Embedding SUMO into Set Theory [*]

Chad Brown[1], Adam Pease[2], and Josef Urban[1]

[1] Czech Technical University in Prague, Czech Republic
[2] Articulate Software, San Jose, CA, USA

The Suggested Upper Merged Ontology (SUMO) [13, 14] is a comprehensive ontology of around 20,000 concepts and 80,000 hand-authored logical statements in a higher-order logic, that has an associated integrated development environment called Sigma [16][1] that interfaces to leading theorem provers such as Eprover [19] and Vampire [12]. In previous work on translating SUMO to THF [2] a syntactic translation to THF was created but did not resolve many aspects of the intended higher order semantics of SUMO. It is our objective in our current efforts to lay the groundwork for a new translation to TH0, based on expressing SUMO in set theory. We believe this will attach to SUMO a stronger set-theoretical interpretation that will allow deciding more queries and provide better intuition for avoiding contradictory formalizations. Once this is done, our plan is to train ENIGMA-style [5–8] query answering and contradiction-finding [20] AITP systems on such SUMO problems and develop autoformalization [9–11, 23, 24] methods targeting common-sense reasoning based on SUMO.

In earlier work, we described [16] how to translate SUMO to the strictly first order language of TPTP-FOF [18] and TF0 [15, 22]. SUMO has an extensive type structure and all relations have type restrictions on their arguments. Translation to TPTP FOF involved implementing a sorted (typed) logic axiomatically in TPTP by altering all implications in SUMO to contain type restrictions on any variables that appear.

We give a grammar for the fragment of the SUMO language in the domain of our translation of SUMO. There are some aspects of SUMO that do not fall into this grammar – namely formulas with modal or probabilistic operators. We have ordinary variables (x), row variables (ρ) and constants (c). We mutually define the sets of terms t, spines s and formulas ψ as follows:

$$t ::= x|c|x\ s|c\ s|(\kappa x.\psi)$$

$$s ::= t\ s|\cdot|\rho$$

$$\psi ::= \bot|\top|(\neg\psi)|(\psi \to \psi)|(\psi \wedge \psi)|(\psi \vee \psi)|(\psi \leftrightarrow \psi)|(\forall x.\psi)|(\exists x.\psi)|(t = t)|c\ s$$

The definition is mutually recursive since the term $\kappa x.\psi$ depends on the formula ψ. Of course, κ, $\forall$ and $\exists$ are binders.

Properly parsing SUMO terms and formulas requires mechanisms for infering implicit type guards for variables (interpreted conjunctively for κ and $\exists$ and via implication for $\forall$). Free variables in SUMO assertions are implicitly universally quantified. For simplicity, we assume all type guards and implicit quantifiers have already been inferred (as in [16]) before beginning the translation.

Our translation maps terms t and spines s to sets and formulas ψ to set theoretic propositions. The particular set theory we use is higher-order Tarski-Grothendieck as described in [4]. For simplicity we assume SUMO variables (both ordinary and row) are also set theoretic variables ranging over sets. We likewise assume all SUMO constants are also set theoretic constants (with the exception of instance and subclass, described below). To translate spines, we need

[*]Supported by by the Czech project AI&Reasoning CZ.02.1.01/0.0/0.0/15_003/ 0000466 and the European Regional Development Fund.

[1]https://www.ontologyportal.org

a way to form lists as sets. We do this generically by simply assuming a set nil, an operator cons taking two sets to a set (meant to be the cons pair) and an operator $\mathsf{listprod}$ taking a two sets to a set (where $\mathsf{listprod}\ A\ B$ is meant to be the set of cons pairs $\mathsf{cons}\ a\ b$ where $a \in A$ and $b \in B$). We also assume a special set U, which will act as the universe of elements that may be members of classes. The universe is assumed to be closed under set theoretic function application and the list operators mentioned above.

