CSC2108HS: Automated Reasoning with Machine Learning 4 Oct 2024

Assignment I, v1

Instructor: Xujie Si
TAs: Sissi Jiang, Zhaoyu Li

Due: 25 Oct 2024, 11:59 PM

Watch the course website for updates to the assignment with clarifications: v1 - initial.

This is an individual assignment.

What to submit: Submit a report, `a1.pdf`, explaining your solutions with up to 2 pages for each problem, and your solutions {`P1a,P1b`}`_result.txt` (i.e., outputs of given tests) for questions **1A, 1B**, `Palindrome.lean` for Part **2**, and `Tactics.lean`, `Solver.lean` for Part **3**, along with a zip containing other relevant code for your solutions or execution logs (e.g., encoded constraints, outputs from the solver).
Warning: Please make sure to strictly follow the file naming given above; files with incorrect names might not be taken into consideration during grading.

Academic integrity is fundamental, and we will do a plagiarism check. Please keep your report, code, and solutions confidential and not share them with anyone.

Part 1: Practicing SAT/SMT Encoding (30 points)

SAT/SMT solvers are convenient tools to solve various practical problems. In this problem, you will use SAT/SMT to encode and solve two tasks using your favorite SAT/SMT solver.

(1A, 15 points)

A scientist studying aliens approached you recently and told you that he accidentally saw two pieces of cipher-texts on his colleague's desk. He is pretty sure that each word in the secret message should correspond to an English word (i.e., a word in the English 10k dictionary [1]). He would like you to help him decode the secret texts because he is not sure if these are some messages from an alien world.

Let's assume that the alien simply maps each alphanumeric character to a unique case-insensitive letter in the English alphabet. Please try your best to help this scientist with your knowledge as well as engineering skills about SAT/SMT solving.

Here is the first message:

```
K4 4Y74AP C3V Q4E7 KLEN4E 7C 5C3, V4ELP4A7E CT IAC7N4V KCVUP. IT74V V4IPLAO
7N4 TCUUCKLAO G4EEIO4, 5C3 ENC3UP NIW4 I QIELS 3AP4VE7IAPLAO CT SLWLULMI7LCA
CA 4IV7N. 7NVC3ON NIVP KCVJ IAP SV4I7LWL75, H4CHU4 NIW4 SV4I74P I OV4I7 SLWLULMI7LCA
TLUU4P KL7N GIA5 PLTT4V4A7 S3U73V4E.  K4 NIW4 IUEC Q4O3A 7C 3AP4VE7IAP 7N4 UIKE
OCW4VALAO 7N4 AI73VIU KCVUP IAP 7N4 P4W4UCHG4A7 CT N3GIA ECSL47L4E. K4 WIU34
4W4V57NLAO K4 NIW4 ISNL4W4P. Q37 C3V KCVUP LE AC7 H4VT4S7. NI74 4YLE7E, IE PC4E KIV.
Q4SI3E4 CT SCATULS7E Q47K44A 7N4 TCVS4E CT HVCP3S7LCA IAP 7N4 V4UI7LCAE CT
HVCP3S7LCA, K4IU7N LE AC7 TILVU5 PLE7VLQ374P, IAP GIA5 H4CHU4 ULW4 LA HCW4V75
IAP E3TT4VLAO. N3GIA ECSL47L4E IV4 KCVJLAO NIVP 7C V4ECUW4 7N4 PLTTLS3U7L4E IAP
HVCQU4GE 7N45 TIS4, KCVJLAO NIVP 7C SV4I74 I Q4774V T373V4 TCV 4IV7N SLWLULMI7LCA.
7N4 SC3A7V5 7NI7 E4A7 7NLE G4EEIO4 LE 4AOIO4P LA 7NLE 4TTCV7. K4 IV4 P4PLSI74P 7C
Q3LUPLAO IA LP4IU ECSL475, KN4V4 7N4 UIQCV IAP WIU34 CT 4W4V5 G4GQ4V CT 7N4 N3GIA
```

[1] `https://github.com/first20hours/google-10000-english/blob/master/google-10000-english.txt`

```
VIS4 IV4 T3UU5 V4EH4S74P, KN4V4 7N4 GI74VLIU IAP EHLVL73IU A44PE CT 4W4V5CA4 IV4
T3UU5 G47, EC 7NI7 SLWLULMI7LCA CA 4IV7N GI5 Q4SCG4 GCV4 H4VT4S7. K4 NCH4 7C GIJ4
SCAA4S7LCAE KL7N C7N4V IPWIAS4P ECSL47L4E LA 7N4 3ALW4VE4. K4 UCCJ TCVKIVP 7C
KCVJLAO 7CO47N4V KL7N 5C3 7C Q3LUP I Q4774V ULT4 LA 7NLE WIE7 3ALW4VE4.
```

