# Embedding SUMO into Set Theory

Chad E. Brown[1] Adam Pease[2] Josef Urban[1]

Czech Technical University in Prague, Czech Republic

Parallax Research, Beavercreek, OH, USA

September 13, 2023

Supported by the Czech project AI&Reasoning CZ.02.1.01/0.0/0.0/15_003/ 0000466 and the European Regional Development Fund and the Ministry of Education, Youth and Sports within the dedicated program ERC CZ under the project POSTMAN no. LL1902.

# Suggested Upper Merged Ontology (SUMO)

- 20+yrs of effort, 20k terms, 80k statements, some in HOL, many tools
- Is SUMO consistent? Most of the time, as far as we know, at least in FOL
  - Regular testing with E prover's contradiction finder that generates 1000's of tests
  - $\sim$75 test problems in TPTP (x3 variations for different portions of SUMO)
  - Testing with Vampire - GitHub runs Vampire for hours triggered by SUMO upload
  - Simple algorithmic checks for type consistency etc.
  - Is any large software system 100% bug-free? Can still be useful. And contradictions are clear since they don't use the conjecture.
  - Only a few experiments in HOL

https://www.ontologyportal.org

https://github.com/ontologyportal

# SUMO to THF

- Work with Chris Benzmüller from 2010
- Mainly a "syntactic" translation
- Didn't use type guards
- Did expand row variables, variable arity relations
- No possible worlds/Kripke semantics

# HOL Relations

- KappaFn
- ProbabilityFn
- attitudeForFormula
- believes
- causesProposition
- conditionalProbability
- confersNorm
- confersObligation
- confersRight
- considers
- containsFormula
- decreasesLikelihood
- deprivesNorm
- describes
- desires
- disapproves
- doubts
- entails
- expects
- hasPurpose
- hasPurposeForAgent
- holdsDuring
- holdsObligation
- holdsRight
- increasesLikelihood
- independentProbability
- knows
- modalAttribute
- permits
- prefers
- prohibits
- rateDetail
- treatedPageDefinition
- visitorParameter

# SUMO statistics

Knowledge base statistics

| Total Terms | Total Axioms | Total Rules |
|---|---|---|
| 16050 | 228841 | 6957 |

Relations: 1705

Ground tuples: 221799

| | |
|---|---|
| of which are binary: | 152069 |
| of which arity more than binary: | 69815 |

Rules: 6957

| | |
|---|---|
| of which are horn: | 2337 |
| first-order: | 5146 |
| temporal: | 772 |
| modal: | 256 |
| epistemic: | 86 |
| other higher-order: | 804 |

# What did we do and why?

- A set theoretic interpretation get us closer to ensuring consistency.
- We can also take queries and turn them into theorem proving problems.
- We did this with 23 examples, many selected from an earlier published set of 35.
- The translated SUMO axioms and queries become large, complex, and difficult to reason with.
- So we did interactive proofs and ask ATPs to prove subgoals in the proofs.
- What are the ATP results?

