The Interactive Interpretation Viewer

Anon One
Some where
Some place
Some country

Anon Two
Some where
Some place
Some country

1 Introduction

Historically, 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 (to 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 is achieved by finding an *interpretation* – a structure that maps terms to domain elements and formulae to truth values, that is a *model* of the axioms – it maps the axioms to *true*, and a *countermodel* of the conjecture – it maps the conjecture to *false* (or equivalently, it is a model of the negated conjecture). 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 (Steen et al. 2022).

2 The TPTP World and Languages

The TPTP World (Sutcliffe 2017) is a well established infrastructure that supports research, development, and deployment of 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), `tcf` (typed clause normal form), `fof` (first-order form), `tff` (typed first-order form), and `thf` (typed higher-order form). The *role*, e.g., `type`, `axiom`, `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 its boolean type `$o`. The logical connectives are `!`, `?`, `~`, `|`, `&`, `=>`, `<=`, `<=>`, and `<~>`. for $\forall$, $\exists$, $\neg$, $\vee$, $\wedge$, $\Rightarrow$, $\Leftarrow$, $\Leftrightarrow$, and $\oplus$ respectively. 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

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 in the formulae (just one domain for untyped logic), and interpretations of the function and predicate symbols with respect to the domains (Hunter 1996). Interpretations with only finite domains are called *finite interpretations*, and interpretations with one of more infinite domains are called *infinite interpretations*. The domains of an interpretation may be finite or infinite, but only finite domains are discussed in this work. Finite domains are commonly explicitly enumerated, but can also take other forms, e.g., the finite Herbrand Universe of a Herbrand interpretation (**?**). The TPTP representation of an interpretation uses an *interpretation formula*, preceded by the necessary type declarations. The interpretation formula is a conjunction providing details of the domains - their types and elements, and the interpretation of the symbols applied to domain elements. Type-promotion functions are used to convert domain elements to terms, to make the interpretation formula well-typed. The representation is also directly usable for untyped first-order logic, where all terms in both 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 type declarations have been omited, and can be seen in URL provided.

```
%----------------------------------------------------------
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/IIVPoster/main/TFF_Finite.s

4 Interpretation Visualization

Proof visualization is well-established, with several tools available, including the Interactive Derivation Viewer (IDV) (Trac, Puzis, and Sutcliffe 2007) – a tool for visualizing 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.

Figure 3 is the visualization of the finite countermodel in Figure 2. 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 in the specification of the symbols' interpretations. Finally, the row of houses and boxes are the interpretations of the symbols applied to those arguments; houses for functions and boxes for predicates. Paths from leaf nodes to root nodes show the interpretations of the symbols applied to the domain elements. For example, in the given formulae the type of loves_most is cat, and loves_most(d_arlene) is interpreted as d_garfield, which is of type d_cat in the interpretation formula.

IIV provides some interactive features: Figure 3 shows the situation with the cursor hovering over the d_garfield node on the path from owns to $true, showing that own(d_jon,d_garfield') 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 owns, 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, e.g., it is easy to highlight what symbol applications are interpreted as $true or $false, or how a specific function symbol is interpreted (here, but hovering over the $loves_most node). This visualization is available in IIV using the URL in the image caption as the "URL to fetch from", selecting IIV 0.0 as the "System", and clicking the "Process Solution" button.

5 IIV Implementation

IIV is implemented on top of IDV, and has benefited from the mature state of IDV. Different terms in an interpretation, e.g., problem types, problem symbols, domain elements, domain types, are extracted into annotated formulae with different languages and roles

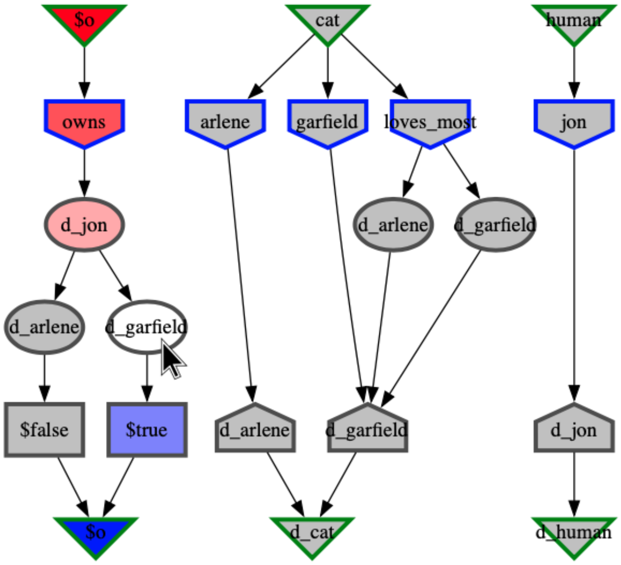

Figure 3: Visualization of the interpretation in Figure 2

that IDV renders differently, e.g., problem symbols are extracted into thf negated_conjectures that IDV renders as inverted houses with a blue outline. A "derivation" is formed by setting the "inference" parents of each node appropriately, which IDV renders as the directed links between nodes.

A Prolog program is used ito extract the components of the interpretation formulae into separate derivation annotated formulae. In order to separate nodes for different kinds of terms into different layers, a level term is added to the source field of each derivation annotated formula, which is used by IDV only when rendering interpretations. An ANTLR parser is used read the extracted annotated formulae, and JavaScript is used to convert the resultant data structure into GraphViz (Ellson et al. 2002) specifications. The GraphViz specification is rendered in the browser by WHAT MAGIC SOME HTML HERE? JACK, HELP. PLEASE CHECK, CORRECT, EXPAND.

6 Future Work

Further inspiration might lead to improvements to IDV, especially for infinite interpretations and Kripke interpretations (Kripke 1963) for non-classical logics.

References

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.

Ellson, J.; Gansner, E.; Koutsofios, L.; North, S.; and Woodhull, G. 2002. Graphviz - Open Source Graph Drawing Tools. In Mutzel, P.; Jünger, M.; and Leipert, S., eds., *Proceedings of Graph Drawing : the 9th International Symposium*, number 2265 in Lecture Notes in Computer Science, 483–484. Springer-Verlag.

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

Kripke, S. 1963. Semantical Considerations on Modal Logic. *Acta Philosophica Fennica* 16:83–94.

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

Steen, A.; Sutcliffe, G.; Fontaine, P.; ; and McKeown, J. 2022. Representation, Verification, and Visualization of Tarskian Interpretations for Typed First-order Logic. In Keshtkar, F., and Bell, E., eds., *Proceedings of the 36th International FLAIRS Conference*, Submitted. Florida Online Journals.

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. 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.

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.