Towards Ontological Correctness of Part-whole Relations with Dependent Types

Richard Dapoigny¹  Patrick Barlatier¹

¹LISTIC/Polytech’Savoie
University of Savoie, Annecy, (FRANCE)

Cite as: Dapoigny, R., Barlatier, P. Towards Ontological Correctness of Part-whole Relations with Dependent Types.
Contents

1. Introduction
2. Type Theory
3. Part-Whole specifications
   - Formalization
   - Applications
4. K-DTT with Triples
5. Conclusion
A core problem in ontologies: Part-Whole modeling and reasoning

Some limitations of classical approaches. Usually, reasoning is achieved with Description Logics (DLs) or fragments of First Order Logic (FOL). But there are many challenging problems such as:

- The differences that appear in parthood relations between classes and parthood relations between instances.
- Meta reasoning about Part-whole relations.
- Non-differentiation among the roles that parts play within the structure of a whole.
- Whether the inheritance property holds for parthood relations.
The Layered Model for ontologies

The DL-based model:

**The ontological perspective**

- Language (e.g., OWL) with primitives
- Interpretation \( (I = (\Delta^I, \mathcal{I})) \)
- FOL fragment, modal logic

**The logic-based perspective**

- **ONTOMOGICAL**
- **EPISAROMOGICAL**
- **LOGICAL**

Models \( (\models) \)
Propositional logic, FOL
The Layered Model for ontologies

The K-DTT model:

The ontological perspective

- Language (e.g., OWL) with primitives
- Interpretation ($I = \langle \Delta^I, \mathcal{I}^I \rangle$)
- FOL fragment, modal logic

The Type-theoretic perspective

- Ontological
- Epistemological
- Logical

Typed structures
Type Theory

K-DTT
Type Theory

- Dependent type theory: foundation for constructive mathematics.
- Using constructive logic ⇒ epistemologically unclear steps in proofs are forbidden.
- In computer science, dependent types ⇒ proof assistants and automated theorem provers.
- Dependent type systems mix types and expressions to produce code that is proven to be correct with respect to its expected behavior.
- The Curry-Howard isomorphism: proving ≡ computing.

So, why bother? idea: apply type theory to express and reason about ontological components.
Basis of Type Theory

Universes are stratified (Russell): $\text{Prop} \subseteq \text{Type}_0 \subseteq \text{Type}_1 \ldots \subseteq \text{Type}_{i-1}$
Types and ontologies

Stratified universes: $\text{Type}_0$ is the universe of "instanciable" types (e.g., $\text{House}$ is instanciable in $\text{MyHouse}$), $\text{Type}_1$ the universe for types of instanciable types (e.g., items of the Dolce ontology), and so forth.
Representing knowledge and reasoning with Dependent Types

- Expressing "for all": product types
  \[ \Pi x : Cell . \Pi y : Membrane . \text{LimitedBy}(x, y) \]

- Expressing "for some": sum types
  \[ \Sigma x : Man . \Sigma y : Donkey . \text{Own}(x, y) \]

But we can do more: quantification can be dependent.

Every man who owns a donkey beats it

\[ \Pi d : (\Sigma x : Man. \Sigma y : Donkey. \text{Own}(x, y)). \text{Beat}(\text{fst}(d), \text{fst}(\text{sec}(d))) \]

the domain of quantification of the product (\(\Pi\)) is over the predicate \(\Sigma x : Man. \Sigma y : Donkey. \text{Owns}(x, y)\).
Expressing knowledge

Relations

- Representing relations with n-ary sum-types.

\[ \Sigma x_1 : T_1 \cdot \Sigma x_2 : T_2 \cdot \ldots \cdot \Sigma x_n : T_n \cdot R(x_1, x_2, \ldots, x_n) \]

where \( T_1, T_2, \ldots, T_n \) are types.