A major commitment of our translation is that, if t_i are SUMO terms translated to sets t'_i, then SUMO formulas of the form $\mathsf{instance}\ t_1\ t_2$ will translate to $t'_1 \in t'_2$ and SUMO formulas of the form $\mathsf{subclass}\ t_1\ t_2$ will translate to $t'_1 \subseteq t'_2$.

We isolate a few special cases in SUMO: Class, $\mathsf{SetOrClass}$, $\mathsf{Abstract}$ and Entity. These are considered *classes* in SUMO, but will be considered *superclasses* in our translation. In particular, the interpretation of Class will be the $\wp\mathsf{U}$, power set of U. Hence every member of the interpretation of Class will be a subset of U. Since SUMO declares $\mathsf{SetOrClass}$, $\mathsf{Abstract}$ and Entity to contain Class, none of these four superclasses can be a member of $\wp\mathsf{U}$. Whenever SUMO globally declares t_1 to be a subclass of t_2, we also declare the corresponding sets t'_1 and t'_2 to be members of $\wp\mathsf{U}$, except in the four special superclass cases.

The translation of spines is the obvious one: we use nil for the empty spine, cons for a term followed by a spine, and the same variable ρ for row variables. For terms, we translate x and c directly, since we assumed these are variables and constants in the set theory. We translate $x\ s$ and $c\ s$ by using set theoretic function application (where the translation of the spine s is the argument). We translate $\kappa x.\psi$ as $\{x \in \mathsf{U} | \psi'\}$ where ψ' is the translation of ψ. Translation of formulas proceeds in the obvious way, with only the $c\ s$ case being noteworthy. Note that $c\ s$ is both a term and a formula. Interpreting $c\ s$ as a term gives a set b and we translate $c\ s$ to the formula $0 \in b$. The idea is that b will be either 0 or 1, with $0 \in b$ being false if b is 0 and true if b is 1.

Future work While we have an incomplete translation of all the elements of SUMO that are beyond syntactic first order expressions, we now have a basis for mapping SUMO into set theory. The current translation addresses the κ binder, a term level binder one cannot represent in standard first-order logic. In addition, we can translate row variables directly as sets, although we expect more work is needed to create proper bounds for quantifiers of row variables. In addition a future translation should account for temporal `holdsDuring` and modal operators (including `modalAttribute`, `confersRight` etc). We expect this to be the next stage of our efforts.

An initial use the mapping will be to have a type-checker that gives immediate feedback to SUMO developers on one aspect of the correctness of their higher-order axioms, in the same way that the TPTP FOF and TF0 translations provide feedback on the correct use of types in the FOF portion of SUMO.

We have some inference tests for SUMO[2] for TPTP FOF and TF0. We will expand this corpus to include tests expressible in the new TH0 translation. This will serve as a regression suite for SUMO's higher-order content, as well as a source for testing the performance of HOL provers, such as Satallax [3], Zipperposition [1] and LEO-III [21].

It is possible that set theory provides a model for (some part or all of) SUMO after the translation. Proving that at least portions of SUMO have a model would be a much stronger statement than current method that only allow us to say that no contradictions have been found with first order theorem proving within a large set of tests and a generous time bound [17].

[2]https://github.com/ontologyportal/sumo/tree/master/tests/TPTP

References

[1] Alexander Bentkamp, Jasmin Christian Blanchette, Simon Cruanes, and Uwe Waldmann. Superposition for lambda-free higher-order logic. *Logical Methods in Computer Science*, 17(2):1:1–1:38, April 2021.

[2] Christoph Benzmüller and Adam Pease. Higher-Order Aspects and Context in SUMO. In Ivan José Varzinczak Jos Lehmann and Alan Bundy, editors, *Special issue on Reasoning with context in the Semantic Web*, volume 12-13. Science, Services and Agents on the World Wide Web, 2012.

[3] Chad E. Brown. Satallax: An automatic higher-order prover. In Bernhard Gramlich, Dale Miller, and Uli Sattler, editors, *IJCAR*, volume 7364 of *LNCS*, pages 111–117. Springer, 2012.

