# The Interactive Interpretation Viewer

**Anon One**
Some where
Some place
Some country

**Anon Two**
Some where
Some place
Some country

## 1 Introduction

Automated Reasoning (AR), and Automated Theorem Proving (ATP) in particular, has focused largely on the task of proving theorems from axioms – the derivation of conclusions that follow inevitably from known facts (Robinson and Voronkov 2001). The axioms and conjecture to be proved (and hence become a theorem) are written in an appropriately expressive logic, and the proofs are often similarly written in logic (Sutcliffe et al. 2006). In this work typed first-order logic is used. In the last two decades the converse task of disproving conjectures has become increasingly important. This process depends on finding an *interpretation*, i.e., a structure that maps terms to domain elements and formulae to truth values. An interpretation that maps a formula to *true* is a *model* of the formula. A conjecture is disproved by finding an interpretation that is a model of the axioms, but maps the conjecture to *false*. A salient application area that harnesses this form of ATP is verification (D'Silva, Kroening, and Weissenbacher 2008). This work describes an interactive interpretation viewer for interpretations written in the (new) TPTP format (**?**).

## 2 The TPTP World and Languages

The TPTP World (Sutcliffe 2017) is a well established infrastructure that supports research, development, and deployment of Automated Theorem Proving (ATP) systems. The TPTP language (Sutcliffe 2022) is used for writing both problems and solutions. The top level building blocks of the TPTP language are *annotated formulae*. An annotated formula has the form:

*language*`(`*name*`,` *role*`,` *formula*`,` *source*`,` *useful_info*`)`

The *language*s supported are `cnf` (clause normal form), `fof` (first-order form), `tff` (typed first-order form), and `thf` (typed higher-order form). The *role*, e.g., `axiom`, `lemma`, `conjecture`, defines the use of the formula in an ATP system. The *formula* follows Prolog conventions, and can additionally include interpreted symbols that start with a `$`, e.g., `$true`, and numbers. The basic logical connectives are `!`, `?`, `~`, `|`, `&`, `=>`, `<=`, `<=>`, and `<~>`. Equality and inequality are expressed as the infix operators `=` and `!=`. The *source* and *useful_info* are optional. Figure 1 is an example of a problem in monomorphic typed first-order form (TF0).



```
%------------------------------------------------------------------------
tff(human_type,type,      human: $tType ).
tff(cat_type,type,        cat: $tType ).
tff(jon_decl,type,        jon: human ).
tff(garfield_decl,type,   garfield: cat ).
tff(arlene_decl,type,     arlene: cat ).
tff(loves_most_decl,type, loves_most: cat > cat ).
tff(owns_decl,type,       owns: ( human * cat ) > $o ).

tff(garfield_not_arlene,axiom,
    garfield != arlene ).

tff(jon_owns_only_garfield,axiom,
    ( owns(jon,garfield) & ~ owns(jon,arlene) ) ).

tff(all_cats_love_garfield,axiom,
    ! [C: cat] : ( loves_most(C) = garfield ) ).

tff(jon_owns_garfields_lovers,conjecture,
    ! [C: cat] :
      ( ( loves_most(C) = garfield ) => owns(jon,C) ) ).
%------------------------------------------------------------------------
```

Figure 1: A TF0 problem (with a finite countermodel)

## 3 Interpretations

A Tarskian-style interpretation (Tarski and Vaught 1956) of formulae in typed first-order logic consists of a non-empty domain of unequal elements for each type used in the formulae (just one domain for untyped logic), and interpretations of the function and predicate symbols with respect to the domains (Hunter 1996). The domains of an interpretation may be finite or infinite. The TPTP representation of an interpretation uses an *interpretation formula*, preceded by the necessary type declarations. *Type-promotion* functions are used to convert domain elements to terms. The interpretation formula is a conjunction specifying:

- for each type in the formulae:
  - the domain type, by a formula that makes the type-promotion function a surjection;
  - the domain elements as a universally quantified disjunction of equalities whose right-hand sides are the domain elements;
  - specification of the distinctness of the domain elements;
  - a formula making the type-promotion function an injection, which with the surjectivity makes it a bijection.
- interpretation of the function symbols, as equalities whose left-hand sides are formed from symbols applied to type-promoted domain elements, and whose right-hand sides

```
%------------------------------------------------------------
tff(equality_lost,interpretation,
%----The domain for human
    ( ( ! [H: human] : ? [DH: d_human] : H = d2human(DH)
      & ! [DH: d_human] : ( DH = d_jon )
      & ! [DH1: d_human,DH2: d_human] :
          ( d2human(DH1) = d2human(DH2) => DH1 = DH2 )
%----The domain for cat
      & ! [C: cat] : ? [DC: d_cat] : C = d2cat(DC)
      & ! [DC: d_cat]: ( DC = d_garfield | DC = d_arlene )
      & $distinct(d_garfield,d_arlene)
      & ! [DC1: d_cat,DC2: d_cat] :
          ( d2cat(DC1) = d2cat(DC2) => DC1 = DC2 ) )
%----Interpret terms and atoms
    & ( jon = d2human(d_jon)
      & garfield = d2cat(d_garfield)
      & arlene = d2cat(d_arlene)
      & loves_most(d2cat(d_garfield)) = d2cat(d_garfield)
      & loves_most(d2cat(d_arlene)) = d2cat(d_garfield) )
    & ( owns(d2human(d_jon),d2cat(d_garfield))
      & ~ owns(d2human(d_jon),d2cat(d_arlene)) ) ) ).
%------------------------------------------------------------
```

Figure 2: A TF0 interpretation with a finite domain
https://raw.githubusercontent.com/GeoffsPapers/ModelVerification/main/TFF_Finite.s

are type-promoted domain elements;

- interpretation of the predicate symbols, as literals formed from symbols applied to type-promoted domain elements; positive literals are *true* and negative literals are *false*.

This representation is also directly usable for untyped first-order logic, where all terms in the given and interpretation formulae are of the same type – "individuals". This obviates the need for type considerations, in particular type-promotion functions are not needed.

Figure 2 is a TF0 interpretation with finite domains – it is a countermodel for the problem in Figure 1. The comments show which parts of the formula specify what aspects of the interpretation, per the list above.

## 4 Interpretation Visualization

Proof visualization is well-established, with several tools available, including the Interactive Derivation Viewer (Trac, Puzis, and Sutcliffe 2007) (IDV) – a tool for visualization of TPTP format proofs. Interpretation visualization, however, has (to the knowledge of the authors) had minimal attention. A visualization for TF0 interpretations has been designed in this work, and an initial implementation is available as the IIV tool in the SystemOnTSTP web interface. IIV is built on top of IDV, and has benefited from the mature state of IDV. The implementation is "initial" because it is fully automated for only finite TF0 interpretations; for infinite interpretations different components of the interpretation formula have to be manually extracted into separate annotated formulae, to mimic a derivation that IDV can render.

Figure 3 is the visualization of the finite countermodel in Figure 2, modified so that `john` is not created equal to the person who got an `A`. The top row of inverted triangles are the types in the given formulae, while the bottom row of inverted triangles are the types of the domains in the interpretation formula. The inverted houses are the function and predicate symbols, and the successive rows of ovals are the successive domain element arguments used to specify the symbols' interpretation. Finally, the row of houses and boxes are the interpretations of the symbols applied to those arguments; houses for functions and boxes for predicates. For example, in the given formulae the type of `grade_of` is `grade`, and `grade_of(d_john)` is interpreted as `d_f`, which is of type `d_grade` in the interpretation formula.

IIV provides some interactive features: Figure 3 shows the situation with the cursor hovering over the lower `d_john` node on the path from `created_equal` to `$true`, showing that `created_equal(d_john,d_john)` is interpreted as `$true`. The nodes above are increasingly darker red (grey if printed) up to the type node `$o` that is the result type of `created_equal`, and increasingly darker blue down to the type node `$o` that is the type of `$true`. This highlighting provides easy focus on the interpretations of chosen symbols. This visualization is available in IIV using `https://raw.githubusercontent.com/GeoffsPapers/ModelVerification/main/TFF_Finite.s` as the "URL to fetch from".

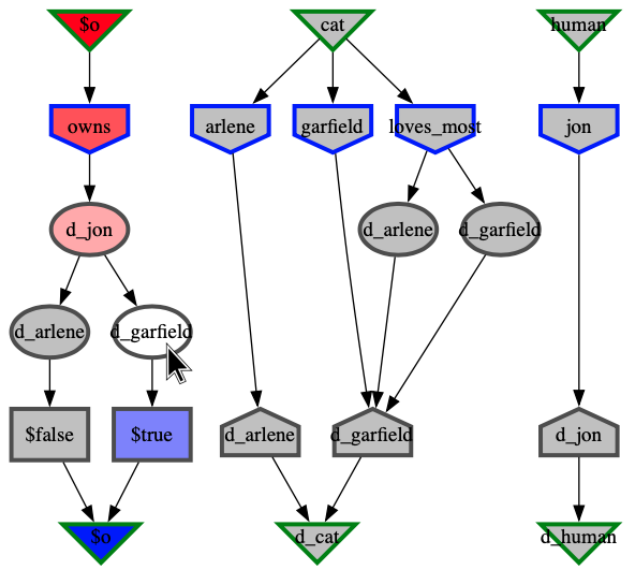


Figure 3: Visualization of the interpretation in Figure 2

Further inspiration might lead to improvements to these visualizations, especially for more complex infinite interpretations.

