# A Modular Consistency Proof for DOLCE

**Oliver Kutz**[1] **and Till Mossakowski**[{1,2}]
1. Research Center on Spatial Cognition (SFB/TR 8), University of Bremen, Germany
2. DFKI GmbH Bremen, Germany
okutz@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 (Masolo et al. 2003), GFO (Herre 2010), 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 than 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.[1] 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



[1] Early work towards the consistency proof for DOLCE presented here appeared in (Kutz, Lücke, and Mossakowski 2008).

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) $\sigma : T \to T''$. 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 $\iota : T \to T'$ 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 $\sigma : T \to T'', \iota_1 : T' \to T''$, and $\iota_2 : T' \to T$ such that the following triangle commutes:

$$\begin{tikzcd}
 & T' \arrow[dl, dotted, "\iota_1"', "def!"] \arrow[dr, "\iota_2", "cons"'] & \\
T'' & & T \arrow[ll, dotted, "\sigma"]
\end{tikzcd}$$

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 $T''$.

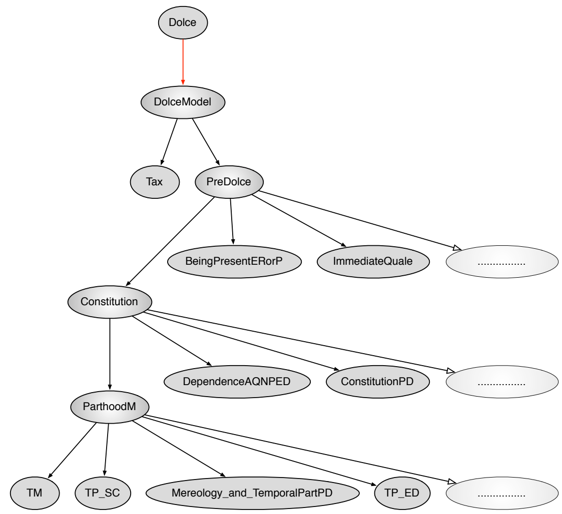



Figure 2: Part of the refinement tree for DOLCE.

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.[2] 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, visualised as follows:

$$T'_1 \longrightarrow T'_2 \longrightarrow \ldots \longrightarrow T'_n \longrightarrow T$$

This represents an intermediate step between a completely monolithic conservativity proof and a tree-like decomposition as shown in Fig. 2, and already is (a) more 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[3]

[2]Models with empty 'categories' are in fact not considered proper models by the DOLCE designers.

[3]In general, such a decomposition can yield any acyclic graph.

as shown in Fig. 2, which illustrates an overall picture[4] of the development of a Dolce model, successively constructing larger units out of smaller units (here, units are models, or functions on models).

The specification DOLCE is refined to an architectural specification DOLCEMODEL. Architectural specifications introduce branching into the development; here, DOLCEMODEL branches into PREDOLCEMODEL and TAX, the latter containing the full Dolce taxonomy. PREDOLCEMODEL in turn branches into CONSTITUTION and others (CONSTITUTION is about objects constituting other objects during certain time intervals). This in turn branches into PARTHOODM and others. With PARTHOODM, we reach the bottom leaves of the tree, which mark the start of the development. Actually, the whole development starts with a model TM of time mereology, formalising the parthood relation for time intervals. This is then extended to a model TP_SC of temporal parthood for 'Society' (SC), which is in turn extended in many steps to a model TP_ED of temporary parthood for endurants. This in turn is extended to a model MEREOLOGY_AND_TEMPORALPARTPD providing a mereology for perdurants as well, completing PARTHOODM. Note that the dependencies among units within PARTHOODM is non-linear; the dependency graph is shown in Fig. 3. Similar dependency graphs exist for the other branching points: CONSTITUTION, PREDOLCEMODEL and DOLCEMODEL; they are omitted here.

Informally, a model for DOLCE can then be constructed by providing models for all the leaves in the refinement tree. 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 dependency graph structure of the branching point (like that in Fig. 3).