| Problem | Subgoals | Zipperposition | Vampire | E | Lash | Leo-III |
|---|---|---|---|---|---|---|
| TQG1 | 50 | 50 (100%) | 50 (100%) | 50 (100%) | 50 (100%) | 50 (100%) |
| TQG3 | 20 | 20 (100%) | 20 (100%) | 14 (70%) | 20 (100%) | 8 (40%) |
| TQG7 | 195 | 188 (96%) | 185 (95%) | 180 (92%) | 160 (82%) | 158 (81%) |
| TQG9 | 19 | 19 (100%) | 19 (100%) | 19 (100%) | 19 (100%) | 19 (100%) |
| TQG10 | 112 | 112 (100%) | 112 (100%) | 100 (89%) | 58 (52%) | 96 (86%) |
| TQG11 | 100 | 76 (76%) | 39 (39%) | 67 (67%) | 45 (45%) | 13 (13%) |
| TQG19 | 37 | 34 (92%) | 22 (59%) | 20 (54%) | 37 (100%) | 11 (30%) |
| TQG20 | 41 | 34 (83%) | 22 (54%) | 20 (49%) | 41 (100%) | 13 (32%) |
| TQG21 | 207 | 154 (74%) | 150 (72%) | 143 (69%) | 101 (49%) | 56 (27%) |
| TQG22alt3 | 319 | 246 (77%) | 214 (67%) | 193 (61%) | 197 (62%) | 136 (43%) |
| TQG22alt4 | 322 | 251 (78%) | 218 (68%) | 197 (61%) | 201 (62%) | 142 (44%) |
| TQG22 | 315 | 271 (86%) | 224 (71%) | 212 (67%) | 201 (64%) | 142 (45%) |
| TQG23 | 67 | 61 (91%) | 67 (100%) | 42 (63%) | 51 (76%) | 38 (57%) |
| TQG25alt1 | 910 | 652 (72%) | 526 (58%) | 580 (64%) | 529 (58%) | 246 (27%) |
| TQG27 | 7 | 7 (100%) | 7 (100%) | 7 (100%) | 7 (100%) | 7 (100%) |
| TQG28alt1 | 600 | 428 (71%) | 386 (64%) | 349 (58%) | 261 (44%) | 213 (36%) |
| TQG30 | 4 | 4 (100%) | 4 (100%) | 3 (75%) | 4 (100%) | 4 (100%) |
| TQG33 | 112 | 82 (73%) | 83 (74%) | 79 (71%) | 85 (76%) | 36 (32%) |
| TQG45 | 162 | 136 (84%) | 131 (81%) | 128 (79%) | 106 (65%) | 36 (22%) |
| TQG46 | 344 | 258 (75%) | 215 (62%) | 225 (65%) | 163 (47%) | 144 (42%) |
| TQG47 | 186 | 141 (76%) | 113 (61%) | 109 (59%) | 93 (50%) | 79 (42%) |
| TQG48 | 336 | 249 (74%) | 234 (70%) | 219 (65%) | 184 (55%) | 146 (43%) |
| wordex | 415 | 315 (76%) | 255 (61%) | 236 (57%) | 284 (68%) | 143 (34%) |
| Total | 4880 | 3788 (78%) | 3296 (68%) | 3192 (65%) | 2897 (59%) | 1936 (40%) |

Table: Number of Subgoals Proven Automatically in 60 seconds

# Some Challenging SUMO Constructs

- Variable arity relations

- Kappa classes
- Row variables and quantified predicates

- Implicit type guards.

# Some Challenging SUMO Constructs

- Variable arity relations
  - Example: `partition`
  - `(partition Animal Vertebrate Invertebrate)`
  - `(partition Organism Animal Plant Fungus Microorganism)`
- Kappa classes
- Row variables and quantified predicates
- Implicit type guards.

# Some Challenging SUMO Constructs

- Variable arity relations
  - Example: `partition`
  - `(partition Animal Vertebrate Invertebrate)`
  - `(partition Organism Animal Plant Fungus Microorganism)`
- Kappa classes
  - `(KappaFn ?X (part ?X ?V))`
- Row variables and quantified predicates

- Implicit type guards.

# Some Challenging SUMO Constructs

- Variable arity relations
  - Example: `partition`
  - `(partition Animal Vertebrate Invertebrate)`
  - `(partition Organism Animal Plant Fungus Microorganism)`
- Kappa classes
  - `(KappaFn ?X (part ?X ?V))`
- Row variables and quantified predicates
  - `(=> (and (subrelation ?REL1 ?REL2) (?REL1 @ROW)) (?REL2 @ROW))`
- Implicit type guards.

# Some Challenging SUMO Constructs

- Variable arity relations
  - Example: `partition`
  - `(partition Animal Vertebrate Invertebrate)`
  - `(partition Organism Animal Plant Fungus Microorganism)`
- Kappa classes
  - `(KappaFn ?X (part ?X ?V))`
- Row variables and quantified predicates
  - `(=> (and (subrelation ?REL1 ?REL2) (?REL1 @ROW)) (?REL2 @ROW))`
- Implicit type guards.
  - In previous: `?REL1` and `?REL2` must `Relations`.
  - and `@ROW` must be a list appropriate for the two particular relations.