## 5 Conclusion

This poster describes

## References

Alrabbaa, C.; Baader, F.; Borgwardt, S.; Dachselt, R.; Koopmann, P.; and Méndez, J. 2022. Evonne: Interactive Proof Visualization for Description Logics (System Description). In Blanchette, J.; Kovacs, L.; and Pattinson, D., eds., *Proceedings of the 11th International Joint Conference on Automated Reasoning*, number 13385 in Lecture Notes in Artificial Intelligence, 271–280.

Bachmair, L.; Ganzinger, H.; McAllester, D.; and Lynch, C. 2001. Resolution Theorem Proving. In Robinson, A., and Voronkov, A., eds., *Handbook of Automated Reasoning*. Elsevier Science. 19–99.

Battel, C. 2018. Treehehe: An interactive visualization of proof trees. https://github.com/seachel/treehehe.

D'Silva, V.; Kroening, D.; and Weissenbacher, G. 2008. A Survey of Automated Techniques for Formal Software Verification. *IEEE Transactions on Computer-aided Design of Integrated Circuits and Systems* 27(7):1165–1178.

Herbrand, J. 1930. Recherches sur la Théorie de la Démonstration. *Travaux de la Société des Sciences et des Lettres de Varsovie, Class III, Sciences Mathématiques et Physiques* 33.

Hunter, G. 1996. *Metalogic: An Introduction to the Metatheory of Standard First Order Logic*. University of California Press.

McCune, W. 2003. Mace4 Reference Manual and Guide. Technical Report ANL/MCS-TM-264, Argonne National Laboratory, Argonne, USA.

Robinson, A., and Voronkov, A. 2001. *Handbook of Automated Reasoning*. Elsevier Science.

Schlyter, C. 2013. Visualization of a Finite First Order Logic Model. Master's thesis, Department of Computer Science and Engineering, University of Gothenburg, Göteborg, Sweden.

Sutcliffe, G.; Schulz, S.; Claessen, K.; and Van Gelder, A. 2006. Using the TPTP Language for Writing Derivations and Finite Interpretations. In Furbach, U., and Shankar, N., eds., *Proceedings of the 3rd International Joint Conference on Automated Reasoning*, number 4130 in Lecture Notes in Artificial Intelligence, 67–81. Springer.

Sutcliffe, G.; Schulz, S.; Claessen, K.; and Baumgartner, P. 2012. The TPTP Typed First-order Form with Arithmetic. In Bjørner, N., and Voronkov, A., eds., *Proceedings of the 18th International Conference on Logic for Programming, Artificial Intelligence, and Reasoning*, number 7180 in Lecture Notes in Artificial Intelligence, 406–419. Springer-Verlag.

Sutcliffe, G. 2008. The SZS Ontologies for Automated Reasoning Software. In Sutcliffe, G.; Rudnicki, P.; Schmidt, R.; Konev, B.; and Schulz, S., eds., *Proceedings of the LPAR Workshops: Knowledge Exchange: Automated Provers and Proof Assistants, and the 7th International Workshop on the Implementation of Logics*, number 418 in CEUR Workshop Proceedings, 38–49.

Sutcliffe, G. 2017. The TPTP Problem Library and Associated Infrastructure. From CNF to TH0, TPTP v6.4.0. *Journal of Automated Reasoning* 59(4):483–502.

Sutcliffe, G. 2022. The Logic Languages of the TPTP World. *Logic Journal of the IGPL* https://doi.org/10.1093/jigpal/jzac068.

Tarski, A., and Vaught, R. 1956. Arithmetical Extensions of Relational Systems. *Compositio Mathematica* 13:81–102.

Tews, H. 2017. Proof tree visualization for proof general. http://askra.de/software/prooftree/.

Trac, S.; Puzis, Y.; and Sutcliffe, G. 2007. An Interactive Derivation Viewer. In Autexier, S., and Benzmüller, C., eds., *Proceedings of the 7th Workshop on User Interfaces for Theorem Provers*, volume 174 of *Electronic Notes in Theoretical Computer Science*, 109–123.