TESC Proof Format Specification

Seulkee Baek

15th July, 2020

1 Overview

Theory Extensible Sequent Calculus (TESC) is a low-level, machine-checkable proof format for first-order ATPs. As its name suggests, TESC is based on sequent calculus [1] and can be flexibly extended to accommodate a variety of first-order theories. TESC files are created by compiling TPTP [3] problem files and their corresponding TSTP [4] solution files (generated by various ATPs) down to the TESC format using the TPTP-TSTP Compiler (TTC). The TSTP-TESC Verifier (TTV) can be used to verify that a TESC proof is a correct proof of a TPTP problem.

2 Proof Calculus

First, we let symbols range over domains as follows:

$$\begin{aligned}
k, m, n &\in \mathbb{N} & f, g &\in \text{functions} \\
d &\in \{l, r\} & r &\in \text{relations} \\
v &\in \text{TPTP names} & \phi, \psi &\in \text{formulas} \\
\pi &\in \text{parameters} & \Phi, \Psi &\in \text{signed formulas} \\
i, j &\in \text{IDs} & h &\in \text{hypotheses} \\
t &\in \text{terms} & \Gamma &\in \text{contexts (sets of hypotheses)} \\
\vec{t} &\in \text{lists of terms} & &
\end{aligned}$$

$$
\begin{aligned}
\pi &::= @_k \\
i &::= \nu \mid \pi \\
t &::= \#_k \mid f(\vec{t}) \mid \pi(\vec{t}) \\
\vec{t} &::= \cdot \mid t, \vec{t} \\
\phi &::= \top \mid \bot \mid r(\vec{t}) \mid \pi(\vec{t}) \mid \neg\phi \mid \phi \vee \chi \mid \phi \wedge \chi \mid \phi \rightarrow \chi \mid \phi \leftrightarrow \chi \mid \forall\phi \mid \exists\phi \\
\Phi &::= +\phi \mid -\phi \\
h &::= (i : \Phi)
\end{aligned}
$$

Figure 1: TESC abstract syntax.

We will also freely add subscripts to symbols as necessary for disambiguation.

Rule	Conditions
$\dfrac{k+1 \gg \Gamma, (i : \Phi), (@_k : \Psi)}{k \gg \Gamma, (i : \Phi)} A(i,d)$	$\Phi \overset{A(d)}{\Longrightarrow} \Psi$
$\dfrac{k+1 \gg \Gamma, (i : \Phi), (@_k : \Psi_1) \qquad k+1 \gg \Gamma, (i : \Phi), (@_k : \Psi_2)}{k \gg \Gamma, (i : \Phi),} B(i)$	$\Phi \overset{B(l)}{\Longrightarrow} \Psi_1, \Phi \overset{B(r)}{\Longrightarrow} \Psi_2$
$\dfrac{k+1 \gg \Gamma, (i : \Phi), (@_k : \Psi)}{k \gg \Gamma, (i : \Phi)} C(i,t)$	$gnd(t),\ k \gg t,\ \Phi \overset{C(t)}{\Longrightarrow} \Psi$
$\dfrac{k+1 \gg \Gamma, (i : \Phi), (@_k : \Psi)}{k \gg \Gamma, (i : \Phi)} D(i)$	$\Phi \overset{D(k)}{\Longrightarrow} \Psi$
$\dfrac{k+1 \gg \Gamma, (@_k : -\phi) \qquad k+1 \gg \Gamma, (@_k : +\phi)}{k \gg \Gamma} F(\phi)$	$gnd(\phi),\ k \gg \phi$
$\dfrac{k+1 \gg \Gamma, (i : \Phi), (@_k : \Psi)}{k \gg \Gamma, (i : \Phi)} S(i)$	$\Phi \overset{S}{\Rightarrow} \Psi$
$\dfrac{k+1 \gg \Gamma, (@_k : \Phi)}{k \gg \Gamma} T(\Phi)$	$adm(k, \Phi)$
$\dfrac{k \gg \Gamma}{k \gg \Gamma, (i : \Phi)} W(i)$	None
$\dfrac{}{k \gg \Gamma, (i : +\phi), (j : -\phi)} X(i,j)$	None

Table 1: TESC inference rules.

