Arithmetic and Inference in a Large Theory

Adam Pease

Infosys, Foothill Research Center, Palo Alto, CA, USA
adam.pease@infosys.com

1 Abstract

Imagine a voice-enabled household robot that can pick up objects and transfer them from room to room. The owner might say "Pick up my red cart and put it in the garage." but the robot might hear either "Pick up my red car and put it in the garage." or the original sentence. A reasoning system might disambiguate the utterance or correct it by reasoning that a typical car weights 1.5 tons and that's 3000 pounds and the carrying capacity of the robot is 100 pounds and the alternate text must be what was said. A system must construct an answer to a question that has never been asked before ("Can the robot carry a car?") perform some simple computation involving unit conversions, to understand that 3000lbs >100lbs and possibly explain its answer if required ("Why didn't you move my car like I asked you to?")

General purpose first-order theorem provers historically haven't done proofs with arithmetic. A relatively new language that addresses sound arithmetic calculation in a first order logic is called Typed First-order Form (TFF) [7]. It is implemented in several of the best modern, first-order provers, including Vampire [2].

To have useful and non-trivial reasoning about, for example, a robot's capabilities, or to do question answering, we need not only a language capable of arithmetic calculation (as well as first order logical reasoning) but also a non-trivial body of axioms that has the information about the real world needed to form answers to such questions. For that reason, we need to use the Suggested Upper Merged Ontology (SUMO) [3, 4], which is a comprehensive and diverse set of logical statements about the world. At approximately 20,000 concepts and 80,000 logical statements (as well as including the large factbase of YAGO [1] and other such resources), it is large enough to answer interesting questions about a wide range of topics. Writing a translator from SUMO's native formalization into TFF should open up many new opportunities for doing practical automated reasoning involving numbers and arithmetic.

In earlier work, we described [5] how to translate SUMO to the strictly first order language of TPTP [6]. SUMO has an extensive type structure and all relations have type restrictions on their arguments. Translation to TPTP involved (among many other steps) implementing a sorted (typed) logic axiomatically in TPTP by altering all implications in SUMO to contain type restrictions on any variables that appear. Note also that all the strictly higher-order content in SUMO is lost in translation to first-order, whether TPTP or TFF.

Like TPTP, TFF forms are valid Prolog syntax (although obviously not the same semantics!) TFF has five disjoint *sorts*: integers, real numbers, rational numbers, booleans and everything else. These are respectively called \$int, \$real, \$rat, \$o and \$i in TFF syntax. Each variable that is used in a logical statement must be declared to be one of these sorts, or by default it will be assumed to be type \$i.

TFF has built in to the language the basic arithmetic functions and arithmetic comparison operators. Each function and operator is *polymorphic* - it is actually a set of three different operators that can handle integers, rationals and reals. Equality is also defined for \$o and \$i.

TFF's creators have planned to include the ability to define subtypes (subsorts) but this is not yet defined in specification or implemented in any prover. An additional issue is that since

all types are disjoint, and SUMO allows multiple inheritance, there is a mismatch between the two type systems. So we have to continue to implement much of SUMO's sort system axiomatically in TFF as in TPTP, but have a special treatment of integers, rationals and reals that does use the TFF type system, so we can use its arithmetic and comparison operators.

We also need to handle SUMO's subsorts of `Integer`, `RealNumber` and `RationalNumber`. This entails adding the constraints that specify these types to any axiom that uses them. For example, `PositiveInteger` has a constraint that it's simply a TFF $int that must be greater than 0.

Because $int and $real are disjoint sorts in TFF, but `Integer` is a subtype of `RealNumber` in SUMO we have to commit to one or the other when there is ambiguity, as in the case of a number appearing without a decimal.

Lastly, all of these types can interact, requiring constraint propagation within an axiom. For example, if we have addition between an `Integer` and a variable that is otherwise constrained only to `Number`, the `Number` will have to be constrained to an `Integer`. An example translation with such an issue is shown in Listing 1.

In the talk for this paper, I present examples of the required transformations and discussion and examples about inferences with the resulting TFF theory translated from SUMO.

```
(=>                                        ! [QUAKE : $i, VALUE : $real] :
  (measure ?QUAKE                            (instance(QUAKE,Object) =>
    (MeasureFn ?VALUE RichterMagnitude))       (measure(QUAKE,
  (instance ?VALUE PositiveRealNumber))           MeasureFn(VALUE,RichterMagnitude)) =>
                                               $greater(VALUE,0)))
```

Listing 1: SUO-KIF to TFF with sorts as conditions, built-in TFF types and comparison operator

References

[1] Gerard de Melo, Fabian Suchanek, and Adam Pease. Integrating YAGO into the Suggested Upper Merged Ontology. *Proceedings of the 20th IEEE International Conference on Tools with Artificial Intelligence*, 2008.

[2] Laura Kovács and Andrei Voronkov. First-order theorem proving and vampire. In *Proceedings of the 25th International Conference on Computer Aided Verification*, volume 8044 of *CAV 2013*, pages 1–35, New York, NY, USA, 2013. Springer-Verlag New York, Inc.

[3] Ian Niles and Adam Pease. Toward a Standard Upper Ontology. In Chris Welty and Barry Smith, editors, *Proceedings of the 2nd International Conference on Formal Ontology in Information Systems (FOIS-2001)*, pages 2–9, 2001.

[4] Adam Pease. *Ontology: A Practical Guide.* Articulate Software Press, Angwin, CA, 2011.

[5] Adam Pease and Stephan Schulz. Knowledge Engineering for Large Ontologies with Sigma KEE 3.0. In *The International Joint Conference on Automated Reasoning*, 2014.

[6] Geoff Sutcliffe. TPTP, TSTP, CASC, etc. In *Proceedings of the Second International Conference on Computer Science: Theory and Applications*, CSR'07, pages 6–22, Berlin, Heidelberg, 2007. Springer-Verlag.

[7] Geoff Sutcliffe, Stephan Schulz, Koen Claessen, and Peter Baumgartner. The TPTP Typed First-order Form with Arithmetic. In *International Conference on Logic for Programming Artificial Intelligence and Reasoning (LPAR 2012)*, pages 406–419, 2012.