A Modular Consistency Proof for DOLCE

Oliver Kutz¹, and Till Mossakowski¹,²
¹. SFB/TR 8 Spatial Cognition, Bremen, Germany
². DFKI Lab Bremen, Germany
{okutz,till}@informatik.uni-bremen.de Till.Mossakowski@dfki.de

Abstract
We propose a novel technique for proving the consistency of large, complex and heterogeneous theories for which ‘standard’ automated reasoning methods are considered insufficient. In particular, we exemplify the applicability of the method by establishing the consistency of the foundational ontology DOLCE, a large, first-order ontology. The approach we advocate constructs a global model for a theory, in our case DOLCE, built from smaller models of subtheories together with amalgamability properties between such models. The proof proceeds by (i) hand-crafting a so-called architectural specification of DOLCE which reflects the way models of the theory can be built, (ii) an automated verification of the amalgamability conditions, and (iii) a (partially automated) series of relative consistency proofs.

Introduction
The field of formal ontology may be subdivided into the study of domain ontologies, devoted to specific application areas, and foundational ontologies, axiomatising fundamental and domain-independent concepts. Foundational ontologies, such as SUMO (Niles and Pease 2001), DOLCE (Gangemi et al. 2002), GFO (Heller and Herre 2004), and BFO (Grenon, Smith, and Goldberg 2004), are typically specified in some variant of first-order logic, and their first-order theories tend to be rather large. DOLCE, for instance, consists of a few hundred axioms, and SUMO of several thousand.

Automated and semi-automated theorem proving systems have successfully been applied to reasoning about foundational ontologies. In particular, using automated provers, a number of inconsistencies in SUMO have been found (Voronkov 2006; Horrocks and Voronkov 2006), and SUMO has been corrected accordingly. The problem of proving the consistency of ontologies, however, is much harder in general.

In the literature, two main approaches for proving consistency are described: model finders and relative consistency proofs. There are several model finders for first-order logic available. Some of them search for finite models by a translation to propositional logic (and then using SAT solvers) (e.g. Isabelle-refute (Weber 2005)), some of them use more advanced methods like the model evolution calculus (e.g. Darwin (Baumgartner and Tinelli 2003; Baumgartner, Fuchs, and Tinelli 2004)), or resolution via detecting a saturated set of clauses (e.g. SPASS (Weidenbach et al. 2002)). However, these techniques currently only suffice to find models for relatively small first-order theories—they do not scale to DOLCE, let alone SUMO. In fact, the difficulties already arise for the rather small sub-theories ‘classical extensional parthood’ (CEP) and ‘constitution’ (CON) of DOLCE. CEP is a theory of mereology, and it is straightforward to see that finite models for it can be obtained by powersets of finite sets, where the empty set has to be excluded. The singleton sets are then just the atoms of the mereology. The above first-order model finders could not find models with more that four atoms for these theories. Moreover, several weeks of computation time did not suffice to find a model for the whole of DOLCE.

An alternative way of proving consistency is to use a relative consistency proof, that is, to provide a theory interpretation into some other theory that is known (or assumed) to be consistent. An obvious disadvantage of this approach is that it not only requires the manual construction of such a theory interpretation, but that such an interpretation will also typically be rather large and complex.

In this work, we propose to construct models not in a monolithic, but in a structured way. We employ a set of operations for model construction that have been introduced in the context of software specification under the name of architectural specifications. These allow for decomposing the task of constructing a model for a (large) theory into smaller subtasks. These subtasks include: (a) automatically finding (or manually constructing) models for (relatively) small theories, (b) proving the conservativity of theory extensions, which can be done performing (local) relative consistency proofs, and (c) establishing amalgamability between already constructed models (or model classes).

Relative Consistency Proofs
For the purposes of this paper we shall identify a (first-order) ontology with a theory in first-order logic, namely a signature (set of non-logical symbols) and a set of axioms. We will say that a theory is consistent (=satisfiable) if it has a model; by completeness this is equivalent to formal consistency, which means that no contradiction can be derived.
When we are unable to directly establish that a certain theory, say \( T \), is consistent, we can instead show that it is consistent provided some other theory \( T' \) is.