A	$-\phi \vee \psi \overset{A(l)}{\Longrightarrow} -\phi$	$-\phi \vee \psi \overset{A(r)}{\Longrightarrow} -\psi$
	$+\phi \wedge \psi \overset{A(l)}{\Longrightarrow} +\phi$	$+\phi \wedge \psi \overset{A(r)}{\Longrightarrow} +\psi$
	$-\phi \to \psi \overset{A(l)}{\Longrightarrow} +\phi$	$-\phi \to \psi \overset{A(r)}{\Longrightarrow} -\psi$
	$+\phi \leftrightarrow \psi \overset{A(l)}{\Longrightarrow} +\phi \to \psi$	$+\phi \leftrightarrow \psi \overset{A(r)}{\Longrightarrow} +\psi \to \phi$
B	$+\phi \vee \psi \overset{B(l)}{\Longrightarrow} +\phi$	$+\phi \vee \psi \overset{B(r)}{\Longrightarrow} +\psi$
	$-\phi \wedge \psi \overset{B(l)}{\Longrightarrow} -\phi$	$-\phi \wedge \psi \overset{B(r)}{\Longrightarrow} -\psi$
	$+\phi \to \psi \overset{B(l)}{\Longrightarrow} -\phi$	$+\phi \to \psi \overset{B(r)}{\Longrightarrow} +\psi$
	$-\phi \leftrightarrow \psi \overset{B(l)}{\Longrightarrow} -\phi \to \psi$	$-\phi \leftrightarrow \psi \overset{B(r)}{\Longrightarrow} -\psi \to \phi$
C	$+\forall\phi \overset{C(t)}{\Longrightarrow} +\phi[0 \mapsto t]$	$-\exists\phi \overset{C(t)}{\Longrightarrow} -\phi[0 \mapsto t]$
D	$+\exists\phi \overset{D(k)}{\Longrightarrow} +\phi[0 \mapsto @_k(\cdot)]$	$-\forall\phi \overset{D(k)}{\Longrightarrow} -\phi[0 \mapsto @_k(\cdot)]$
S	$+\neg\phi \overset{S}{\Rightarrow} -\phi$	$-\neg\phi \overset{S}{\Rightarrow} +\phi$

Table 2: Decomposition relations between signed formulas.

The abstract syntax and inference rules of TESC are given in Figure 1 and Table 1, respectively. Some of their notable features are:

- Positive formulas correspond to left formulas of two-sided sequent calculus, and negative formulas to right.
- Instead of left and right rules for each connective, there are A, B, C, D, S rules for decomposition of signed formulas. The A, B, C, D rules are similar to the $\alpha, \beta, \gamma, \delta$ rules of Smullyan's analytic tableaux [2], and the S rule is used for decomposing negations. The application of decomposition rules require certain decomposition relations to hold between signed formulas, which are given in Table 2.
- $gnd(X) := X$ is ground (i.e., has no free variable).
- The F, W, and X rules correspond to the cut, weakening, and init rule (or axiom) of standard sequent calculi.
- Variables have the form $\#_k$, where k is the De Brujin index of the variable. Due to the use of De Brujin indices, quantifiers are written without variables.
- $X[k \mapsto t]$ is the result of replacing all variables in X bound to the kth quantifier with term t. Substitution is recursively defined by
  - ⋆ $\#_m[k \mapsto t] = \#_m$, if $m < k$
  - ⋆ $\#_k[k \mapsto t] = t$
  - ⋆ $\#_{m+1}[k \mapsto t] = \#_m$, if $m + 1 > k$
  - ⋆ $(f(t_0, \ldots, t_m))[k \mapsto t] = f(t_0[k \mapsto t], \ldots, t_m[k \mapsto t])$
  - ⋆ $C[k \mapsto t] = C$, if $C \in \{\top, \bot\}$
  - ⋆ $(\neg\phi)[k \mapsto t] = \neg(\phi[k \mapsto t])$
  - ⋆ $(\phi \bullet \psi)[k \mapsto t] = (\phi[k \mapsto t]) \bullet (\psi[k \mapsto t])$, if $\bullet \in \{\vee, \wedge, \rightarrow, \leftrightarrow\}$
  - ⋆ $(Q\phi)[k \mapsto t] = Q(\phi[k+1 \mapsto incr(t)])$, if $Q \in \{\forall, \exists\}$, where $incr(t)$ is the result of incrementing all De Brujin indices in term t by 1
  - ⋆ $(r(t_0, \ldots, t_m))[k \mapsto t] = r(t_0[k \mapsto t], \ldots, t_m[k \mapsto t])$

  Note that the incrementation and decrementation of De Brujin indices are necessary to ensure that variables are bound to the same quantifiers before and after substitution.

