TouIST in Progress

Hovhannes Sirunyan
French University in Armenia (UFAR)
Yerevan, Armenia
e-mail: I19.Sirunyan.Hovhannes@etud.ufar.am

Dominique Longin
IRIT, CNRS
Toulouse, France
e-mail: Dominique.Longin@irit.fr

Frederic Maris
IRIT, University of Toulouse
Toulouse, France
e-mail: Frederic.Maris@irit.fr

Sergei Soloviev
IRIT, University of Toulouse
Toulouse, France
e-mail: Sergei.Soloviev@irit.fr

***Abstract*—This work deals with logical formalization and problem solving using automated solvers. The goal of TouIST is to allow the user to concentrate on the modeling of a given problem without worrying about the technical details related to the use of solvers: TouIST provides a simple language to generate logical formulas (as an input to solvers) from a problem description, and allows to model many static or dynamic combinatorial problems. All this can be very helpful as a teaching support for logics and discrete mathematics. Users may benefit from the regular improvements of SAT, QBF, or SMT solvers to solve concrete problems efficiently, e.g., different classes of planning tasks in Artificial Intelligence. We consider recent updates to TouIST dealing with modal logic (mostly due to the first author), and discuss possible applications of TouIST in scientific research.**

***Keywords*—TouIST, logical solvers, finite modelling, modal logic**

I. Introduction

TouIST (Toulouse Integrated Satisfiability Tool), an open source project[1], offers a high-level, friendly language for logically modeling various problems in a very compact way using already existing solvers to solve concrete logical and combinatorial problems efficiently. It consists of a graphical interface allowing interactive input of the target model; a translation module from the input language of TouIST into a language directly understandable by different logical solvers; a module for viewing models calculated by the solvers.

The flexible design of TouIST makes it possible to use TouIST with different solvers. TouIST can call on four different types of solvers: SAT solvers (propositional logic or logic of predicates on a finite domain), QBF (authorizing quantification on propositional formulas), SMT (SAT Modulo Theories for the treatment of problems involving numeric calculus on integer or rational numbers) and MODAL (Modal Logic). TouIST is an open platform that may be progressively extended.

As TouIST comes with a well-written documentation, it has become an important teaching support tool for logics and discrete mathematics at the University of Toulouse [1]. However, the potential of TouIST as a pedagogical and research tool is far from being fully exploited. It is good not only for creating "demos" but also for the verification of examples (combinatorial in nature) that are too complex to be reliably checked by a human researcher.

[1]https://github.com/touist/touist

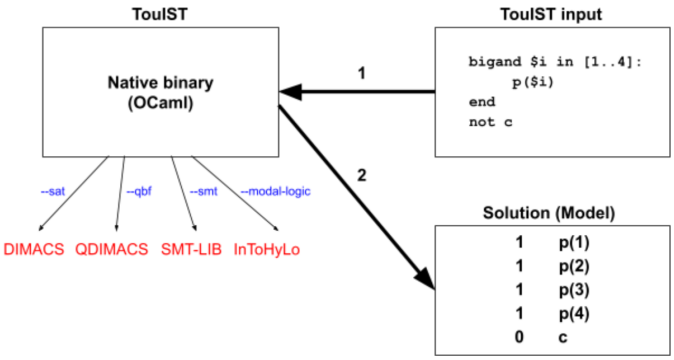

Fig. 1. TouIST architecture

II. TouIST Capacities

Iterated connectives. Iterated conjunction and disjunction ($\bigwedge$ and $\bigvee$) used in TouIST may be seen as the universal quantifier $\forall$ and the existential quantifier $\exists$ over finite sets of values of indices. It permits even dependent quantification over finite sets, since subsets of a given set may be used as indices.

Static reasoning. TouIST allows us to encode and solve static generalized games such as the well known Sudoku for an $N \times N$ grid (composed by N regions, i.e., grids of size $R \times R$ with $N = R^2$). In a similar way to Sudoku, the Binario (binary game) consists in filling a grid by deduction with only 0s and 1s. In particular, we can encode the rule "each row and column must contain as many 0s as 1s".

Finding a Winning Strategy. The language of QBF allows us to express naturally and concisely the existence of winning strategies, which are described in [2]. In a typical 2-player turn-based game, the moves of player 0 (for whom we are searching a winning strategy) will be existentially quantified, while those of his opponent will be universally quantified: we look for the moves of player 0 that will lead him to

victory, regardless of his opponent's moves. This scenario is well presented in [3] using the Nim game.

Planning tasks. A planning task can be transformed into a propositional formula with models corresponding to solution plans (i.e., sequences or steps of actions starting from an initial state and leading to a goal). These models can be found using a SAT solver [4].

Beyond classical planning. TOUIST allows us to encode and solve conformant planning tasks with QBF. It can also be used to solve temporal planning tasks involving durative actions, exogenous events, and temporally extended goals with SMT encodings.