The general method behind this is as follows: \( T' \) is extended conservatively with new definitions (call the resulting theory \( T'' \)), and then \( T \) is interpreted in \( T'' \), via a theory morphism (interpretation of theories) \( \sigma : T \rightarrow T'' \). Now if \( T' \) is consistent, it has a model. Since \( T'' \) is a conservative extension, it has a model, too, and this model can be reduced (via \( \sigma \)) to a model of \( T \). Hence, altogether, consistency of \( T' \) implies that of \( T \).

Let us make the notion of ‘conservative extension with new definitions’ a bit more precise. A conservative extension of a theory \( T \) is a theory extension \( \iota : T \rightarrow T' \) such that for any model \( M \) of \( T \), there is a \( \iota \)-expansion of \( M \) to a \( T' \)-model \( M' \), i.e. such that the reduct \( M'||\iota \) of \( M' \) via \( \iota \) is again \( M \). If the \( \iota \)-expansion is in fact always unique, then the theory extension is called **definitional**.

We can summarise the above as follows: diagrammatically, we can represent relative consistency proofs in the form of **conservativity triangles** as shown in Fig. 1, i.e. we are given theories \( T, T', T'' \), and signature morphisms \( \sigma : T \rightarrow T'', \iota_1 : T' \rightarrow T'' \), and \( \iota_2 : T' \rightarrow T \) such that the following triangle commutes:

\[
\begin{array}{c}
T' \\
\sigma \downarrow & \downarrow \iota_2 \\
T'' & \rightarrow & T
\end{array}
\]

**Figure 1: A conservativity triangle**

We then have the following:

**Lemma 1.** Let \( T \) be a conservativity triangle. Suppose \( T' \) is consistent. If \( \iota_1 \) is conservative (definitional) and \( \sigma \) is a theory interpretation, then \( \iota_2 \) is conservative and \( T \) is consistent.

Since conservativity of theory extensions is in general undecidable (not even semi-decidable), we need syntactic criteria that are sufficient (but not necessarily necessary) to ensure conservativity. Obviously, extensions by explicit definitions of function and predicate symbols are conservative in this sense. Considering a many-sorted first-order logic, we will also allow extensions by definitional introduction of new sorts, where the new sort is either given by a finite enumeration of constants that are pairwise different and jointly exhaustive, or by a disjoint union of a finite number of already existing (i.e. in \( T' \)) sorts. All this is easily expressible in first-order logic.

With these criteria for conservativity, we can even show the (absolute) consistency of a theory \( T \): namely, if we let \( T' \) be the empty theory (which trivially is consistent), conservatively extend it to \( T'' \) and then interpret \( T \) in \( T'' \).

Of course, this method is still quite unstructured. If \( T \) is significantly larger than \( T' \) (say, more than 10 new non-trivial axioms), showing the conservative extension property may be quite unmanageable. Moreover, this method often easily succeeds with trivial results: we can find a model of DOLCE by interpreting most concepts as the empty set. But then, the question whether this emptiness is essential or whether other, non-trivial models exists, remains unresolved.\(^1\) Note, in this context, that in typical applications of foundational ontologies, namely when the foundational ontology is refined against a domain ontology (see e.g. (Gangemi et al. 2002)), the main problem with regard to consistency is to settle the question whether models instantiating certain parts of the foundational signature exist. This cannot be resolved by consulting trivial models, but requires knowledge about the structure of possible models of the foundational ontology.

A more structured approach uses a decomposition into a sequence of conservativity proofs:

\[
T_1' \rightarrow T_2' \rightarrow \ldots \rightarrow T_n' \rightarrow T
\]

This is (a) better manageable, and (b) due to the decomposition into small and independent steps, it is easier to find non-trivial models.

**Consistency and Architectural Specification**

It turned out that such a linear decomposition still is not well-suited for dealing with the subtle interactions that occur in DOLCE. In fact, we need a tree-like decomposition\(^2\) as shown in Fig. 2:

\[\text{Figure 2: The Refinement Tree for DOLCE.}\]

At the branching points, models need to be combined (amalgamated), and it has to be ensured that overlapping parts of models that are amalgamated are indeed equal, based purely on certain sharing conditions induced by the graph structure.