CASL architectural specifications (Bidoit, Sannella, and Tarlecki 2002), originally invented in the context of software specification, provide a language for writing down such branching points, with a semantics ensuring amalgamability. Architectural specifications are agnostic with respect to both the underlying logic (e.g. first-order logic) and the language for structuring specifications (e.g. a hierarchic structure, where specifications may import other ones, cf. imports in $\mathcal{OWL}$ or Common Logic). Essential is that each such structured specification has as its semantics a signature (= vocabulary) and a class of models (typically, these are those models over the signature that satisfy the axioms of the specification). Note that, while structured specifications do provide a hierarchical structure for logical theories, they still treat models in a monolithic way. This is where architectural specifications come in.

The building blocks of an architectural specification are **units**. A unit can be just a declaration of a model, e.g. TM : TIME_MEREOLOGY states that TM is a model of (the structured specification) TIME_MEREOLOGY. Alternatively, a unit can be a **parametrised unit**, mapping models to models. For example, TP_SC : TIME_MEREOLOGY $\rightarrow$ TEMPORARY_PARTHOOD declares a parametrised unit that takes any model of the parameter specification TIME_MEREOLOGY and extends it to a model of the result specification TEMPORARY_PARTHOOD. With the so-called **unit application** TP_SC[TM], we can apply TP_SC to the model TM. As shorthand, we may also write TP_SC : TEMPORARY_PARTHOOD GIVEN TM. This declares TP_SC to be a parametrised unit that is applied once.

[4] Which is here very simplified: many nodes are omitted and hidden in the white nodes with dotted content.

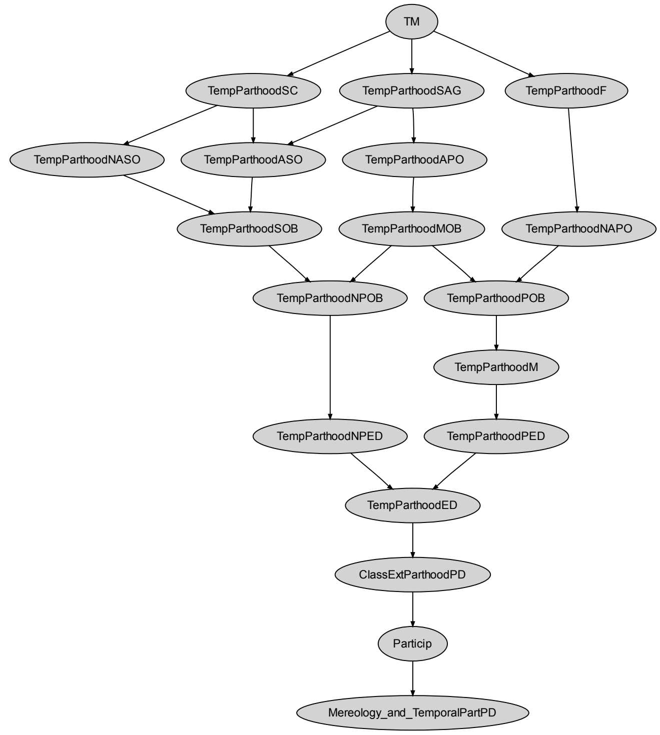



Figure 3: Dependency graph for ParthoodM.

Parametrised units introduce an important *abstraction barrier*. In the above example, the task of constructing a model TM for TIME_MEREOLOGY is separated from the task of extending an (arbitrary) model of TIME_MEREOLOGY to a model of TEMPORARY_PARTHOOD. When constructing the latter, we are not allowed to "look inside" TM, but can only exploit that it satisfies the axioms in TIME_MEREOLOGY.