III. S5 EXTENSION

Let us explain in more detail the Modal Logic S5 extension and its implementation. The state-of-the-art Cheetah [5] solver for Modal Logic S5 has been added on top of the existing solvers in TOUIST. It is now possible to solve Modal Logic S5 problems in the most efficient way. We plan to add a similar Modal Logic S4 extension.

Modal Logic S5 modalities can be used via:

- $[](...)\ for\ \Box(...)$ in order to express necessity
- $<>(...)\ for\ \diamond(...)$ in order to express possibility

If the formula provided by the user is satisfiable, the possible worlds and their valuations are returned to the user.

In TOUIST, we have further minimized the model obtained from S5 Cheetah solver by eliminating possible worlds with duplicated valuations.

Example 1:

- Real-World "Weather Scenario".
  1) If the sky is cloudy, then it is possibly going to rain.
  2) If it is raining, then the ground is necessarily wet.
  3) The ground is wet.
- Modal Logic.
  1) $C \rightarrow \diamond R$
  2) $R \rightarrow \Box W$
  3) W
- TOUIST code.
  1) $C =><> (R)$
  2) $R => [](W)$
  3) W
- TOUIST result.
  SAT
  $w1 : -R\ -W$
  $w2 : -C\ -R\ \ W$
  worlds count: 2
- TOUIST result explanation.
  On the first line, SAT was received, meaning the used formulas are satisfiable in the Modal Logic S5. Then the possible worlds with their corresponding valuations have been returned, where $w2$ becomes the actual world, since W was used in 3). (Here, "$-R$" indicates that R is false, and the same applies to the other similar propositions).

IV. HIGHER ORDER LOGICS AND ALGEBRA

One of the useful features of TOUIST is the indexation of $\bigwedge$ and $\bigvee$ by the elements of finite sets. Combined with the construction of $powerset$ (also supported), it opens the way to various applications to higher-order logics and algebra.

For example, predicates over a finite set X may be encoded using indexation by subsets of X.

Using the Cartesian product of sets (defined in TOUIST) and the $powerset$ operation one may define n-ary predicates as well.

Example 2: The program in TOUIST

```
$N = 3
$X=[1..$N]
$Y = powerset($X)
bigand $j in $Y:
  (p($j)<=> (bigand $i in $j: q($i) end) and
  (bigand $i in $X when (not ($i in $j)): not q($i) end))
end
```

will produce the formula

$$\bigwedge_{j\in Y}(p_j \iff \bigwedge_{i\in j} q_i \wedge \bigwedge_{i\in X, i\notin j} \neg q_i).$$

It says that p_j is true exactly on elements of $j \subset [1..N]$ (q_i represents the value of p_j on i).

Taking the subsets generated by the iterated $powerset$ construction as indices allows representing higher-order logic formulas over finite sets.

The formula $\forall P.P(x)$ in second-order logic (with P being a predicate variable) is false. In TOUIST it can be represented by $\bigwedge_{j\in Y} p_j$ (with p_j as above), which is false (in particular, because $\emptyset$ is among the indices).

Remark 1: One may also recall that if we consider propositional formulas over a finite set $S = \{s_1, ..., s_k\}$ of propositional variables, then the constant $False$ may be replaced by conjunction $s \wedge s_1 \wedge ... \wedge s_k$ (with $s \notin S$), preserving provability. This is true for many systems of propositional calculus, and not only for classical logic [8].

If necessary, one may add to the previous TOUIST code:

```
$Z = powerset($Y)
bigand $k in $Z when $k!=[]:
  (S($k)<=> (bigor $j in $k: p($j) end))
end
```

The indices k are the families of subsets of Z (the empty family is excluded), and S are defined as disjunctions of p over subsets $j \in k$. This code represents the formula

$$\bigwedge_{k\in Z, k\neq\emptyset} (S(k) \iff (\bigvee_{j\in k} p(j))).$$

This example shows how to obtain dependent types $S(i)$. Respectively, $\bigwedge$ and $\bigvee$ with appropriate dependencies between indices will represent dependent $\forall$ and $\exists$ over finite sets.

The same formula may be written down with quantifiers as

$$\forall k \in Z, k \neq \emptyset.(S(k) \iff \exists j \in k.p(j)).$$

(The scope of $\exists$ depends on k.)

Other examples of this kind, as well as examples representing various algebraic structures, may be found in [1], [3].

Finite models, especially those containing hierarchical structures and relations, are notoriously difficult for human analysis and prone to error. Given that, we believe that TouIST may be especially useful as a research tool.

V. Applications to Linguistics

Some applications of dependent types to linguistic analysis were considered in [6].

For example, dependencies between linguistic quantifiers (such as "all", "some", and "no one") may influence the word order in natural language phrases or affect their meaning interpretation. Finite models (supported by TouIST) are usually sufficient there, and TouIST may be used for illustrative purposes or as a tool for analysis.

Example 3: J.R.R. Tolkien mentioned that in his childhood, he was puzzled why one may say "A great green dragon", but to say "A green great dragon" is considered wrong ([7], p.31).

Here is an outline of the explanation.