- We say that k is *fresh* in X if, for any natural number m such that $@_m$ occurs in X, $m < k$. X can be any kind of expression, including a term, a formula, or a context.

- $k \gg X := k$ is fresh in X.

- Sequents have the form $k \gg \Gamma$, where k is the *counter* associated with the sequent.

  - ⋆ The overloading of the symbol '$\gg$' is intended: the invariant in any well-formed TESC proof is that, for any sequent $k \gg \Gamma$ in the proof, k is indeed fresh in Γ.
  - ⋆ Due to the invariant, we can use parameters of the form $@_k$ whenever we need to introduce new symbols that are guaranteed not to occur in a given context.
  - ⋆ For instance, the D rule uses parameters as instantiation terms for existential formulas, which automatically takes care of eigenvariable conditions.
  - ⋆ Parameters are used as IDs of new hypotheses, which ensures that no two hypotheses in a context have shared IDs.
  - ⋆ Parameters are also used as fresh relation symbols in predicate definitions or fresh function symbols in choice axioms.

- The T rule introduces new hypotheses that preserve satisfiability under the given background theory. There are two main uses of the T rule:

  - ⋆ Some steps in TSTP solutions (e.g. Skolemization) introduce formulas that are not logically entailed by existing premises, but still preserve satisfiability in respect to all existing formulas. The T rule is used to introduce such formulas in TESC proofs.
  - ⋆ It also is used to introduce axioms of the background theory. For instance, it may be used to add $+(0 < 1)$ to the context when working in the theory of integer arithmetic.

- The set of signed formulas Φ for which $adm(k, \Phi)$ holds (i.e., the set of formulas admitted by the T rule) depends on both the target background theory and the kind of satisfiability-preserving inferences one intends to use. In this sense, TESC is better understood as a scheme for proof formats (rather than a single, concrete proof format) where one can obtain specific *implementations* of TESC by giving a suitable definition of the *adm* predicate.

- In order to be machine-checkable, each definition of *adm* must also come with an efficient algorithm for determining whether $adm(k, \Phi)$ holds for any given k and Φ.

- In the current implementation of TESC, $adm(k, \Phi)$ holds if and only if Φ is ground and has one of the following forms:

  - ⋆ $+\top$
  - ⋆ $-\bot$
  - ⋆ $+\forall(\#_0 = \#_0)$
  - ⋆ $+\forall\forall(\#_1 = \#_0 \to \#_0 = \#_1)$
  - ⋆ $+\forall\forall\forall(\#_2 = \#_1 \to (\#_1 = \#_0 \to \#_2 = \#_0))$
  - ⋆ $+\forall\forall(\#_1 = \#_0 \to \ldots \forall\forall(\#_1 = \#_0 \to (f(\#_{2k+1}, \ldots, \#_1) = f(\#_{2k}, \ldots, \#_0))) \ldots)$, where f is any function symbol and $k+1$ is the number of equational antecedents
  - ⋆ $+\forall\forall(\#_1 = \#_0 \to \ldots \forall\forall(\#_1 = \#_0 \to (r(\#_{2k+1}, \ldots, \#_1) \to r(\#_{2k}, \ldots, \#_0))) \ldots)$, where r is any relation symbol and $k+1$ is the number of equational antecedents
  - ⋆ $\forall \ldots \forall(\exists\phi \to \phi[0 \mapsto @_k(\#_0, \ldots, \#_k)])$, where ϕ is any formula and k is the number of outer universal quantifiers $\forall \ldots \forall$.
  - ⋆ $\forall \ldots \forall(@_k(\#_0, \ldots, \#_k) \leftrightarrow \phi)$, where ϕ is any formula and k is the number of outer universal quantifiers $\forall \ldots \forall$.