Example

\[ \Sigma x : \text{Person} \cdot \Sigma y : \text{Good} \cdot \Sigma z : \text{Shop} \cdot \text{purchaseFrom}(x, y, z) \]

with an example of proofs:

\[ \langle \text{John}, \langle \text{PartsbySimons}, \langle \text{Amazon}, p_1 \rangle \rangle \rangle \]

with: \( p_1 = \text{purchaseFrom}(\text{John}, \text{PartsbySimons}, \text{Amazon}) \)
Subtyping

Basic subtyping relation:

\[
\Gamma \vdash M : A \quad \Gamma \vdash A' : Type \quad A \leq A' \\
\frac{}{\Gamma \vdash M : A'} \text{ (Sub)}
\]

Subtyping of relations: corollary [Luo94]

The relation \( \leq \) is the smallest partial order over terms with respect to conversion such that:

if \( A \leq A' \) and \( B \leq B' \), then \( \Sigma x : A \cdot B \leq \Sigma x : A' \cdot B' \)

Here, \( A \) and \( B \) can be types or sorts (e.g., \( Type_0, Type_1, \ldots \))

In K-DTT, subtyping will be used to denote intensional subsumption.
Type Theory

Representing (binary) *part – of* relations

- \( \text{Rel}(T, T', R) \triangleq \sum T : \text{Type}_i \cdot \sum T' : \text{Type}_i \cdot R(T, T') \) with \( R : T \rightarrow T' \rightarrow \text{Prop} \).
- Formalizing the binary part-whole relations:
  \[
  \text{PW} \triangleq \text{Rel}(T_1, T_2, \text{PartOf})
  \]
- Example of such a relation:
  \[
  \text{Rel}(\text{shaftOfFemur}, \text{Femur}, \text{PartOf}) \triangleq \sum x : \text{shaftOfFemur} \cdot \sum y : \text{Femur} \cdot \text{PartOf}(x, y)
  \]
- How to formally conceptualize the properties of such a relation (e.g., transitivity) within the theory?
- A solution: *specifications*.
What is a specification?

Specifications have been introduced in theoretical computer science [Burstall-Goguen80, Luo94] for proving the correctness of program modules. A specification $Sp$ in K-DTT consists of a pair:

$$Sp = \langle S, P \rangle$$

where $P : S \rightarrow Prop$

- **Structure description**
- **Predicate describing the constraints that the structure should satisfy**

$$Struc[Sp] : Type, \quad Pr : Struc[Sp] \rightarrow Prop$$
Notation

Higher-order logic symbols:

\[\top \implies \text{true (provable in any valid context)}\]
\[\bot \implies \text{absurdity (has no proof)}\]
\[\supset \implies \text{if } P \supset Q \text{ and } P \text{ are provable, then so is } Q\]
\[\& \implies \text{if } P \& Q \text{ is provable, so are } P \text{ and } Q\]
\[\forall \implies \text{if } P[x] \text{ is provable then } \forall x : A.P[x] \text{ is provable}\]
\[\ldots \implies \]
Parameterized specification: an example

Provided that a part–whole relation operating on particulars ($PT$) is transitive, one can introduce parameterized specifications (p-specifications) for transitive relations with:

$$\text{Struc}[\text{Trans}](\text{Rel}(PT, PT, Tr)) \triangleq \text{Rel}(PT, PT, Tr)$$

with $Tr : PT \rightarrow PT \rightarrow \text{Prop}$

$$\text{Pr}[\text{Trans}](\text{Rel}(PT, PT, Tr)) \triangleq \forall x : PT . \forall y : PT . \forall z : PT . (Tr(x, y) \& Tr(y, z)) \supset Tr(x, z)$$
Reasoning with transitive relations

If one has a proof (i.e., knows) that a relation is transitive: \( \text{Rel}(\text{PT}, \text{PT}, \text{PartOf}) \) and since \( \text{Valve} \leq \text{PT} \), \( \text{CarEngine} \leq \text{PT} \) and \( \text{Car} \leq \text{PT} \), then from the corollary, it follows that:

\[
\begin{align*}
\sum x : \text{Valve} \cdot \sum y : \text{CarEngine} \cdot \text{PartOf}(x, y) & \leq \text{Rel}(\text{PT}, \text{PT}, \text{Tr}) \\
\sum x : \text{CarEngine} \cdot \sum y : \text{Car} \cdot \text{PartOf}(x, y) & \leq \text{Rel}(\text{PT}, \text{PT}, \text{Tr})
\end{align*}
\]

