TouIST, a Pedagogical Tool for Logic, Algebra and Discrete Mathematics

PCA-2023, Saint-Petersburg

Sergei Soloviev, IRIT, France

21/04/2023

TouIST as a Pedagogical Tool

- This introduction is based on joint paper:
- O. Gasquet, D. Longuin, E. Lorini, F. Maris, P. RÃ©gnier, 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.

TouIST as a Pedagogical Tool

- TouIST offers a high-level friendly language for logically modeling various problems in a very compact way using solvers.
- **It consists of**:
- A graphical interface allowing interactive input of the target model;
- A translation module (compiler) from the input language of TouIST into a language directly understandable by different solvers;
- A module for viewing models calculated by the solvers.
- This flexible design permits to use TouIST with different solvers.

TouIST as a Pedagogical Tool

- **Currently:**
- with SAT solvers (propositional logic or logic of predicates on 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).
- **The goal:** 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.
- (linear space translation of formulas into prenex and conjunctive normal form using extension rules [Tseitin1983], translation into language DIMACS, QDIMACS or SMT-LIB depending on the selected solver)
- (languages used as standard as input to solvers are not very easy to handle directly)

TouIST as a Pedagogical Tool

- Beyond the Boolean connectives of propositional logic, the input language of `TouIST` has sets, conjunctions and disjunctions parametrized by sets, abbreviations...
- One can directly express complex propositional formulas such as:
- $$\bigwedge_{i \in \{1..N\}} \bigvee_{X \in S(i)} \bigwedge_{n \in X} \bigwedge_{m \in Y | m \neq n} (p_{i,X,n} \Rightarrow \neg p_{i,X,m})$$
- the variable N may be here a particular integer, the $S(i)$ as sets of sets of symbols for each $i \in \{1, \ldots, N\}$, and Y as a set of symbols.
- E.g., $N = 2$, $S(1) = \{\{blue, red\}, \{red\}\}$, $S(2) = \{\{red\}, \{blue\}, \{white, blue, red\}\}$, and $Y = \{white, red\}$.

TouIST as a Pedagogical Tool

- In the TouIST input language:

```
$N = 2
$S(1) = [[blue, red], [red]]
$S(2) = [[red], [blue], [white, blue, red]]
$Y = [white, red]
bigand $i in [1..$N]:
  bigor $X in $S($i):
    bigand $n in $X:
      bigand $m in $Y when $m!=$n:
        p($i,$X,$n) => not p($i,$X,$m)
      end
    end
  end
end
```

TouIST as a Pedagogical Tool

- One can use multiple binding of indexes as in $\bigwedge_{i \in A, j \in B}$ and rich computations on indexes as well as on domain sets as in $\bigwedge_{i \in (A \cup (B \cap C))}$.
- In the `TouIST` input language:

```
bigand $i,$j in $A,$B:
  ...
end
bigand $i in $A union ($B inter $C):
  ...
end
```

- **Remark.** One may remember that $\bigwedge$ and $\bigvee$ represent also universal quantifier $\forall$ and existential quantifier $\exists$ over finite sets of indexes.

TouIST as a Pedagogical Tool

- Running the solver only consists in clicking a button.
- The tool displays the models successively computed by the solvers in the syntax of the input formula.
- Literals of interest can be filtered by regular expressions.
- `TouIST` can also be used entirely from the command line and/or batch modus for interfacing with intelligent agent architectures capable of reasoning and planning actions.
- Typically for example, checking the validity of an argument, determining the executable actions, checking that a plan is valid or even calculate a complete plan of actions to satisfy a goal, etc.
- `TouIST` is publicly available for download from the following site: `https://www.irit.fr/TouIST/`

Examples

- Usefulness of TouIST in teaching comes from the possibility of easily encode (and then display the solutions) of sufficiently sophisticated examples.
- We shall briefly present several examples in each of the categories mentioned in the title:
- Logics
- Algebra
- Discrete maths

Examples

- Solvers, associated with `TouIST` produce models that may be displayed "sequentially" (one after another) on the screen.
- Suppose that $H = \{H_1, \ldots, H_n\}$ is a set of n assumptions (logical formulas) and C a formula.
- It would be tedious to verify that C is true for all models of H.
- However, one proceeds indirectly by using the property: $H \models C$ if and only if $H \cup \{\neg C\}$ is unsatisfiable. in other words, to check if $H \models C$, we will test if the formulas H_1, H_2, $\ldots$, H_n, $\neg C$ taken *all together* are satisfiable.
- If this is not the case one can conclude that $H \models C$. If this is the case, one will have at least one counter-model.

Examples (admissibility of rules)

In the sequent calculus for classical logic

- Each sequent $S = H_1, ..., H_n \vdash C$ may be represented as the formula $\overline{S} = H_1 \wedge ... \wedge H_n \Rightarrow C$.
- To verify the validity of the rule

$$\frac{S_1 ... S_k}{S}$$

- one needs just to verify that the set $\overline{S_1}, ..., \overline{S_n}, \neg\overline{S}$ is not satifiable.