[4] Chad E. Brown and Karol Pąk. A tale of two set theories. In Cezary Kaliszyk, Edwin C. Brady, Andrea Kohlhase, and Claudio Sacerdoti Coen, editors, *Intelligent Computer Mathematics - 12th International Conference, CICM 2019, Prague, Czech Republic, July 8-12, 2019, Proceedings*, volume 11617 of *Lecture Notes in Computer Science*, pages 44–60. Springer, 2019.

[5] Karel Chvalovský, Jan Jakubuv, Martin Suda, and Josef Urban. ENIGMA-NG: efficient neural and gradient-boosted inference guidance for E. In Pascal Fontaine, editor, *Automated Deduction - CADE 27 - 27th International Conference on Automated Deduction, Natal, Brazil, August 27-30, 2019, Proceedings*, volume 11716 of *Lecture Notes in Computer Science*, pages 197–215. Springer, 2019.

[6] Jan Jakubuv, Karel Chvalovský, Miroslav Olšák, Bartosz Piotrowski, Martin Suda, and Josef Urban. ENIGMA anonymous: Symbol-independent inference guiding machine (system description). In Nicolas Peltier and Viorica Sofronie-Stokkermans, editors, *Automated Reasoning - 10th International Joint Conference, IJCAR 2020, Paris, France, July 1-4, 2020, Proceedings, Part II*, volume 12167 of *Lecture Notes in Computer Science*, pages 448–463. Springer, 2020.

[7] Jan Jakubuv and Josef Urban. ENIGMA: efficient learning-based inference guiding machine. In Herman Geuvers, Matthew England, Osman Hasan, Florian Rabe, and Olaf Teschke, editors, *Intelligent Computer Mathematics - 10th International Conference, CICM 2017, Edinburgh, UK, July 17-21, 2017, Proceedings*, volume 10383 of *Lecture Notes in Computer Science*, pages 292–302. Springer, 2017.

[8] Jan Jakubuv and Josef Urban. Hammering Mizar by learning clause guidance. In John Harrison, John O'Leary, and Andrew Tolmach, editors, *10th International Conference on Interactive Theorem Proving, ITP 2019, September 9-12, 2019, Portland, OR, USA*, volume 141 of *LIPIcs*, pages 34:1–34:8. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019.

[9] Cezary Kaliszyk, Josef Urban, and Jirí Vyskocil. Automating formalization by statistical and semantic parsing of mathematics. In *ITP*, volume 10499 of *Lecture Notes in Computer Science*, pages 12–27. Springer, 2017.

[10] Cezary Kaliszyk, Josef Urban, and Jiří Vyskočil. Learning to parse on aligned corpora (rough diamond). In Christian Urban and Xingyuan Zhang, editors, *Interactive Theorem Proving - 6th International Conference, ITP 2015, Nanjing, China, August 24-27, 2015, Proceedings*, volume 9236 of *Lecture Notes in Computer Science*, pages 227–233. Springer, 2015.

[11] Cezary Kaliszyk, Josef Urban, Jiří Vyskočil, and Herman Geuvers. Developing corpus-based translation methods between informal and formal mathematics: Project description. In Stephen M. Watt, James H. Davenport, Alan P. Sexton, Petr Sojka, and Josef Urban, editors, *Intelligent Computer Mathematics - International Conference, CICM 2014, Coimbra, Portugal, July 7-11, 2014. Proceedings*, volume 8543 of *LNCS*, pages 435–439. Springer, 2014.

[12] 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.

[13] 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.

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

[15] Adam Pease. Arithmetic and inference in a large theory. In *AI in Theorem Proving*, 2019.

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

[17] Adam Pease and Stephan Schulz. Contradiction detection and repair in a large theory. In *The International FLAIRS Conference Proceedings*, 2022.

