SAT and SMT Solving | WS 2022 | LVA 703147

Exercises 5 | November 18, 2022

$\boxed{1}$ Consider the theory of equality EQ, and the theory of equality with uninterpreted functions EUF, where $\mathcal{F} = \{\mathsf{f}/1, \mathsf{g}/2, \mathsf{a}/0, \mathsf{b}/0\}$ and $\mathcal{P} = \{=/2, \mathsf{P}/1\}$ (so there are a unary function symbol f, a binary function symbol g, and constants a and b; and a unary predicate P).

[1] (a) Determine which of the following conjunctions of literals are T-satisfiable, and give a model if a formula is satisfiable. In EQ:

- $x = y \land y = z \land x \neq w$
- $x = y \land y = z \land x \neq w \land w = z$

In EUF:

- $\mathsf{a} = \mathsf{b} \land \mathsf{P}(\mathsf{f}(\mathsf{a})) \land \neg\mathsf{P}(\mathsf{f}(\mathsf{b}))$
- $\mathsf{a} \neq \mathsf{b} \land \mathsf{f}(\mathsf{a}) = \mathsf{b} \land \mathsf{g}(\mathsf{a}, \mathsf{b}) = \mathsf{b}$

[2] (b) Determine which of the following entailments and equivalences modulo EQ or EUF hold. If an entailment or equivalence does *not* hold, give a counterexample (i.e., a model that satisfies only one but not the other formula).

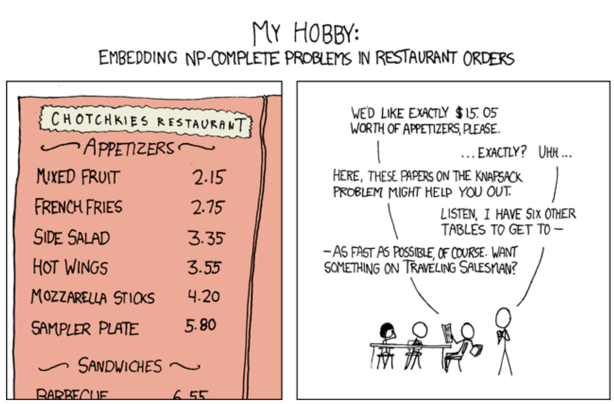
- $x = y \land y = z \land x \neq w \vDash_{\mathsf{EQ}} w \neq z$
- $x \neq y \land y \neq z \vDash_{\mathsf{EQ}} x \neq z$
- $\mathsf{a} = \mathsf{f}(\mathsf{a}) \land \mathsf{g}(\mathsf{a}, \mathsf{b}) = \mathsf{g}(\mathsf{b}, \mathsf{a}) \vDash_{\mathsf{EUF}} \mathsf{g}(\mathsf{f}(\mathsf{a}), \mathsf{b}) = \mathsf{g}(\mathsf{b}, \mathsf{f}(\mathsf{f}(\mathsf{f}(\mathsf{a}))))$
- $\mathsf{a} = \mathsf{b} \land \mathsf{f}(\mathsf{a}) \neq \mathsf{g}(\mathsf{b}, \mathsf{b}) \vDash_{\mathsf{EUF}} \mathsf{f}(\mathsf{b}) \neq \mathsf{g}(\mathsf{a}, \mathsf{b})$
- $\mathsf{a} = \mathsf{b} \land \mathsf{f}(\mathsf{a}) \neq \mathsf{g}(\mathsf{b}, \mathsf{b}) \equiv_{\mathsf{EUF}} \mathsf{f}(\mathsf{b}) \neq \mathsf{g}(\mathsf{a}, \mathsf{b})$

[3] $\boxed{2}$ Check satisfiability of the following formulas using DPLL(EUF).

(a) $\mathsf{a} = \mathsf{b} \land (\mathsf{b} = \mathsf{c} \lor \mathsf{b} = \mathsf{d}) \land (\mathsf{f}(\mathsf{a}) \neq \mathsf{f}(\mathsf{c}) \lor \mathsf{f}(\mathsf{a}) \neq \mathsf{f}(\mathsf{b})) \land \mathsf{f}(\mathsf{b}) \neq \mathsf{f}(\mathsf{d})$

(b) $\mathsf{f}(\mathsf{b}) = \mathsf{g}(\mathsf{f}(\mathsf{a}), \mathsf{b}) \land \mathsf{f}(\mathsf{f}(\mathsf{a})) = \mathsf{g}(\mathsf{b}, \mathsf{b}) \land (\mathsf{f}(\mathsf{a}) = \mathsf{a} \lor \mathsf{f}(\mathsf{a}) = \mathsf{b}) \land \mathsf{g}(\mathsf{a}, \mathsf{b}) = \mathsf{b} \land$
$(\mathsf{f}(\mathsf{b}) \neq \mathsf{b} \lor \mathsf{f}(\mathsf{g}(\mathsf{a}, \mathsf{b})) \neq \mathsf{g}(\mathsf{b}, \mathsf{b}))$

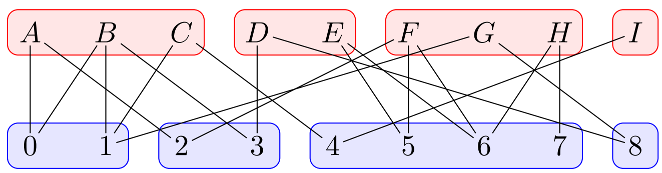
$\boxed{3}$



[2] (a) Help the waiter: use an SMT encoding with linear arithmetic to determine a possible combinations of appetizers for \$15.05.
You can formulate the problem in `SMT-LIB2` and solve it with the Z3 web interface, or use the Python bindings.
Hint: if you use Z3 you might have to apply `ToReal(...)` (in Python) or `fp.to_real` (in SMTLIB) to integer variables to make the solver compute with rational numbers.

[1] ⋆(b) Field survey: Pose an NP complete problem to a waiter, document the reaction on video and submit it.

[2] $\boxed{4}$ Nine prisoners A, B, C, ..., I are to be assigned jobs $0, 1, \ldots, 8$ in a workshop within the prison. Due to their qualifications, not all assignments are possible. The following graph shows who is qualified for which job:



Every job is situated in a manufacturing unit (indicated in blue). Prisoners are also known to be associated with four different mafia clans (indicated in red). Now the prison management wants to assign everyone a workplace such that no two people from the same clan work in the same manufacturing unit. Every workplace can be taken by only one person.[1]

Can you find an SMT encoding to solve the problem?

Hint: You could define a sort `person` and a sort `place`, and define an uninterpreted function `job` that maps a person to a place, like so:

```
person = DeclareSort('person')
place = DeclareSort('place')
persons = dict([ (p, Const(p, person)) \
  for p in ["A","B","C","D","E","F","G","H","I"] ])
places = [ Const("P"+str(b), place) for b in range(0,9) ]
job = Function('job', P1, P2)
...
solver.add(Or(job(persons["A"]) == places[0], job(persons["A"]) == places[2]))
...
```

(but more constraints will be necessary). However, you can also use a different encoding.

Exercises marked with a ⋆ are optional. Solving them gives bonus points if you submit them before the course via OLAT or email.

[1] This is an instance of a known NP-complete matching problem.