# Simple Example

- 3 SUMO assertions:
- The “uses” relation is asymmetric:
- `(instance uses AsymmetricRelation)`

# Simple Example

- 3 SUMO assertions:

- The "uses" relation is asymmetric:
- `(instance uses AsymmetricRelation)`

- Every asymmetric relation is irreflexive:
- `(subclass AsymmetricRelation IrreflexiveRelation)`

# Simple Example

- 3 SUMO assertions:
- The “uses” relation is asymmetric:
- `(instance uses AsymmetricRelation)`
- Every asymmetric relation is irreflexive:
- `(subclass AsymmetricRelation IrreflexiveRelation)`
- What irreflexive means:
- ```
  (=> (instance ?REL IrreflexiveRelation)
      (forall (?INST) (not (?REL ?INST ?INST))))
  ```
  ```

# Simple Example

- 3 SUMO assertions:

- The "uses" relation is asymmetric:
- `(instance uses AsymmetricRelation)`

- Every asymmetric relation is irreflexive:
- `(subclass AsymmetricRelation IrreflexiveRelation)`

- What irreflexive means:
- ```
  (=> (instance ?REL IrreflexiveRelation)
      (forall (?INST) (not (?REL ?INST ?INST))))
  ```

- Idea: instantiate `?REL` with `uses`.
- But it's used as both a constant and binary relation?
  ```

# Simple Example (Partly Translated)

- The 3 SUMO assertions set theoretically:

- `(instance uses AsymmetricRelation)`
- `uses` is interpreted as a set we call USES.
- `AsymmetricRelation` is also interpreted as a set we call ASYMMETRICRELATION.
- Assertion: USES $\in$ ASYMMETRICRELATION.

# Simple Example (Partly Translated)

- The 3 SUMO assertions set theoretically:

- `(instance uses AsymmetricRelation)`
- `uses` is interpreted as a set we call USES.
- `AsymmetricRelation` is also interpreted as a set we call ASYMMETRICRELATION.
- Assertion: USES $\in$ ASYMMETRICRELATION.

- `(subclass AsymmetricRelation IrreflexiveRelation)`
- ASYMMETRICRELATION $\subseteq$ IRREFLEXIVERELATION

# Simple Example (Almost Translated)

- ```
  (=> (instance ?REL IrreflexiveRelation)
      (forall (?INST) (not (?REL ?INST ?INST))))
  ```
- Almost this: $\forall r \in \text{IRREFLEXIVERELATION}.\forall x.\neg r(x, x)$.
  ```

# Simple Example (Almost Translated)

- ```
  (=> (instance ?REL IrreflexiveRelation)
      (forall (?INST) (not (?REL ?INST ?INST))))
  ```

- Almost this: $\forall r \in \text{IRREFLEXIVERELATION}. \forall x. \neg r(x, x)$.
- Note: USES is a set and $r$ quantifies over sets.
  ```

# Simple Example (Almost Translated)

- ```
  (=> (instance ?REL IrreflexiveRelation)
      (forall (?INST) (not (?REL ?INST ?INST))))
  ```

- Almost this: $\forall r \in \mathrm{IRREFLEXIVERELATION}.\forall x.\neg r(x, x)$.
- Note: USES is a set and $r$ quantifies over sets.
- Why *almost*? For a few reasons.
  ```

# Simple Example (Almost Translated)

- ```
  (=> (instance ?REL IrreflexiveRelation)
      (forall (?INST) (not (?REL ?INST ?INST))))
  ```
- Almost this: $\forall r \in \text{IRREFLEXIVERELATION}.\forall x.\neg r(x, x)$.
- Note: USES is a set and $r$ quantifies over sets.
- Why *almost*? For a few reasons.
- One important reason: type guards.
  ```

# Simple Example (Almost Translated)

- ```
  (=> (instance ?REL IrreflexiveRelation)
      (forall (?INST) (not (?REL ?INST ?INST))))
  ```