Indeed, there is a language from the software specification community that allows for writing and analysing such decompositions: CASL’s **architectural specifications** (Bidoit, Sannella, and Tarlecki 2002). Basically, an architectural specification consists of a sequence of declarations of **units** (which are just named models), declarations of **unit functions** (mapping models of a smaller theory to models of an extended theory), and definition of units by **unit terms**. A unit term may refer to named units, apply unit functions to

---

\(^1\)Models with empty ‘categories’ are in fact not considered proper models by the DOLCE designers.

\(^2\)In general, such a decomposition can yield any acyclic graph.
other units, take reducts of units, and amalgamate units to larger units. Finally, an overall result unit term yields the overall model that is provided by the architectural specification.

Additionally to units, we have unit specifications. These can either be structured specifications of single units, be structured specifications of single units, for which a model has to be found directly (after flattening), or specifications of parametrised units, with separate specifications for arguments and results. The latter roughly correspond to theory extensions, since the result specification always must extend the union of the argument specifications. A unit declaration declares a unit name to have a given unit specification. Given a context of such unit declarations, unit terms express more complex units (derived from the declarations).

A unit term can be a single unit name, or the application of a parametrised unit to unit terms or an amalgamation of unit terms with the keyword and. Finally, an architectural specification has the form as in Fig. 3, where the $U_i$ are unit declarations and $UT$ is a unit term. It is also possible to define named units at any place in the units-part and to use them further on.

The semantics of architectural specifications ensures that this is done in such a way that any realisation of the declared units leads to a model corresponding to the result unit term. In particular, this means that appropriate sharing conditions are checked; namely, if two (or more) units are amalgamated, then the shared symbols must originate from the same declared unit.

In this way, the consistency of large theories can be reduced to the consistency of a number of unit declarations. The latter amounts to consistency of smaller theories (in case of simple units) or to conservativity of theory extensions (in the case of parametrised units). Consistency of small theories as well as conservativity of theory extensions can be checked with the means discussed above.

Figure 5: DOLCE’s taxonomy.
• the theory extensions must be large enough to guarantee the amalgamability conditions.

Indeed, the check of the amalgamability conditions has been implemented as part of the Heterogeneous Tool Set HETS (Klin et al. 2001; Mossakowski, Maeder, and Lüttich 2007). This is of great help when designing an architectural decomposition for DOLCE.

An important design principle of architectural specifications is the presence of an abstraction barrier between the different units. Namely, when providing the realisation of a parametric unit (which extends smaller models to larger ones), it is not allowed to look into the specific construction of the parameter units (models). Rather, only the properties of the parameter models can be exploited, as given by their specifications. Although this principle sometimes makes the construction of models for parametrised units more difficult, because less properties can be exploited for the parameter models, in the end, there is a great pay-off: namely, the overall model construction has been split in a number of subtasks that are really independent. That is, we can locally change the construction of the model (say, e.g., by interpreting concept SC (‘Society’) in a more complex way), without losing the guarantee that the different subparts can always be amalgamated to a global model of DOLCE.

Our first attempt of designing an architectural specification for DOLCE largely followed the specification structure of DOLCE as shown in Fig. 4. The various notions are introduced for certain concepts in the taxonomy (Fig. 5) and automatically inherited for the subconcepts. Therefore, the taxonomy itself can be integrated separately at a quite late stage. However, this attempt badly failed at this late stage: namely, after having successfully covered most of the sub-theories, we faced the problem that the specification DEPENDENCE introduces subtle dependencies between various parts of DOLCE’s taxonomy.

(Timed) Mereology: Bottom Up vs. Top Down

Hence, we needed to completely restructure the architectural specification. Most importantly, the model for TEMPORARY_PARTHOD cannot be constructed for the top concepts in the taxonomy and then be inherited for the subconcepts. Rather, it has to be introduced in a bottom up manner.

This bottom-up strategy also has an impact on the choice of the logic. DOLCE originally has been formulated in single-sorted first-order logic. However, this logic complicates a modular consistency proof. If we wanted to fix the interpretation of the different concepts of the taxonomy in a step-by-step fashion, we would repeatedly need to extend the universe of discourse. By contrast, when using a sub-sorted variant of first-order logic, we can step by step add interpretations of individual concepts: the interpretations of super-concepts (aka supersorts) just combine and possibly extend the interpretations of their sub-concepts (aka subsorts). We here use the logic SubFOL used in CASL, see (Astesiano et al. 2002) for details. Note that from a subsorted model, it is straightforward to construct a single-sorted model by mapping subsorts into predicates.