**arch spec** ASP =
  **units** $U_1 : USP_1$;
    ...
    $U_n : USP_n$
  **result** $UT$
**end**

Figure 4: Architectural specification

An architectural specification (see Fig. 4) consists of a sequence of declarations of (possibly parametrised) units, and definition of units by **unit terms**. A unit term may refer to named units, apply a parametrised unit to 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.

The semantics of architectural specifications ensures that any realisation of the units $U_1, \ldots, U_n$ 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.

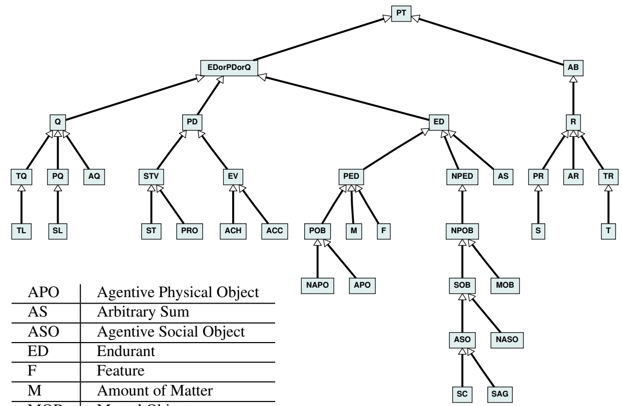
In this way, the consistency of large theories can be reduced to the consistency of a number of unit declarations. The latter amounts to consistency of smaller theories, in case of non-parametrised units. For parametrised units, we always require that the result specification extends the parameter specification. The parametrised unit is consistent iff this extension is (model-theoretically) conservative (that is, any model of the parameter specification can be extended to a model of the result specification). Now consistency of small theories as well as conservativity of theory extensions can be checked with the means discussed above.

# The Consistency of DOLCE

## The DOLCE Ontology

DOLCE is the 'Descriptive Ontology for Linguistic and Cognitive Engineering', developed at the Laboratory For Applied Ontology (LOA) in Trento (Gangemi et al. 2002; Masolo et al. 2003). It contains several hundred axioms, formulated in first-order logic.[5]

The complexity of the DOLCE ontology stems from the fact that it combines several (non-trivial) formalised ontological theories into one theory, viz. the theories of essence and identity, parts and wholes (mereology), dependence, composition and constitution, as well as properties and qualities. Fig. 6 shows a graph of some interesting such subtheories of DOLCE. The taxonomy of DOLCE's concepts is shown in Fig. 5.




| | |
|---|---|
| APO | Agentive Physical Object |
| AS | Arbitrary Sum |
| ASO | Agentive Social Object |
| ED | Endurant |
| F | Feature |
| M | Amount of Matter |
| MOB | Mental Object |
| PED | Physical Endurant |
| SC | Society |
| SAG | Social Agent |
| SOB | Social Object |
| POB | Physical Object |

Figure 5: DOLCE's taxonomy.

## Building an Architectural Specification for DOLCE

We have carefully analysed the DOLCE theory and have designed an architectural specification for it. In the process of this design, we had to re-arrange the architectural decomposition several times in order to find an optimal decomposition. The forces to be balanced out are the following:

- both the theories of the individual units and the theory extensions (for parametrised units) should be small enough in order to keep the consistency and conservativity checks feasible;
- the theory extensions of the parametrised units must be large enough to make the conservativity checks work (that is, if a new symbol is introduced, the theory extension should contain all essential constraints for that symbol);
- the theory extensions must be large enough to guarantee the amalgamability conditions.

[5]There are also versions of DOLCE including some axioms using modal logic, but they do not concern the heart of DOLCE, and leaving them out does not in any way trivialise the consistency problem (Masolo et al. 2003).

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 the above mentioned *abstraction barrier* between the different units. Recall that when providing the realisation of a parametrised 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 parametrised models, in the end, there is a great pay-off: namely, the overall model construction has been split into 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 at designing an architectural specification for DOLCE largely followed the specification structure of DOLCE as shown in Fig. 6. 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 subtheories, we faced the problem that the specification DEPENDENCE introduces subtle dependencies between various parts of DOLCE's taxonomy.

