, f(a_1), f(b)], \quad [a_4, a_6, f_A(a_3), f_A(a_5), c]$$

Now, what remains is to check the shared equalities $a_4 = c$ and $a_6 = c$ within $\mathcal{T}_Z$. This results in a contradiction as we get $c = c + 4$.

Hence, since both branches inside $\mathcal{T}_A$ result in a contradiction, $\mathcal{T}_A$ returns unsatisfiable and the current propositional model is refuted. Hence, the propositional clauses $S \cup \{\neg p_1 \vee \neg p_2 \vee \neg p_3 \vee \neg p_4 \vee \neg p_5\}$ are given to DPLL, excluding the current model.

(DPLL-2) The new propositional model by DPLL is $\{p_1, p_2, p_3, p_4, \neg p_5\}$ which results in a similar contradiction. (*Remark:* this contradiction originates from the literals p_1, p_3 and p_4 which imply $c = c + 4$. However, our simple algorithm cannot eliminate the conjunction of these three contradicting literals, only full models.)

(DPLL-3) The next propositional model by DPLL is $\{p_1, p_2, p_3, \neg p_4, p_5\}$, which results in the following sets of literals:

– in $\mathcal{T}_E$:

$$\begin{aligned} &f(b) = a_2 \\ &f(a_1) = a_3 \\ &f(b) = a_5 \end{aligned}$$

– in $\mathcal{T}_A$:

$$\begin{aligned} &read(A, a_3) = a_4 \\ &read(A, a_5) \neq c \\ &read(write(A, b, a_3), a_5) = a_6 \end{aligned}$$

– in $\mathcal{T}_Z$:

$$\begin{aligned} &b = c + 3 \\ &a_1 = c + 3 \\ &a_2 = c - 1 \\ &a_4 = b + 1 \\ &a_6 = c + 4 \end{aligned}$$

Similarly as for the first propositional model in *(DPLL-1)*, we get a contradiction from applying the read-over-write axiom with the assumption $b = a_5$ in $\mathcal{T}_A$. Hence, in $\mathcal{T}_A$ we assume $b \neq a_5$, replace $write$ operations on A with f_A to get the following set of $\mathcal{T}_E$ equivalence classes and the disequality $f_A(a_5) \neq c$:

$$[a_1, b], \quad [c], \quad [a_4, a_6], \quad [a_2, a_3, a_5], \quad [f_A(a_3), a_4],$$
$$[f_A(a_5), a_6], \quad [f(b), a_2], \quad [f(a_1), a_3], \quad [f(b), a_5]$$

We get the following equivalence classes in $\mathcal{T}_E$:

$$[a_1, b], \quad [c], \quad [f(b), f(a_1), a_2, a_3, a_5], \quad [f_A(a_3), f_A(a_5), a_4, a_6]$$

The shared equalities result in no new contradictions/shared equalities from the other theories. Hence, we can build a satisfying interpretation for this propositional model.

We construct model I of F_2, by satisfying the above equalities, as follows. Let α be a sort and let us consider the domain $\mathbb{Z}$ of integers. We consider array indexes and elements to be of sort α. We interpret the sort α as the domain $\mathbb{Z}$, that is, $\alpha^I := \mathbb{Z}$. We next interpret arrays as functions on $\mathbb{Z} \to \mathbb{Z}$, that is, for an array A, we have $A^I : \mathbb{Z} \to \mathbb{Z}$. We set $f : \alpha \to \alpha$ and interpret $f^I : \mathbb{Z} \to \mathbb{Z}$. We consider b, c to be of sort α and define:

$$b^I = 3, \quad c^I = 0, \quad f^I(u) = -1, \text{ for all } u \in \mathbb{Z}, \qquad A^I(u) = 4, \text{ for all } u \in \mathbb{Z}$$

Exercise Problem 3.7. Consider two axiomatizable theories $\mathcal{T}_1$ and $\mathcal{T}_2$ such that $\Sigma_{\mathcal{T}_1} = \Sigma_{\mathcal{T}_2} = \Sigma$ and $A_{\mathcal{T}_2} \subseteq A_{\mathcal{T}_1}$. Recall that $\Sigma_{\mathcal{T}_i}$ and $A_{\mathcal{T}_i}$ denote, respectively, a signature and set of axioms defining $\mathcal{T}_i$, for $i = 1, 2$. Let F be a formula over signature Σ.