Intensional subsumption assumes that these two relations must satisfy the predicative part of their super type, i.e., we get a proof for \( \text{PartOf}(\text{Valve1}, \text{MyCar}) \) with \( \text{Valve1} : \text{Valve} \) and \( \text{MyCar} : \text{Car} \). It means that we have constructed the type:

\[
\sum x : \text{Valve} \cdot \sum y : \text{Car} \cdot \text{PartOf}(x, y)
\]
Operations on specifications

Fusion of specifications

Given two specifications $S$ and $S'$ having the same structure $S_J$:

\[
\text{Struc}[\text{Fusion}(S, S')] \triangleq S_J \\
\text{Pr}[\text{Fusion}(S, S')](z) \triangleq \text{Pr}[S](z) \& \text{Pr}[S'](z)
\]

Extension of specifications

Given a specification $\text{Struc}[S]$ and an extent $E_S$ of that structure:

\[
\text{Struc}[\text{Extent}(S, E_S, Pr_S)] \triangleq \Sigma x : \text{Struc}[S] . E_S(x) \\
\text{Pr}[\text{Extent}(S, E_S, Pr_S)](z) \triangleq \text{Pr}[S](\text{fst}(z)) \& \text{Pr}_S(z)
\]
Introducing meta-properties of relations

With the respective p-specifications $Trans(T, T', R)$, $Refl(T, T', R)$, $Antsym(T, T', R)$ that hold for transitive, reflexive and anti-symmetric relations, one can introduce a p-specification $POR$ for partial order relations:

$$POR(T, T', R) \triangleq Fusion(Refl(T, T', R), Trans(T, T', R), Antsym(T, T', R))$$
$$Struc[POR(T, T', R)] \triangleq Rel(T, T', R)$$
$$Pr[POR(T, T', R)](z) \triangleq Pr[Refl(T, T', R)](z) \& Pr[Trans(T, T', R)](z) \& Pr[Antsym(T, T', R)](z)$$

with $z : Rel(T, T', R)$. 
Introducing meta-properties of relations

Similarly, an intransitive relation $ITR$ is such that:

$$ITR(T, T', R) \triangleq Fusion(Asym(T, T', R), ITrans(T, T', R))$$

$$Struc[ITR(T, T', R)] \triangleq Rel(T, T', R)$$

$$Pr[ITR(T, T', R)](z) \triangleq Pr[Asym(T, T', R)](z) \& Pr[ITrans(T, T', R)](z)$$

and the specification $IRT(PT, PT, PPartOf)$ for proper-part-of p-specifications:

$$IRT(T, T', R) \triangleq Fusion(Asym(T, T', R), Trans(T, T', R))$$

$$Struc[IRT(T, T', R)] \triangleq Rel(T, T', R)$$

$$Pr[IRT(T, T', R)](z) \triangleq Pr[Asym(T, T', R)](z) \& Pr[Trans(T, T', R)](z)$$
Part-whole p-specifications

Following [Keet08] different types of part-whole relations can be specified,

**Generic Part-whole relations**

**Transitive PartOf:**  \( \text{POR}(PT, PT, \text{PartOf}) : \ POR(T, T', R) \)

**Non-transitive meronymic PartOf:**  \( \text{NTR}(PT, PT, \text{MPartOf}) : \ \text{Asym}(T, T', R) \)

**Intransitive proper PartOf:**  \( \text{IRT}(PT, PT, \text{PPartOf}) : \ \text{IRT}(T, T', R) \)

**Intransitive meronymic PartOf:**  \( \text{ITR}(PT, PT, \text{MPartOf}) : \ \text{ITR}(T, T', R) \)
Consequence

If a specification $S$ subsumes a specification $S'$, then $S'$ must satisfy all the axioms of $S$.