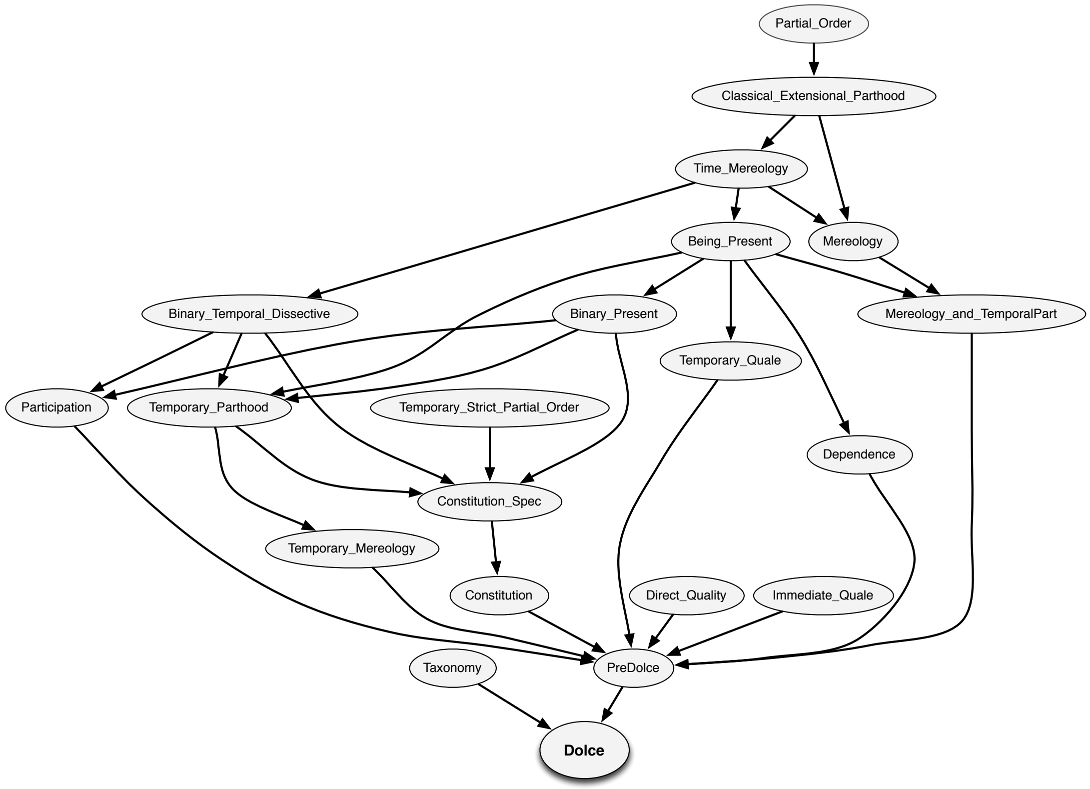



Figure 6: Structure of DOLCE's subtheories.

## (Timed) Mereology: Bottom Up vs. Top Down

Hence, we needed to completely restructure the architectural specification. Most importantly, the model for TEMPORARY_PARTHOOD cannot be constructed for the top concepts in the taxonomy and then be inherited to 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 *subsorted* 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 $\mathbf{SubFOL}^=$ used in CASL, see (Astesiano et al. 2002) for details. (Note that if needed, from a subsorted model it is straightforward to construct a single-sorted model by mapping subsorts into predicates.)