Examples (admissibility of rules)

In non-classical calculi (to some extent)

- If one wants to use `TouIST` to verify admissibility in other calculi, the idea to use conservativity results comes to mind.
- E.g., Glivenko classes of formulas where classical derivability is conservative w.r.t. intuitionistic.
- However, the description of these classes is rather complex if one thinks about a pedagogical "demo".
- There are some others that may be used. An old and rather technical theorem of mine [Babaev-Soloviev 1979, Soloviev 1984]) established that
- If the sequent $S = H_1, ..., H_n \vdash C$ in the $(\wedge, \Rightarrow)$-fragment of propositional calculus is **balanced** (each variable occurs exactly twice with opposite signs) then it is derivable classically iff it is derivable intuonistically (or even in multiplicative linear logic).

1. So, for example, the rules

$$\frac{\Gamma, C \Rightarrow p \vdash p}{\Gamma \vdash C} \quad \frac{\Gamma, A \Rightarrow p, p \Rightarrow B, \Delta \vdash C}{\Gamma, A \Rightarrow B, \Delta \vdash C}$$

   are admissible in all these calculi (with p fresh),

2. and the rule

$$\frac{\Gamma, A, \Delta \vdash C}{\Gamma, (A \wedge p \Rightarrow q) \Rightarrow q, \Delta \vdash C \wedge p}$$

   is not.

3. This kind of examples may help to discuss such questions as the question of conservativity, the role of substitution, structural rules.

Examples (second order logic)

- The possibility to use sets as indexes and quantification over finite sets may be used to illustrate higher order properties.
- $$\bigwedge_{j \in \mathcal{P}\{1..N\}} \bigvee_{i \in j} \bigwedge_{k \in j} p_{ik}$$
- Here p_{ik} represent some relation on $[1..N]$, $j \in \mathcal{P}\{1..N\}$ are subsets, and the formula represents the condition of wellordering. (Other formulas need to be added to make p_{ij} an order relation.)

```
$N = 10
$Y = powerset([1..$N])
bigand $j in $Y when $j!=[]:
bigor $i in $j$:
bigand $k in $j:
  p($i, $k)
end
end
end
```

An algebraic example

- Since indexed propositional variables in TouIST may be routinely used to represent predicates over finite sets, it is convenient to use predicates to represent algebraic operations and express their properties via $\vee$ and $\wedge$.
- E.g., x_{ijk} may represent $i \times j = k$ for some binary operation $\times : X \times X \to X$ on a finite set X.
- $$\bigwedge_{i \in X} \bigwedge_{j \in X} \bigwedge_{k \in X} x_{ijk} \Rightarrow x_{jik}$$

  will represent commutativity of $\times$.
- Associativity of $\times$ will be expressed by

$$\bigwedge_{i \in X} \bigwedge_{j \in X} \bigwedge_{k \in X} \bigwedge_{l \in X} \bigwedge_{m \in X} \bigwedge_{n \in X} (x_{ijk} \wedge x_{jlm} \wedge x_{kln} \Rightarrow x_{imn}).$$

An algebraic example

- The formulas

$$\bigwedge_{i\in X}\bigwedge_{j\in X}\bigvee_{k\in X} x_{ijk},$$

- $$\bigwedge_{i\in X}\bigwedge_{j\in X}\bigwedge_{k\in X}\bigwedge_{l\in X, l\neq k}(x_{ijk}\Rightarrow\neg x_{ijl})$$

- express the fact that $\times$ is defined for all i, j and has unique value k.
- They may be easily written using `TouIST`. (See demo in the end.)
- `TouIST` is sufficiently powerful to produce all such operations $\times$ for the sets X that are large enough for teaching purposes. Each operation will be displayed as a line in the truth table.
- Moreover, `TouIST` may be used to verify quickly simple equalities modulo theory, associativity and commutativity in this example.

Other kinds of examples

- TouIST is good for reasoning with other kinds of examples of more applied character.
- **Static reasoning.**
- **Dynamic reasoning.**
- **Solving planning tasks.**
- Many of them may be found in our paper [Gasquet et al., 2021].

Other kinds of examples

- Here I just briefly outline the examples.
- **Solving puzzles with SAT**
- `TouIST` allows to encode and solve static generalized games such as the well known Sudoku for a $N \times N$ grid.
- It also allows to solve well-known puzzles and games involving epistemic deductive reasoning, given existing polynomial embeddings of fragments of epistemic logic into propositional logic. This includes “Guess Who?” and the muddy children puzzle
- **Solving Puzzles with SMT**
- Binario (binary game) consists in filling a grid by deduction with only 0 and 1. It is possible to model it in propositional logic, but to obtain a more compact encoding one can use SMT (SAT Modulo Theories) with atoms of QF-LIA (linear arithmetic on integers)

Other kinds of examples

- **Dynamic Reasoning with** TouIST
- One may simulate a two-players game, for example Nim.
- It is possible to describe the tree of a game.
- The language of QBF allows to express naturally and concisely the existence of winning strategies (TouIST includes QBF as one of possible options).
- TouIST natively integrates the QBF solver Quantor 3.2 and can be interfaced with other solvers supporting the QDIMACS format. Selecting this prover in TouIST allows to use quantifiers $\forall$ and $\exists$ on propositional variables.