Example

Let us introduce the specification $involved – in$ such that $\text{Struc}[involved – in(PD, PD, InvolvedIn)] \leq \text{Struc}[POR(PT, PT, PartOf)]$, we have:

$$(\Sigma x : Chewing. \Sigma y : Eating. InvolvedIn(x, y)) : involved – in(PD, PD, InvolvedIn)$$

then $Rel(Chewing, Eating, InvolvedIn)$ satisfies all the predicates of its super-type (i.e., it is reflexive, transitive and antisymmetric).
The formal ontology of part-whole p-specifications
Example 1

Correctness of user-defined relations

From DOLCE: $ED \leq PT$ and $PD \leq PT$. If we add the types $Car$ and $CarChassis$ such that: $Car : ED$ and $CarChassis : ED$, a part-of relation where $CarChassis$ is a part of $Car$. Since $ED \leq PT$, the relation is of type $POR(PT, PT, PartOf)$, then it is logically correct to speak of part-of relation between $CarChassis$ and $Car$.

If $\sigma_1 \triangleq \Sigma x : CarChassis . \Sigma y : Car . PartOf(x, y)$, it is possible to automatically derive the inverse relation:

$$\Pi z : \sigma_1 . HasPart(fst(sec(z)), fst(z))$$
Example 1 (cont.)

Correctness of user-defined relations

If we apply the *involved – in* relation with the types *Car* and *CarChassis*:

\[ \Sigma x : CarChassis . \Sigma y : Car . InvolvedIn(x, y) \]

From the ontology of p-specifications this relation type should be subsumed by:

\[ \Sigma x : PD . \Sigma y : PD . InvolvedIn(x, y) \]

but *Car* \( \leq PD \) \( \supset \bot \) and *CarChassis* \( \leq PD \) \( \supset \bot \)
then this definition is rejected.
Example 2

Arguments of a relation more general than those of its parent relation

Suppose that we assert:

- The respective argument types are $PD$ and $PT$ for $PartOf$ and $involved – in$.
- $CarChassis : PT$ and $Car : PT$.

The ontology of specifications expects that the following equation holds:

$$\sum x : PT \cdot \sum y : PT \cdot InvolvedIn(x, y) \leq \sum x : PD \cdot \sum y : PD \cdot PartOf(x, y)$$

The subtyping requires that $PT \leq PD$, which is in conflict with the DOLCE ontology of particulars.
Example 3

High Level Reasoning

→ Let us define a subquantity — of specification that is intransitive.
→ Its p-specification (in Type$_2$) must be also irreflexive.
→ Choice: $ITR(PT, PT, MPartOf)$ subsuming subquantity — of.
→ An instance of the specification subquantity — of is created in the KB.
→ By propagating this instance upwards in the ontology of specifications, an instance of the p-specification $ITrans(T, T', R)$ is created, result: Ok.
→ If a Type$_1$-specification inherits of a specification which generates an incoherence at the Type$_2$ level, then either:
  ▶ a constraint in Type$_2$ is relaxed (e.g., intransitivity)
  ▶ or the Type$_1$-specification is moved across the ontology until it inherits from correct properties.
Expressing K-DTT with Triples

- Any information is expressed in the form of triples store, but without using the semantic related to RDF/OWL since we have designed a dedicated inference engine with AllegroCL.
- The ontology is divided into sub-domains that are mapped into different namespaces with the purpose of distinguishing between the related terminologies.
- The predicate part of triples describe K-DTT operations, e.g., ":" gives $IsOfType$, $\leq$ gives $SubTypeOf$ and so forth.
- Data structures are given with the following triples.
K-DTT with Triples

Triples

\[ T = \sum_{a:A} B(a) \]

\[ B = \prod_{a:A} Type_1 \]

Dapoigny, Barlatier (University of Savoie)
Towards Ontological Correctness of Part-whol...
\[ T = \sum_{a:A} (\sum_{b:B} C(a, b)) \]
Major Benefits

- Main result: an ontology of p-specifications providing a number of benefits:
  - Meta-reasoning is done within a single logic.
  - Syntactic differentiation between \textit{partOf} and meronymic relations through p-specifications.
  - \textit{partOf} relations are unambiguously identified through their p-specifications with argument restriction.
  - The unifying logical theory is able to provide the logical consistency for user-defined structures.

- However, there is a cost of greater difficulty during the modeling stage → a guidance should be designed through, for instance, a question answering process.
Perspectives

- This could be generalized by defining the ontology as a (structured) type and defining rules of inference as inductive relations so as to give a computational understanding of the ontology.
- Specifications can be shared between systems and enhance the expressive power of these systems.
Now, it’s time for some questions.