The temporal mereology in TEMPORARY_PARTHOD is specified in DOLCE using a ternary predicate $tP$, where $tP(x, y, t)$ means that at time $t$, $x$ is part of $y$. For fixed $t$, this is required to be a partial order. In terms of $tP$, further concepts like overlap and sum are specified:

```plaintext
forall x : s; y : s; t : T
 . tOv(x, y, t) <=> exists z : s . tP(z, x, t) \ tP(z, y, t)
```

```plaintext
forall z : s; x : s; y : s
 . tSum(z, x, y)
 <=>
forall w : s; t : T
 . tOv(w, x, t) \ tOv(w, y, t)
```

Moreover, mereological sums are required to exist:

```plaintext
forall x, y : s . exists z : s . tSum(z, x, y)
```

(and similarly for differences). Central concepts of DOLCE are endurant (ED, roughly: objects) and perdurants (PD, roughly: processes). The concept ED is required to be a temporal mereology, while PD is only required to be a normal mereology (i.e. where the time parameter $t$ is omitted).

Now the bottom-up construction of the TEMPORARY_PARTHOD model bears one important problem: DOLCE requires concepts occurring higher in the taxonomy to be a disjoint union of their subconcepts. Yet, the model class of TEMPORARY_PARTHOD is not closed under disjoint unions, essentially because these are in general not closed under the mereological sum and difference operations. Therefore, we have introduced a subtheory TEMPORARY_PARTHOD\NoSUM of TEMPORARY_PARTHOD that omit the requirement of existence of sums and differences. We then have constructed a model for DOLCE’s timed mereology as follows:

• we take arbitrary models of TEMPORARY_PARTHOD for the leaves of the taxonomy below ED: SC, SAG, NASO, APO, NAPO, MOB, F, and M. Here, we can take model of different cardinality and structure for different sorts, if we like;

• at inner nodes below ED, we take the disjoint union of subconcepts, ending up in a model of a of TP\NoSUM (the latter being closed under disjoint unions);

• for the concept ED (the top concept of the temporal mereology), we take terms made up of formal sums and differences of all elements in ED’s immediate subconcepts PED (physical endurant) and NPED (non-physical endurant). Such a formal term is then taken to live within PED or NPED iff the corresponding (posssibly nested) sum/difference does already exist in PED or NPED, respectively. Otherwise, it is put into AS (arbitrary sum).

A similar construction would be possible for the static mereology PD, if it also had a subconcept for arbitrary sums. However, in DOLCE, it does not. Since the problem with dependency relations discussed above does not appear among the subconcepts of PD, we instead can define a model in a top-down manner. However, note that and put the sums PD is specified to be the disjoint union of its two subconcepts EV (event) and STV (stative). This means that means that have to put sums of mixed atoms (say, a sum of an atom in EV and one in STV) arbitrarily into either EV or STV. Although this suffices for showing consistency, it is conceptually wrong (indeed, the sum of an event and a stative need
be neither an event nor a stative). Indeed, we think that this reveals a design flaw of DOLCE.

The architectural design is summed up in the architectural specification PARThOOD_MODEL in Fig. 6. It actually corresponds to the lower-most branching of the refinement tree in Fig. 2. In context $U : SP_1$, the notation $V : SP_2$ GIVEN $U$ expresses that unit (model) $V$ is built over unit $U$. Technically, this is syntactical sugar for the definition of an anonymous parameterised unit $SP_1 \rightarrow SP_2$, which is then applied once (to unit $U$). The notation $\text{TEMPORARY\_PARThOOD}$ with $s \rightarrow SC$ renames sort $s$ in $\text{TEMPORARY\_PARThOOD}$ appropriately. The notation $\text{free type ASO ::= sort SC | sort SAG}$ is CASL’s shorthand for the first-order sentence expressing that ASO is the disjoint union of SC and SAG.