(a) If F is valid in $\mathcal{T}_1$, it is also valid in $\mathcal{T}_2$? *Prove your answer or give a counterexample.*

Solution:

No.

Consider standard first-order logic, with signature Σ. Let F be a first-order formula that is satisfiable, but not valid. (For example, the atomic formula $p(x)$, with $p \in \Sigma$ and variable x).

Take $A_{\mathcal{T}_2} = \emptyset$ and $A_{\mathcal{T}_1} = F$. Then, F is valid in $\mathcal{T}_1$. Yet, F is not valid in $\mathcal{T}_2$ as F is a satisfiable but not a valid first-order formula.

(b) If F is valid in $\mathcal{T}_2$, it is also valid in $\mathcal{T}_1$? *Prove your answer or give a counterexample.*

Solution:

Yes.

Assume F is valid in $\mathcal{T}_2$. That is, for any interpretation J of Σ with $J \models A_{\mathcal{T}_2}$, we have $J \models F$.

Take *an arbitrary* $\mathcal{T}_1$*-interpretation* I of Σ, that is an arbitrary interpretation I of Σ with such that $I \models A_{\mathcal{T}_1}$. As $A_{\mathcal{T}_2} \subseteq A_{\mathcal{T}_1}$, we have $I \models A_{\mathcal{T}_2}$. From the assumption that F is valid in $\mathcal{T}_2$, we then *obtain* $I \models F$. Thus, F is a valid in $\mathcal{T}_1$.

Exercise Problem 3.8. Let Σ be an arbitrary signature. Let T be the theory of equality for Σ such that for every T-satisfiable formula $F \in T$ and for every finite T-interpretation I with $I \models F$, there exists a finite T-interpretation J such that $J \models F$ and $|J| = |I| + 1$. Here, $|J|$ and $|I|$ respectively denote the cardinalities of the domains of J and I.

Is T a stably infinite theory? Provide a sufficiently detailed explanation for your answer.

Solution:
Let F be a T-satisfiable formula and let I be a finite T-interpretation of F. By assumption, F also has a finite T-interpretation J such that $|J|$=$|I| + 1$. Morever, using the assumption and by induction on $|I|$, we have that F has a model of any finite cardinality $\mathcal{C} > |I|$. By compactness of first-order logic, F then has a countably infinite model, and hence T is stably infinite.

Note(advanced):* As F has a countably infinite model, we can derive that F has a model of any infinite cardinality $\mathcal{C} > |I|$ (thanks to the Löwenheim-Skolem Theorem).

Exercise Problem 3.9. Show that the theory $\mathcal{T}_A$ of arrays is not convex. Provide a counterexample to convexity.

Solution:
Let F the $\mathcal{T}_A$-formula $read(write(A, x, v), y) = v$. By the axioms of $\mathcal{T}_A$, we have

$$F \rightarrow (x = y \vee read(A, y) = v).$$

However, it does not hold that $F \rightarrow (x = y)$, nor that $F \rightarrow (read(A, y) = v)$.

Project Problem 3.1. Consider the following C-like imperative program p:

```
**assume**$(x > 0)$;
**assume**$(y > 0)$;
$A[x] := y$;
**if** $x = y$ **then**
   $A[y] := x$;
**else**
   $A[x] := y$;
**end if**
```

We assume x, y to be variables over mathematical integers, and A an (unbounded) array variable of mathematical integer values. The program lines annotated with **assume** impose additional requirements on the inputs x, y.

Let F be the formula

$$2 \cdot read(A, x) = 2 \cdot y \ \wedge \ 2 \cdot read(A, x) > 0$$

interpreted over the theory of $\mathcal{T}_E \cup \mathcal{T}_A \cup \mathcal{T}_{\mathbb{Z}}$.