- A (finite) nonempty set of all dragons D is given. It contains a nonempty subset of green dragons G.
- In every nonempty subset $S \subseteq D$ (including D), there exists exactly one great dragon. This can be defined by the relation $Great(S, d)$.
- If $Great(S, d)$ then $d \in S$. However, to be great in S does not mean to be great in another subset S' or in D.
- Thus, $Great(G, d) \nRightarrow Great(D, d)$. In particular, a great green dragon is not necessarily a great dragon.
- And a green great dragon may not exist even if there exists a great green dragon, or (if it does exist) may be somewhat ridiculous (if the green color among dragons is not appreciated).

How this may be formalized in TouIST?

```
$D=[small, medium, large]
$P=powerset($D)
bigand $S in $P when $S!=[]:
(bigor $d in $S:
great($S,$d) and
(bigand $e in $D when $e!=$d :
not great($S,$e)
end)
end)
end
```

This program represents the formula

$$(*) \bigwedge_{S \subseteq D, S \neq \emptyset} (\bigvee_{d \in S} great_{S,d} \wedge (\bigwedge_{e \in D, e \neq d} \neg great_{S,e}))$$

that claims the existence of a unique great dragon in every $S \subseteq D$, $S \neq \emptyset$. With quantifiers and relation $great(S, d)$:

$$\forall S \subseteq D, S \neq \emptyset. \exists d \in S.$$

$$(great(S, d) \wedge \forall e \in D, e \neq d. \neg(great(S, e))).$$

We may add the line:

```
$Green=[small]
```

(the subset of green dragons contains only the "small"), and then (*) implies that $great_{Green,small}$ is true.

Possible definitions of greatness, according to (*), include the case when only "large" is great in D and "a green great dragon" does not exist, while "a great green dragon" does.

The same reasoning may be reproduced using dependent types as in [6].

VI. Conclusion

TouIST makes it possible to easily use SAT, SMT, QBF, and Modal Logic S5 solvers by abstracting away their internal complexities and optimizations. By using TouIST, one has to focus only on their main problem without worrying about the underlying solvers. The Modal Logic S5 extension in TouIST has opened a new window of opportunity to use logical modalities, such as possible and necessary in TouIST by utilizing the state-of-the-art Modal Logic S5 solver Cheetah.

TouIST displays strong sides of flexible, user-friendly interfacing between specialized programs (solvers) and general-purpose languages (logic) in teaching and research. In the near future, we plan to update TouIST to include other Modal Logic solvers.

This work is currently under active development.

We consider extending TouIST in the direction of game semantics of Modal Logics and apply it to study Boolean Games.

Technically speaking, Modal Logic solvers are based on Kripke semantics. Kripke semantics for Intuitionistic Logic is also well known. Many features of Modal, Intuitionistic, Linear and other non-classical logics may be studied using higher-order logics (cf. [8]). This direction of study is also on the agenda.

A. Contact address

Sergei Soloviev, IRIT, University of Toulouse
118 route de Narbonne
31069 Toulouse, France
E-mail: Sergei.Soloviev@irit.fr
Phone: +(33) 675241341

Acknowledgment

The authors would like to thank Andreas Herzig for helpful discussions.

References

[1] O. Gasquet, D. Longin, E. Lorini, F. Maris, P. Regnier, S. Soloviev. "TouIST, a Teacher-and Student-Friendly Language for Propositional Logic and Discrete Mathematics", *Computer tools in education*, no. 2, pp. 13-25, 2021.

[2] D. Kroening, O. Strichman, *Decision Procedures - An Algorithmic Point of View*, Second Edition. Texts in Theoretical Computer Science. An EATCS Series, Springer, 2016.

[3] J. Fernandez, O. Gasquet, A. Herzig, D. Longin, E. Lorini, F. Maris, P. Régnier, "TouIST: a Friendly Language for Propositional Logic and More", *Proceedings of the 29th International Joint Conference on Artificial Intelligence (IJCAI)*, Yokohama, Japan, pp. 5240-5242, 2020.

[4] H.A. Kautz, B. Selman, "Planning as Satisfiability", *Proceedings of the 10th European Conference on Artificial Intelligence, ECAI 92*. Vienna, Austria, pp. 359-363, 1992.

[5] P. Huang, M. Liu, P. Wang, W. Zhang, F. Ma, J. Zhang, “Solving the Satisfiability Problem of Modal Logic S5 Guided by Graph Coloring”, *Proceedings of the 28th International Joint Conference on Artificial Intelligence, IJCAI 2019, main track*. Macao, China, pp. 1093-1100, 2019.
[6] Z. Luo Z., S. Soloviev, “Dependent Event Types”, *WoLLIC20, LNCS 10388*, Springer, 2017.
[7] H. Carpenter. *J.R.R. Tolkien. A biography*, Paperback edition. George Allen and Unwin, 1978.
[8] S. Soloviev, “Reductions in Linear Logic”, *Mathematical Structures in Computer Science*, vol. 5, pp. 483-499, 1995.