- A TESC proof is a correct proof of a TPTP problem if its root sequent has the form $0 \gg \Gamma$ such that

$$\begin{aligned}\Gamma = \{(\nu : +\phi) \mid {} & \text{A lemma/axiom/hypothesis/negated conjecture } \phi_{\text{TPTP}} \\ & \text{with name } \nu \text{ exists in the TPTP problem}\} \cup \\ \{(\nu : +\neg\phi) \mid {} & \text{A conjecture } \phi_{\text{TPTP}} \text{ with name } \nu \text{ exists in the TPTP problem}\}\end{aligned}$$

  with the following caveats:

- ⋆ 0 must be fresh in Γ. In practice, this means that the original TPTP problem should not include any annotated formulas with names that match the regex pattern `@(0|[1-9][0-9]*)`.
- ⋆ All formulas occurring in the TPTP problem have unique names.
- ⋆ ϕ is the result of translating a TPTP formula ϕ_{TPTP} into TESC syntax. The translation is mostly straightforward, except that:
  - ∗ Some care is required to ensure that TESC variables with De Brujin indices still bind to the same quantifiers after translation.
  - ∗ Some operators (e.g. $\not\leftrightarrow$) in TPTP formulas have no direct equivalents in TESC and must be suitably normalized (e.g., from $\phi_{\text{TPTP}} \not\leftrightarrow \psi_{\text{TPTP}}$ to $\neg(\phi \leftrightarrow \psi)$).

3 Concrete syntax

The TESC concrete syntax uses existing TPTP syntax where possible. The parts of TESC syntax whose definitions are taken from TSTP are shown in Figure 2. The concrete syntax specific to TESC are given in Figure 3, also presented in similar BNF. Note that:

- The syntax uses prefix notation and `<dot>` extensively, which makes parsing of TESC files trivial.
- The concrete symbols `T`, `F`, `~`, `|`, `&`, `>`, `=`, `!`, `?` correspond to abstract symbols $\top, \bot, \neg, \vee, \wedge, \leftrightarrow, \rightarrow, \forall, \exists$, respectively.
- In place of `<name>` and `<functor>` of TPTP syntax, it uses `<atom>`.

```
<arrow> ::: [>]
<dot> ::: [.]
<vline> ::: [|]
<zero_numeric> ::: [0]
<non_zero_numeric> ::: [1-9]
<numeric> ::: [0-9]
<positive_decimal> ::- <non_zero_numeric><numeric>*
<decimal> ::- (<zero_numeric>|<positive_decimal>)
```

Figure 2: TESC concrete syntax (taken from TPTP)

```
<not_dot> ::: [^.]
<atom> ::= <not_dot><not_dot>*<dot>
<number> ::= <decimal><dot>
<id> ::= <atom>
<term> ::= #<number> | ^<atom><term_list>
<term_list> ::= <dot> | ;<term><term_list>
<formula> ::= ^<atom><term_list>
            | T
            | F
            | ~<formula>
            | <vline><formula><formula>
            | &<formula><formula>
            | <arrow><formula><formula>
            | =<formula><formula>
            | !<formula>
            | ?<formula>
<signed_formula> ::= +<formula> | -<formula>
<dir> ::= l | r
<proof> ::= A<id><dir><proof>
          | B<id><proof><proof>
          | C<id><term><proof>
          | D<id><proof>
          | F<formula><proof><proof>
          | S<id><proof>
          | T<signed_formula><proof>
          | W<id><proof>
          | X<id><id>
```

Figure 3: TESC-specific concrete syntax

References

[1] Gerhard Gentzen. Untersuchungen über das logische schließen. i. *Mathematische zeitschrift*, 39(1):176–210, 1935.

[2] Raymond M Smullyan. *First-order logic*. Courier Corporation, 1995.

[3] Geoff Sutcliffe. The TPTP problem library and associated infrastructure. *Journal of Automated Reasoning*, 43(4):337, 2009.

[4] Geoff Sutcliffe, Jürgen Zimmer, and Stephan Schulz. TSTP data-exchange formats for automated theorem proving tools. *Distributed Constraint Problem Solving and Reasoning in Multi-Agent Systems*, 112:201, 2004.