- Almost this: $\forall r \in \text{IRREFLEXIVERELATION}.\forall x.\neg r(x, x)$.
- Note: USES is a set and $r$ quantifies over sets.
- Why *almost*? For a few reasons.
- One important reason: type guards.
- The SUMO relation `uses` expects
  - an instance of `Object` as first argument and
  - an instance of `Agent` as second argument.
  ```

# Simple Example (Almost Translated)

- ```
  (=> (instance ?REL IrreflexiveRelation)
      (forall (?INST) (not (?REL ?INST ?INST))))
  ```
- Almost this: $\forall r \in \text{IRREFLEXIVERELATION}.\forall x.\neg r(x, x)$.
- Note: USES is a set and $r$ quantifies over sets.
- Why *almost*? For a few reasons.
- One important reason: type guards.
- The SUMO relation `uses` expects
  - an instance of `Object` as first argument and
  - an instance of `Agent` as second argument.
- So irreflexivity of USES should be guarded and say

$$\forall x \in \text{OBJECT} \cap \text{AGENT}.\neg\text{USES}(x, x)$$
  ```

# Simple Example (Almost Translated)

- `(=> (instance ?REL IrreflexiveRelation)`
  `(forall (?INST) (not (?REL ?INST ?INST))))`
- Almost this: $\forall r \in \mathrm{IRREFLEXIVERELATION}.\forall x.\neg r(x,x)$.
- Note: USES is a set and $r$ quantifies over sets.
- Why *almost*? For a few reasons.
- One important reason: type guards.
- The SUMO relation `uses` expects
  - an instance of `Object` as first argument and
  - an instance of `Agent` as second argument.
- So irreflexivity of USES should be guarded and say

$$\forall x \in \mathrm{OBJECT} \cap \mathrm{AGENT}.\neg \mathrm{USES}(x,x)$$

- But how can we know the guards before we instantiate the $r$?

# Simple Example (Type Guards)

- Instead of $\forall r \in \text{IRREFLEXIVERELATION}.\forall x.\neg r(x, x)$
- we almost do this:

$$\forall r \in \text{IRREFLEXIVERELATION}.$$
$$\forall x \in \text{dom}_1(r) \cap \text{dom}_2(r).\text{ap}(r)(x, x)$$

- Here part of the translation asserts the "typing" information:
- $\text{dom}_1(\text{USES}) = \text{OBJECT}$
- $\text{dom}_2(\text{USES}) = \text{AGENT}$
- ap(USES) is the function taking a pair to a boolean.
- Now we should think of the set USES as a tuple $(q, n, u, \ldots)$ where $q$ is the actual relation, $n$ is arity information and $u$ gives the typing information.

# Another Simple Example (Variable Arity)

- `uses` has a fixed arity of 2.
- `partition` has variable arity of at least 2.
- Let P be the set interpreting `partition`.
- Apply ap(P) to 1 argument: a list.

```
(partition Organism Animal Plant Fungus Microorganism)
```

becomes

```
ap(P)   (cons ORGANISM
         (cons ANIMAL
          (cons PLANT
           (cons FUNGUS
            (cons MICROORGANISM nil))))).
```

# Type Guards for Row Variables

```
(=> (partition @ROW)
     (and (exhaustiveDecomposition @ROW)
          (disjointDecomposition @ROW)))
```

becomes

$$\forall \rho . \Gamma(\rho) \rightarrow \mathsf{ap}(\mathsf{P})(\rho) \rightarrow \mathsf{ap}(\mathsf{ED})(\rho) \wedge \mathsf{ap}(\mathsf{DD})(\rho)$$

where $\Gamma(\rho)$ is the guard for $\rho$:

- Length of $\rho$ is at least 2 (min arity of P, ED and DD).
- Each item in $\rho$ is a member of CLASS.