Other kinds of examples

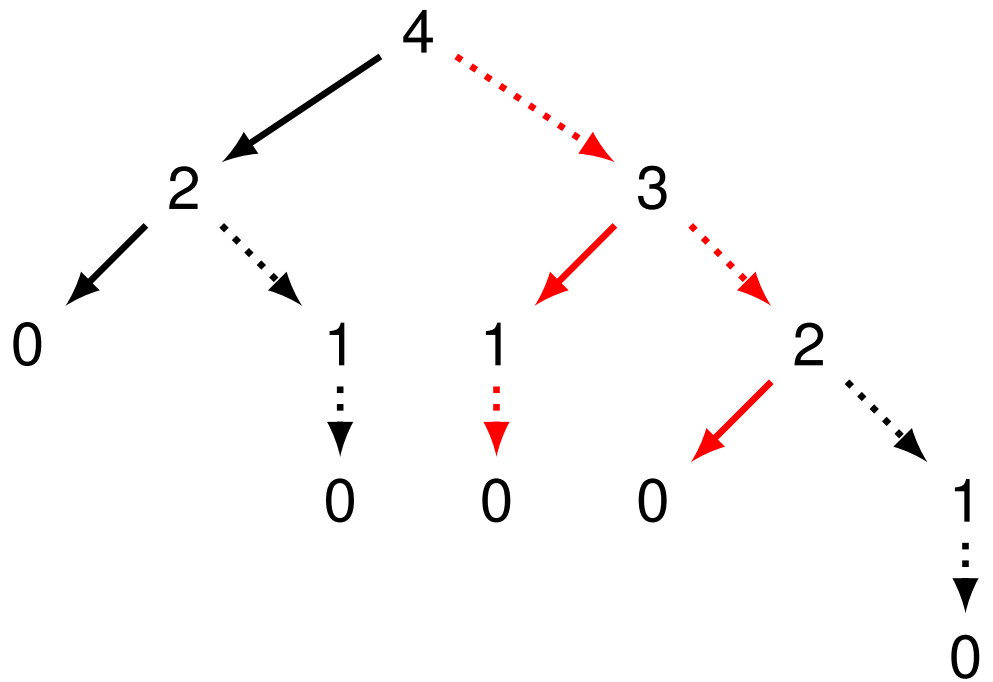

Figure: Solutions for Nim's game with 4 matches and 2 players. The winning strategy for player 0 is in red.

Other kinds of examples

- **Planning as satisfiability**
- A planning task can be transformed into a propositional formula whose models correspond 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
- **It is possible to include temporal aspects:**
- beyond classic planning, `TouIST` allows 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.

Conclusion

- TouIST offers a friendly language together with a modular tool that makes it easier to use SAT, SMT and QBF solvers.
- TouIST can be seen as a compiler from extended and high-level logical languages to efficient independent solvers.
- These two sides give it great ease of use, a wide application spectrum and good computational performance.
- As such, it constitutes an original and unique tool of its kind.

Conclusion

- We use it as part of the introductory course to logic of bachelor of mathematics and computer science, but also for the master's degree, as part of practical work and projects. Students are thus called upon to go through the entire process, from formalization to problem solving that goes far beyond toy problems that can be solved by hand.
- Even more, `TouIST` is already used by researchers in the context of work carried out in our laboratory and involving logical modeling (planning, epistemic reasoning via translation into QBF, . . .), it fills a lack existing in formal calculation software such as Maple, SageMath, Mathematica or Maxima which only anecdotally integrate logical tools.
- In fact, `TouIST` may be useful in all research domains where finite modeling is relevant.

Conclusion

THANKS FOR YOUR ATTENTION!

References

[Gasquet et al., 2023] O. Gasquet, D. Longuin, E. Lorini, F. Maris, P. Régnier, 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 (Engl.); doi:10.32603/2071- 2340-2021-2-13-25

[Tseitin 1983] Tseitin G.S. On the Complexity of Derivation in Propositional Calculus. Automation of Reasoning: 2: Classical Papers on Computational Logic 1967-1970, pp. 466-483, 1983.

[Babaev-Soloviev 79] A. A. Babaev, S. V. Solov'ev. A coherence theorem for canonical maps in Cartesian Closed Categories. Zap. Nauchn. Seminarov LOMI, v.88, pp.3 -29 (1979). English Translation: Journal of Soviet Math., v.20 , pp.2263-2282 (1982)

[Soloviev 84] Ph.D. Thesis (LOMI).