**Strengthening Specifications**

Another lesson learned from the subtle interactions introduced by the specification $\text{DEPENDENCE}$ is a follows: sometimes, in induction proofs, it is necessary to strengthen the inductive theorem in order to prove it. While this sounds paradoxically at first sight, the reason becomes clear when considering that strengthening the theorem also strengthens the inductive hypothesis. Likewise, by strengthening the specification $\text{DEPENDENCE}$, we could rely on stronger assumptions for the interpretation of $\text{DEPENDENCE}$ for various subconcepts when extending it to a superconcept. (This, of course, is an instance of the above mentioned abstraction barrier.) This can be made formal as follows. Call an architectural specification exactly matching if for all unit applications $F[A]$, if $A : SP$, then $F : SP \rightarrow SP_1$ for some $SP_1$. That is, specifications of formal parameter actual parameter match exactly.

**Theorem 1.** Let $\text{ASP}$ be an exactly matching architectural specification, and $SP, SP'$ be structured specifications such that $SP' \models SP$ (i.e. $SP'$ is stronger than $SP$). Let $\text{ASP}'$ be obtained by replacing every occurrence of specification $SP$ in $\text{ASP}$ by $SP'$. Then the consistency of $\text{ASP}'$ implies that of $\text{ASP}$.

Note that the replacement of $SP$ by $SP'$ affects both argument positions (this can be compared to inductive hypotheses) and results positions (this can be compared to inductive steps) of parameterised units.

An example is again $\text{TEMPORARY\_PARThOOD}$, or more precisely, its subtheory $\text{TP\_NO\_SUM\_ETERNAL}$. We have strengthened this specification to $\text{TP\_NO\_SUM\_ETERNAL}$, which requires that the binary predicate $\text{PRE}(x, t)$ (‘temporal presence’) is universally true on one selected ‘eternal object’. This is needed in order to deal with specific dependence (SD), which is defined as follows:

\[
\text{SD} \quad \text{(x:MOB; y:APO)} \iff \exists t : T. (\text{PRE}(x, t)) \land (\forall t. t. (\text{PRE}(x, t)) \Rightarrow \text{PRE}(y, t))
\]

SD introduces dependencies between different sorts, here MOB and APO, regarding temporal presence. Intuitively, $\text{SD}(x, y)$ implies that $y$ is present at more time points $t$ than $x$. Introducing eternal objects here is a technical device that allows to have greater control in the concrete model construction, i.e. the definitional introduction of the sort MOB

```
arch spec Parthood_Model =
  units
  TM : Time_Mereology;
  TP_SC : {Temporary_Parthood with s |-> SC} given TM;
  TP_SAG : {Temporary_Parthood with s |-> SAG} given TM;
  TP_NASO : {Temporary_Parthood with s |-> NASO} given TM;
  and OneSide_Generic_Dependence with s1 |-> NASO, s2 |-> SC
  given TP_SC;
  TP_APO : {Temporary_Parthood with s |-> APO} given TP_SAG;
  and OneSide_Generic_Dependence with s1 |-> SAG, s2 |-> APO
  given TP_SAG;
  TP_F : {Temporary_Parthood with s |-> F} given TM;
  TP_NAPO : {Temporary_Parthood with s |-> NAPO} given TP_F;
  and OneSide_Generic_Dependence with s1 |-> F, s2 |-> NAPO
  given TP_F;
  TP_ASO : {Temporary_Parthood_No with s |-> ASO}
  and (free type ASO ::= sort SC | sort SAG)
  given TP_SC, TP_SAG;
  TP_MOB : {Temporary_Parthood with s |-> MOB} given TP_APO;
  and OneSide_Generic_Dependence with s1 |-> MOB, s2 |-> APO
  given TP_APO;
  TP_SOBO : {TP_NoNoSum_Eternal with s |-> SOBO}
  and (free type SOBO ::= sort ASO | sort NASO)
  given TP_NASO, TP_ASO;
  TP_POB : {TP_NoNoSum_Eternal with s |-> PBOB}
  and (free type PBOB ::= sort APO | sort NAPO)
  given TP_MOB, TP_NAPO;
  TP_NPOB : {TP_NoNoSum_Eternal with s |-> NPOB}
  and (free type NPOB ::= sort SOB | sort MOB)
  given TP_SOBO, TP_MOB;
  TP_M : {Temporary_Parthood with s |-> M} given TP_POB;
  TP_NFED : {TP_NoNoSum_Eternal with s |-> NFED}
  and (free type NFED < NFED) given TP_NPOB;
  TP_FED : {TP_NoNoSum_Eternal with s |-> FED}
  and (free type FED ::= sort PBO | sort M | sort F)
  given TP_M;
  TP_ED : {Temporary_Parthood with s |-> ED}
  and (free AS;
  free type ED ::= sort PED | sort NPED | sort AS)
  given TP_FED, TP_NPED;
  CEP_PD : {{Classical_Extensional_Parthood and sort s} with s |-> PD} given TP_ED;
  Particip : Participation given CEP_PD;
  Mereology_and_TemporalPartPD : Mereology_and_TemporalPart
  given Particip;
  result Mereology_and_TemporalPartPD
end
```