# Query

- Query: Must every organism that is not an animal and not a microogranism be a plant or a fungus? In SUO-KIF:

```
(query
  (forall (?O)
    (=>
      (instance ?O Organism)
      (=>
        (not
          (instance ?O Animal))
        (=>
          (not
            (instance ?O Microorganism))
          (or
            (instance ?O Plant)
            (instance ?O Fungus)))))))
```

- For every exhaustive decomposition of a class into a list of subclasses and member $O$ of the class, there is an element $I$ in the list of subclasses such that $O$ is in $I$. In SUMO:

```
(=>
    (exhaustiveDecomposition ?CLASS @ROW)
    (forall (?OBJ)
        (=>
            (instance ?OBJ ?CLASS)
            (exists (?ITEM)
                (and
                    (inList ?ITEM (ListFn @ROW))
                    (instance ?OBJ ?ITEM))))))
```

# Proof sketch:

- Let $O$ be an organism that is not an animal and not a microorganism.
- There is an exhaustive decomposition of Organism into Animal, Plant, Fungus and Microorganism.
- There is an $I$ in the list of the four subclasses (Animal, Plant, Fungus and Microorganism) such that $O \in I$.
- $I \neq$ Animal since $O$ is not an animal.
- If $I =$ Plant, then we're done.
- If $I =$ Fungus, then we're done.
- $I \neq$ Microorganism since $O$ is not a microorganism. QED

# Proof sketch:

- Let $O$ be an organism that is not an animal and not a microorganism.
- There is an exhaustive decomposition of Organism into Animal, Plant, Fungus and Microorganism.
- There is an $I$ in the list of the four subclasses (Animal, Plant, Fungus and Microorganism) such that $O \in I$.
- $I \neq$ Animal since $O$ is not an animal.
- If $I =$ Plant, then we're done.
- If $I =$ Fungus, then we're done.
- $I \neq$ Microorganism since $O$ is not a microorganism. QED
- This simple sketch of a proof corresponds to 910 subgoals.
- ATPs prove 27% to 72% of the 910 subgoals.

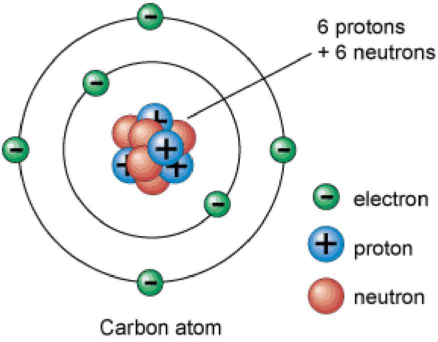
## Kappa Example

Given an atom $V$, SUMO can represent the class of electrons of $V$.

```
...
(instance ?V Atom)
...
(KappaFn ?X
  (and
    (part ?X ?V)
    (instance ?X Electron)))
```

$$
\begin{aligned}
\{X \in \mathsf{U} | X \in & \ \mathsf{OBJECT} \wedge \\
& X \in \mathsf{ENTITY} \wedge \\
& \mathsf{bp}\ (\mathsf{ap}\ \mathsf{PART}\ (\mathsf{cons}\ X\ (\mathsf{cons}\ V\ \mathsf{nil}))) \wedge \\
& X \in \mathsf{ELECTRON}\}
\end{aligned} \tag{1}
$$

# Example 1:




- Query: For every atom $V$ and every electron $E$ that is part of $V$, $E$ is a member of the class of all electrons of $V$.

Example 2:

- Query: For every atom $V$ and every electron $E$ in the class of all electrons of $V$, $E$ is part of $V$.

Example 3:

- Query: For every atom $V$, there is a class $C$ such that for every electron $E$, $E$ is part of $V$ if and only if $E$ is in $C$.

# Conclusion

- We can translate SUMO axiom and queries into higher-order set theory.
- Using higher-order, we can represent SUMO's $\kappa$-class formers naturally as set separation: $\{x \in \mathsf{U} | \psi\}$.
- In 23 test cases the translated queries are provable (interactively).
- In 10 cases the translated queries are provable by at least one higher-order ATP.
- In the other 13 test cases, ATPs can prove a percentage of the subgoals.
- Future work: modalities