Your task is to prove that, under the imposed assumptions upon x, y, the formula F is valid after the execution of p. Doing so, you are asked to encode the problem as a *single* `SMT-LIB2` file[2] consisting of three parts:

(1) The declarations of symbols used in your encoding.

(2) The encoding of the semantics of program p and the input constraints.

(3) Encoding that F needs to be valid, triggering the SMT solver to prove the validity of F. Use F as it is and do not do simplifications on F.

Use `SMT-LIB2` comments in your encoding to mark each of these three parts.

Encoding instructions. Use the following template for your encoding; the template is also available on TUWEL. Please note to leave the three comments in square brackets untouched for grading purposes and add your lines accordingly. Changing these three squared comments or misplacing your commands with respect to these three parts can result in your submission being graded with 0 points.

```
; [PART1-DECLARATIONS]
(declare-const ...)
...


; [PART2-PROGRAM]
(assert ...)
...

; [PART3-CONJECTURE]

(assert ...)
(check-sat)
```

Part (1) of your encoding should only contain the syntax declarations of your problem, such as new constant/function symbols declared via `(declare-const ...)` or `(declare-fun ...)`.

[2] standard at `https://smt-lib.org/papers/smt-lib-reference-v2.7-r2025-02-05.pdf`

Part (2) of your encoding should contain `SMT-LIB2` assertions capturing the semantics of program p and the input requirements. Each formula in this part should thus be encoded via `(assert ...)`.

Part (3) contains `SMT-LIB2` assertions that encode that the statement F should be valid. Apart from these assertions, you need a `(check-sat)` command.

Encoding submission. Please save your encoding as `proj1.smt2` and submit it as such. An `SMT-LIB2` template file is provided to you on TUWEL. Make sure that your submissions keep the comments of the template in square brackets untouched, as they are required for automating submission grading. Feel free to add further comments in your submission.

Project deliverables. If you solve Project Problem 3.1, you are asked to submit two files.

- `proj1.smt2`: An encoding of the problem as an input to the Z3 SMT solver in the `SMT-LIB2` format. Respect the encoding instructions, especially the three parts of your encoding as described above.
- `proj1.pdf`: A description and analysis of your encoding. This document should answer at the following points:
  - What are the constraints in your `SMT-LIB2` encoding expressing?
  - How do you conclude validity of the formula F with respect to p from your encoding, using Z3?

  The PDF has to be a single A4 page with a reasonable font size and no embedded code.

Running Z3. You can use the SMT solver Z3 by

- building or downloading Z3 binaries from `https://github.com/Z3Prover/z3`
- or running Z3 in your browser at `https://microsoft.github.io/z3guide/`

Further details on using the Z3 SMT solver are given in Lecture 19.

Project Problem 3.2. Robot Path Planning
You are a systems architect in a robotics lab. Your team is developing a new "AI" for a robot that operates autonomously on a square grid of size $n \times n$ (with $n > 1$). The robot moves differently, depending on the cell on the grid it currently occupies. Each cell is programmed with a fixed movement command of the form: "If the robot is currently at this cell, move it to cell $\langle x, y\rangle$." The positions on the grid are numbered from 0 to $n-1$ on both axis. As computer scientists, we obviously consider the left top corner as position $\langle 0, 0\rangle$ and the bottom right one as $\langle n-1, n-1\rangle$. That means, for e.g., $n = 3$ we have the positions:

$\langle 0,0\rangle$	$\langle 1,0\rangle$	$\langle 2,0\rangle$
$\langle 0,1\rangle$	$\langle 1,1\rangle$	$\langle 2,1\rangle$
$\langle 0,2\rangle$	$\langle 1,2\rangle$	$\langle 2,2\rangle$

Your task is to find such a command assignment for each cell in the $n \times n$ grid using the API of the SMT solver Z3 and given some constraints. Different movement commands $\langle x, y\rangle$ are assigned individually to each of the n^2 cells of the grid. The movement commands satisfy the following constraints to ensure safety and coordination within the system:

1. Horizontal Diversity: no two cells being on the same row require the robot to go to a cell with the same y-value.
2. Vertical Diversity: no two cells being on the same column require the robot to go to a cell with the same x-value.
3. No Zero Movement: the robot is required to move – thus, no cell can require the robot to go to the cell's own position.
4. Stay Within the Observed Area: a cell cannot require the robot to go to a position outside of the grid's boundaries – recall that these boundaries are $[0, n-1]$ for both x and y.
5. No Collision: no two cells may target the same cell as the destination. Equivalently: Every cell is targeted by exactly one other cell.
6. Predefined Movements: some of the movement instructions on the grid are already fixed: For some cells, only the x or y value is given, for some both are set, and for some, neither is given. This predefined movement information is given to you.

Note that those constraints do not necessarily enforce that there is a single cyclic path through the grid. Your task is to implement three functions within the provided `proj2.py` template: `encode`, `decode`, and `exclude_model`. See `proj2.py` for a detailed description of what to implement. In a nutshell:

- `encode`: adds the constraints for (1–6) to the Z3 solver
- `decode`: extracts the movement assignments from Z3's model
- `exclude_model`: adds constraints to the Z3 solver to exclude (only) the given movement assignment as a solution

Resources to be used. You can download the project archive for Project Problem 3.2 from TUWEL. This archive contains:

- A Jupyter notebook `robots_navigator.ipynb` containing code to load and run the solver on input files using your code from `proj2.py`. There is no need to edit nor submit this file. This notebook will call the Python file `proj2.py` you are supposed to submit.

- A python file `proj2.py` containing the template for your project. Your task is to complete the functions `encode`, `decode`, and `exclude_model`. Submit *this file* only. Additional/external files will not be evaluated.
- A Python file `robots_utils.py` containing useful functions to check the correctness of your encoding and a pretty printer. There is no need to edit nor submit this file.
- A folder `examples` containing example inputs and their complete solution sets. You are invited to create more to test your solution more thoroughly. If your encoding fails on any of the benchmarks provided by us, you will receive 0 points on this project. Again, you are neither required to submit our, nor your own additional test files.

Libraries to be used. This project requires coding in Python. At least Python version 3.8[3] is required to run the provided notebook.

- **Jupyter notebooks**[4]. Jupyter notebooks are a very useful tool to display intermediate results of a Python script. VS Code[5] has support for notebooks, but you can also run them on the browser.
- **Z3**. Z3 is an SMT solver developed mainly at Microsoft Research and contains an official Python wrapper for most of the C API functions. You can simply install the Python interface via Python's `pip`. The wrapper installed via pip ships with the required version of the Z3 solver.

Project deliverables. If you solve Project Problem 3.2, you are asked to deliver two files:

1. `proj2.py`: Your code for encoding and solving the problem, as instructed above.
2. `proj2.pdf`: A description of your encoding and how you solved the problem. The PDF has to be a single A4 page with a reasonable font size and no embedded code.

Submission instructions.

1. You are **not** allowed to add additional imports.
2. You are **not** allowed to call external files, even if you submit them as well.
3. You are **not** allowed to rename the `encode`, `decode`, and `exclude_model` functions or add/remove parameters from them.
4. Your encoding has to **work at least on all** test files provided to you to get any points.

For transparency: To evaluate your submission, we will ask your code in `proj2.py` to output all solutions and compare them to the correct set of solutions: This means, we call the function `encode` once and repeatedly call Z3 followed by `decode` and `exclude_model` to enumerate all solutions. A similar process is also implemented in `robots_navigator.ipynb`. Note, the order in which the models are generated is irrelevant. Make sure that your functions do not have side-effects that change their behaviour when being called multiple times and do not output the same solution (with respect to the command assignment grid) multiple times – your `exclude_model` function is likely containing a bug in that case. Also, none of the benchmarks we will use has more than 1000 different solutions. Further, the solver should not need more than 5 seconds to find an assignment.

[3] `https://www.python.org/downloads/`
[4] `https://jupyter.org/`
[5] `https://code.visualstudio.com/docs/datascience/jupyter-notebooks`