Figure 6: Architectural specification for mereology.

and the predicate SD, and is employed for the theories of $\text{DEPENDENCE}$ as well as $\text{TEMPORARY\_PARThOOD}$. Given the eternal object in APO we can give the following definitions:

\[
\forall x : MOB; y : APO. \text{SD}(x, y) \iff \forall t : T. \text{PRE}(y, t)
\]

Here, MOB as a new sort can be locally instantiated with e.g. a singleton or $n$-element universe, and the definitions shown will be part of the theory $T'\nu$ in the corresponding conservativity triangle as depicted in Fig. 1. These definitions now make the conservativity and theory interpretation claims that need to be established easily verifiable by
a reasoner such as SPASS. We have introduced a total of 10 eternal objects to make the relative consistency proofs go through, namely on the sorts SC, SAG, F, APO, NPED, NAPO, M, SAG, PED, PD. The existence of the eternal PED and NPED objects is however already implied by e.g. APO and SAG objects, given the taxonomy.

**Putting Things Together**

Altogether, the resulting architectural specification consists of 37 units, one (of 'Time Mereology', which is the mereology of the time-line $T$) unparametrised (a model could be found directly by a model finder), the others parametrised— that is, 36 conservativity statements had to be proved. Moreover, the specification involves 18 amalgamations, the corresponding (non-trivial, due to the presence of subsorting) amalgamability checks can automatically be checked by HETS. HETS can also automatically discharge some of the proof obligations yielded by the conservativity triangles by purely structural reasoning, e.g. in the case a theory, instantiated with different subsorts, has to be repeatedly verified on a certain finite model. An example is given by an $n$-point model for temporal parthood. We stress that the choice of the details of the model can be made independently for each of the 37 cases. This leads to a plethora of models for DOLCE, obtained by combining suitable independent local decisions concerning the interpretation of individual concepts. However, note that sometimes the local decision can be quite constrained; e.g. for ED, which by an axiom of DOLCE always has to be interpreted as disjoint union of PED, NPED and AS, the interpretations of PED and NPED already given, and only AS is newly interpreted (see Fig. 6).  

**Summary and Outlook**

We have argued that the problem of establishing consistency for complex first-order theories is currently beyond the scope of standard automated reasoning techniques. Moreover, even if consistency can be established by providing a ‘trivial’ model, in ontology engineering this is often of rather limited use as the existence of models variously instantiated parts of the foundational signature has to be established.

We have proposed a methodology based on architectural specification that breaks down difficult consistency proofs into (a) small and easy consistency proofs, (b) (manageable) proofs of conservativity of theory extensions, and (c) automated proofs of amalgamability of ‘partial’ models.

The main difficulty here is the design of an appropriate architectural specification. Once this is achieved, the technique allows to automate consistency proofs that cannot be obtained by ‘standard’ means to a high degree. Furthermore, it is suited in particular to analyse the fine-structure of possible models for complex first-order theories, namely by allowing to locally modify parts of a (global) model without affecting overall consistency.

While proving consistency in this way, we have found a design flaw of DOLCE, which probably would not have been found with monolithic methods, since it does not destroy consistency. Future work includes studying the fine-structure of the DOLCE models that can be thus obtained, and applying the technique to other foundational ontologies (like SUMO) as well as complex first-order theories in general.

**References**


---

4 All formal specifications, including DOLCE itself, its architectural specification, and the 37 model constructions can be found at the following anonymous URL http://aaa1ldolce.tripod.com/