[18] Adam Pease, Geoff Sutcliffe, Nick Siegel, and Steven Trac. Large Theory Reasoning with SUMO at CASC. *AI Communications, Special issue on Practical Aspects of Automated Reasoning*, 23(2-3):137–144, 2010.

[19] Stephan Schulz. E - A Brainiac Theorem Prover. *AI Commun.*, 15(2-3):111–126, 2002.

[20] Stephan Schulz, Geoff Sutcliffe, Josef Urban, and Adam Pease. Detecting inconsistencies in large first-order knowledge bases. In *Proceedings of CADE 26*, pages 310–325. Springer, 2017.

[21] Alexander Steen and Christoph Benzmüller. The higher-order prover leo-iii. *CoRR*, abs/1802.02732, 2018.

[22] 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.

[23] Qingxiang Wang, Chad E. Brown, Cezary Kaliszyk, and Josef Urban. Exploration of neural machine translation in autoformalization of mathematics in mizar. In *CPP*, pages 85–98. ACM, 2020.

[24] Qingxiang Wang, Cezary Kaliszyk, and Josef Urban. First experiments with neural translation of informal to formal mathematics. In *CICM*, volume 11006 of *Lecture Notes in Computer Science*, pages 255–270. Springer, 2018.

1 Appendix

Examples We briefly consider two first-order example queries and three queries involving κ. Queries differ from assertions in that their free variables are implicitly existentially quantified, with implicit type guards added conjunctively. We prove the translated query in the Megalodon interactive prover (the successor to the Egal system [4]).

Oure first example is given by the SUMO query:

```
(instance Org1-1 Organization)
(query (member ?MEMBER Org1-1))
```

This translates into Megalodon as follows:

```
Variable s_ORG1_x2D1:set.
Hypothesis p5315: (s_ORG1_x2D1 :e s_ORGANIZATION).
Theorem p5316: exists v_MEMBER, v_MEMBER :e s_PHYSICAL
    /\ (bp (s_MEMBER (cons v_MEMBER (cons s_ORG1_x2D1 nil)))).
```

The (interactively constructed) proof makes use of (translated) SUMO assertions that all collections have a physical object member and that organizations are collections.

Our second example is given by the SUMO query:

```
(=>
  (and
    (instance ?A Animal)
    (not
      (exists (?PART)
        (and
          (instance ?PART SpinalColumn)
          (part ?PART ?A)))))
  (not
    (instance ?A Vertebrate)))

(not
  (exists (?SPINE)
    (and
      (instance ?SPINE SpinalColumn)
      (part ?SPINE BananaSlug10-1))))

(instance BananaSlug10-1 Animal)

(and
  (instance BodyPart10-1 BodyPart)
  (component BodyPart10-1 BananaSlug10-1))

(query (instance BananaSlug10-1 Invertebrate))
```

This translates into the following Megalodon formalization:

```
Variable s_SPINALCOLUMN:set.
Hypothesis p5320: forall v_A, v_A :e s_ENTITY -> v_A :e s_OBJECT ->
 ((v_A :e s_ANIMAL)
   /\ (~ (exists v_PART, v_PART :e s_ENTITY /\ v_PART :e s_OBJECT
            /\ (v_PART :e s_SPINALCOLUMN) /\ (bp (s_PART (cons v_PART (cons v_A nil))))))
   -> (~ (v_A :e s_VERTEBRATE))).
Variable s_BANANASLUG10_x2D1:set.
Hypothesis p5321: (~ (exists v_SPINE, v_SPINE :e s_ENTITY /\ v_SPINE :e s_OBJECT
     /\ (v_SPINE :e s_SPINALCOLUMN) /\ (bp (s_PART (cons v_SPINE (cons s_BANANASLUG10_x2D1 nil)))))).
Hypothesis p5322: (s_BANANASLUG10_x2D1 :e s_ANIMAL).
Variable s_BODYPART10_x2D1:set.
Hypothesis p5323: (s_BODYPART10_x2D1 :e s_BODYPART)
  /\ (bp (s_COMPONENT (cons s_BODYPART10_x2D1 (cons s_BANANASLUG10_x2D1 nil)))).
Theorem p5324: (s_BANANASLUG10_x2D1 :e s_INVERTEBRATE).
```

The proof uses the translation of the SUMO assertion that the classes of vertebrates and invertebrates form a partition of the class of animals.