Here is the second message:

```
7NLE KCVUP NIE V4S4LW4P 5C3V G4EEIO4. L IG I H4VECA KNC Q4UL4W4 LA H4IS4 LA 7NLE KCVUP.
L7 LE 7N4 U3SJ CT 5C3V SLWLULMI7LCA 7NI7 L IG 7N4 TLVE7 7C V4S4LW4 5C3V G4EEIO4.
L IG KIVALAO 5C3, PC AC7 IAEK4V. PC AC7 IAEK4V. PC AC7 IAEK4V.
7N4V4 IV4 GLUULCAE CT E7IVE LA 5C3V PLV4S7LCA.
IE UCAO IE 5C3 PC AC7 IAEK4V, 7NLE KCVUP KLUU AC7 Q4 IQU4 7C TLO3V4 C37 KN4V4
5C3V G4EEIO4 SIG4 TVCG. Q37 LT 5C3 PC IAEK4V, 7N4 EC3VS4 KLUU Q4 UCSI74P VLON7 IKI5.
H4CHU4 TVCG 7NLE KCVUP KLUU 7IVO47 5C3V HUIA47 IAP 7IJ4 CW4V 5C3V KCVUP.
PC AC7 IAEK4V. PC AC7 IAEK4V. PC AC7 IAEK4V.
```

What is the plain text for these two messages? You cannot simply answer an alien who wrote this! Please describe how you solved it, and provide the mapping key (if there exists one) and SAT encoding in a language of your choice in your report.

(1B, 15 points)

A palindrome number is a number that remains the same when its digits are reversed. For example, $21_{10} = 10101_2$ is a palindrome number in base 2. Given a number n, please try to find two palindrome numbers in base 2 such that $n = a + b$. Please encode this problem using SAT and call a SAT solver to solve it. You may assume a and b have the same length in base 2. Here is an example:

```
Input:
44
Output:
17
27
```

where $17_{10} = 10001_2$ and $27_{10} = 11011_2$.

We provide 4 test cases as follows and you are encouraged to create and share new test cases on Ed (but of course, not your solutions)

```
test-1:  426
test-2:  130686
test-3:  7887885102
test-4:  30174351474473570888398004459264386876869045194067045787317983758480550292850
726010027516741429727891945486366493691712785577992990514585975914146892632401418066 81
094244905551964089286644802238610346527693067000795579380080347934761405906051456566 92
273313364807613593082036221225110204142908645394181908936406
```

Please describe how you encode this problem in your report and submit outputs a and b (in base 10) for each test. *Hint: consider encoding an adder.*

Part 2: Practicing Interactive Theorem Proving (40 points)

In the second part, you will prove some interesting properties of Palindrome (e.g., its repetitions are Palindrome as well, and vice versa) in a formal manner. Theorems have been formalized in `Lean 4`, while proofs are left as `sorry`. Please fill those `sorry` place-holders with your proofs. See `Palindrome.lean` for more details and hints.

Note that since your solutions are machine-checkable, you can easily validate whether your answers (proofs) are correct or not before your submission.

Part 3: Implementing an Automation Tactic (30 points)

In the third part, you will implement your own automation tactic to solve the Maze puzzle introduced in the Week 4 lecture.

Hint: there are two possible approaches to building an automation tactic. Either developing a new tactic completely in Lean 4 or designing an external decision procedure (aka solver) and invoking it through a tactic wrapper. An implementation sketch has been provided for each approach (see `Maze/Tactics.lean` for details).

Lean 4 Cheatsheet

In the following tables, *name* always refers to a name already known to Lean while *new_name* is a new name provided by the user; *expr* means an expression, for example the name of an object in the context, an arithmetic expression that is a function of such objects, a hypothesis in the context, or a lemma applied to any of these; *proposition* is an expression of the type `Prop` (e.g. `0 < x`).

Logical symbol	Appears in goal	Appears in hypothesis
$\forall$ (for all)	`intro` *new_name*	`apply` *expr* or `specialize` *name expr*
$\rightarrow$ (implies)	`intro` *new_name*	`apply` *expr* or `specialize` *name expr*
$\leftrightarrow$ (if and only if)	`constructor`	`rw [`*expr*`]` or `rw [←` *expr*`]`
$\wedge$ (and)	`constructor`	`obtain` $\langle$*new_name* , *new_name*$\rangle$ `:=` *expr*

$\vee$ (or)	`left` or `right`	`cases` *expr* `with`
		`| inl` *new_name* `=> ...`
		`| inr` *new_name* `=> ...`

In the left-hand column of the following table, the parts in parentheses are optional. The effect of these parts is also in parentheses in the right-hand column.

Tactic	Effect
`apply` *expr*	prove the goal with *expr* and further prove relevant assumptions
`simp` (`at` *hyp*)	simplify the goal (or the hypothesis *hyp*) using standard equalities
`have` *new_name* : *proposition*	introduce a name *new_name* asserting that *proposition* is true; at the same time, create and focus a goal for *proposition*
`unfold` *name* (`at` *hyp*)	unfold the definition *name* in the goal (or in the hypothesis *hyp*)
`rw [` (`←`) *expr* `]` (`at` *hyp*)	in the goal (or in the hypothesis *hyp*), replace (all occurrences of) the left-hand side (or the right-hand side, if `←` is present) of the equality or equivalence *expr* by its other side
`rw [` *expr* `,` *expr* `,` *expr* `]` (`at` *hyp*)	do more rewrites in the given order (any number of `←` possible)
`cases n`	perform case analysis on `n`
`induction n`	perform induction analysis on `n`
`assumption`	prove the goal with an existing assumption
`contradiction`	prove the goal by a False (or equivalent) assumption