Our κ examples are all variants of the same idea, and all are easily provable. The first κ example query is given in SUMO as follows:

```
(query (forall (?V) (=> (instance ?V Atom)
        (forall (?E) (=> (instance ?E Electron)
  (=> (part ?E ?V)
      (instance ?E (KappaFn ?x (and (part ?x ?V) (instance ?x Electron))))))))))
```

This translates to the following Megalodon formalization:

```
Theorem p5326: (forall v_V, v_V :e s_ENTITY -> v_V :e s_OBJECT -> ((v_V :e s_ATOM)
 -> (forall v_E, v_E :e s_ENTITY -> v_E :e s_OBJECT -> ((v_E :e s_ELECTRON)
 -> ((bp (s_PART (cons v_E (cons v_V nil))))
 -> (v_E :e {v_X :e Univ1 | v_X :e s_OBJECT /\ v_X :e s_ENTITY
      /\ (bp (s_PART (cons v_X (cons v_V nil)))) /\ (v_X :e s_ELECTRON)})))))).
```

The second κ example query is given in SUMO as follows:

```
(query (forall (?V) (=> (instance ?V Atom)
        (forall (?E) (=> (instance ?E Electron)
  (=> (instance ?E (KappaFn ?x (and (part ?x ?V) (instance ?x Electron))))
      (part ?E ?V)))))))
```

This translates to the following Megalodon formalization:

```
Theorem p5327: (forall v_V, v_V :e s_ENTITY -> v_V :e s_OBJECT -> ((v_V :e s_ATOM) ->
 (forall v_E, v_E :e s_ENTITY -> v_E :e s_OBJECT -> ((v_E :e s_ELECTRON)
 -> ((v_E :e {v_X :e Univ1 | v_X :e s_OBJECT /\ v_X :e s_ENTITY
         /\ (bp (s_PART (cons v_X (cons v_V nil)))) /\ (v_X :e s_ELECTRON)})
 -> (bp (s_PART (cons v_E (cons v_V nil))))))))).
```

The final κ example does not use κ in the statement, though κ is vital to proving the translated theorem. In SUMO the example is given as follows:

```
(query (forall (?V) (=> (instance ?V Atom)
        (forall (?E) (=> (instance ?E Electron)
  (exists (?C)
    (and (instance ?C Class)
      (<=> (part ?E ?V)
           (instance ?E ?C)))))))))
```

This translates to the following Megalodon formalization:

```
Theorem p5328: (forall v_V, v_V :e s_ENTITY -> v_V :e s_OBJECT -> ((v_V :e s_ATOM) ->
 (forall v_E, v_E :e s_ENTITY -> v_E :e s_OBJECT -> ((v_E :e s_ELECTRON)
 -> (exists v_C, v_C :e s_ENTITY /\ v_C :e s_CLASS /\ (v_C :e s_CLASS)
      /\ ((bp (s_PART (cons v_E (cons v_V nil)))) <-> (v_E :e v_C))))))).
```

Converting SUMO to First Order All the strictly higher-order content in SUMO was previously lost in translation to first-order, whether TPTP or TF0. The translation steps include:

- expanding "row variables" which allow for stating axioms without commitment to the number of arguments a relation has, similar to Lisp's @REST construct
- instantiating "predicate variables" with all possible values. This is needed for any axiom that has a variable in place of a relation.
- expanding the arity of all variable arity relations as set of relations with different names depending upon their fixed number of arguments
- renaming any relations given as arguments to other relations

SUMO has no native implementation in a theorem prover, and has no formal semantics beyond that of standard first order logic, so the process of translating SUMO into a language with a fully specified semantics, such as TPTP_FOF, TF0 or THF gives SUMO its semantics.

Type Mechanisms All relations (including functions) in SUMO have a type signature. As a consequence, we don't need an explicit syntax for types/sorts of variables, and can deduce them automatically. We can have classes as well as instances as arguments. The `domain` and `range` relations are meta-predicates that direct the Sigma translators to state that arguments to a given relation (or the return type of a function, respectively) are instances of a given type. The `domainSubclass`, and `rangeSubclass` relations state that arguments to a given relation (or the return type of a function, respectively) are a given class or one of its subclasses. For example

```
(domain DensityFn 1 MassMeasure)
(domain DensityFn 2 VolumeMeasure)
(instance DensityFn BinaryFunction)
(range DensityFn FunctionQuantity)
```

`DensityFn` is a `BinaryFunction` that takes an instance of a `MassMeasure` and a `VolumeMeasure`, respectively, as its first and second arguments. In

```
(domainSubclass typicalPart 1 Object)
(domainSubclass typicalPart 2 Object)
(instance typicalPart BinaryPredicate)
```

the first and second arguments to the `typicalPart` relation are of the class `Object` or one of its subclasses.

THF Translator Below by *SUMO objects (SOs)* we mean arbitrary SUMO classes and instances.

Our translator maps all SUMO objects to sets in HO TG set theory. The subclass relation is translated as inclusion and the instance relation as membership. SOs that are potentially large such as abstract, mathematical and related SOs thus become sets that may live in higher TG universes.

SOs can be applied to other SOs and variables, creating terms and formulas. Such SOs will be sets that encode relations and functions. Their application to other SOs is the corresponding application of the set theoretical functional and relational sets to other sets. To handle variable arities and row variables, arguments are always appended together into lists.

SUMO quantifiers and logical connectives are mapped directly to their FOL counterparts. Applications that are at predicate positions in formulas are casted by a special *bp* predicate into propositions.

To illustrate a significant higher order construct in SUMO, consider the following problem that uses an axiom with `KappaFn`, which defines a class on the fly, without the need to reify it.

```
(<=>
    (totalFacilityTypeInArea ?AREA ?TYPE ?COUNT)
    (cardinality
        (KappaFn ?ITEM
            (and
                (instance ?ITEM ?TYPE)
                (located ?ITEM ?AREA))) ?COUNT))

(instance DejvickaStation TrainStation)
(located DejvickaStation PragueCzechRepublic)
(instance HradCanskaStation TrainStation)
(located HradCanskaStation PragueCzechRepublic)

Q: (totalFacilityTypeInArea ?AREA ?TYPE ?COUNT)
A: [?AREA=PragueCzechRepublic,?TYPE=TrainStation,?COUNT=2]
```

The first axiom states that for the ternary relation of `totalFacilityTypeInArea`, which related an area, a class of `Object` and a count of those objects within that area, it is equivalent to the cardinality of the instances of the class that are defined to be instances of the same type, and present within a particular `?AREA`.

We should be able to ask what relations are deducable for `totalFacilityTypeInArea` and get the answer that, for this knowledge base, there are two instances of `TrainStation` that are known to be in the `CzechRepublic`.

```
(SLEEPING c= PSYCHOLOGICALPROCESS).
(ASLEEP :e CONSCIOUSNESSATTRIBUTE).
((bp (ATTRIBUTE (cons v_AGENT (cons ASLEEP nil))))
\/ (bp (ATTRIBUTE (cons v_AGENT (cons AWAKE nil))))
-> (bp (ATTRIBUTE (cons v_AGENT (cons LIVING nil))))).
```

are the translations of:

```
(subclass Sleeping PsychologicalProcess)
(instance Asleep ConsciousnessAttribute)
(=>
    (or
        (attribute ?AGENT Asleep)
        (attribute ?AGENT Awake))
    (attribute ?AGENT Living))
```