The temporal mereology in TEMPORARY_PARTHOOD 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:

$\forall\, x : s;\ y : s;\ t : T$
$\bullet\ tOv(x, y, t) \Leftrightarrow \exists\, z : s \bullet tP(z, x, t) \wedge tP(z, y, t)$
$\forall\, z : s;\ x : s;\ y : s$
$\bullet\ tSum(z, x, y)$
$\quad \Leftrightarrow \forall\, w : s;\ t : T \bullet tOv(w, z, t) \Leftrightarrow tOv(w, x, t) \vee tOv(w, y, t)$

Here, $T$ is DOLCE's sort for time, while $s$ is a generic sort that is instantiated by various categories that require temporal parthood (see below for discussion). Moreover, mereological sums are required to exist:

$\forall\, x, y : s;\ \bullet \exists\, z : s \bullet tSum(z, x, y)$

(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_PARTHOOD 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_PARTHOOD 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_PARTHOOD_NOSUM of TEMPORARY_PARTHOOD 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_PARTHOOD for the leaves of the taxonomy below ED: SC, SAG, NASO, APO, NAPO, MOB, F, and M. Here, we may take models of different cardinality and structure for different sorts;
- at inner nodes below ED, we take the disjoint union of subconcepts, ending up in a model 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 (possibly nested) sum/difference does already exist in PED or NPED, respectively. Otherwise, it is put into AS (arbitrary sum). This corresponds to a reduced powerset construction (wrt the equivalence relation just sketched), and is illustrated in Fig. 7. Here, certain formal sums are being identified, such as $n_1 + n_3$ and $n_2 + n_3$; the double-headed arrows from NPED and PED to AS illustrate sums that have to go into AS because they do not already exist in the respective sort, such as $p_1 + p_2$ in PED. All mixed sums (single-headed arrows that meet) go into AS as well.

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 could define a model in a top-down manner. However, as the designers told us, the general DOLCE methodology is to start with instantiating the leaves of the taxonomy, that is, a model should be built in a bottom-up manner. In any case, PD is specified to be the disjoint union of its two subconcepts EV (event) and STV (stative). This means that one has 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 could be considered conceptually wrong. Concrete refinements of DOLCE's perdurants can make precise which sums are considered events and statives respectively, and most such refinements of perdurants that largely follow linguistic criteria give such a complete classification. However, this comprises an artificial restriction as, indeed, an ontologist might decide that certain sums of an event and a stative should neither be classified as an event, nor as a stative. Alternatively, DOLCE could refrain from postulating arbitrary sums of perdurants.

The architectural design is summed up in the architectural specification PARTHOOD_MODEL in Fig. 8 (and the full architectural specification can be found in the Appendix).

It actually corresponds to the lower-most branching PARTHOODM of the refinement tree in Fig. 2. The notation TEMPORARY_PARTHOOD WITH S ↦ SC renames sort $s$ in TEMPORARY_PARTHOOD appropriately. The notation

**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.

```
arch spec PARTHOOD_MODEL =
    units TM : TIME_MEREOLOGY;
    TP_SC : TEMPORARY_PARTHOOD_ETERNAL[sort SC]
            given TM;
    TP_SAG : TEMPORARY_PARTHOOD_ETERNAL[sort SAG]
            given TM;
    TP_NASO :
            {TEMPORARY_PARTHOOD with s ↦ NASO
            and  ONESIDE_GENERIC_DEPENDENCE
                 with s1 ↦ NASO, s2 ↦ SC
            } given TP_SC;
    TP_APO :
            {TEMPORARY_PARTHOOD_ETERNAL[sort APO]
            and  ONESIDE_GENERIC_DEPENDENCE
                 with s1 ↦ SAG, s2 ↦ APO } given TP_SAG;
    TP_F : TEMPORARY_PARTHOOD_ETERNAL[sort F]
            given TM;
    TP_NAPO :
            {TEMPORARY_PARTHOOD with s ↦ NAPO
            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_NPOB :
            {TP_NOSUM_ETERNAL[sort NPOB]
            and  free type NPOB ::= sort SOB | sort MOB
            } given TP_SOB, TP_MOB;
    TP_M : {TEMPORARY_PARTHOOD with s ↦ M}
            given TP_POB;
    TP_NPED :
            {TP_NOSUM_ETERNAL[sort NPED]
            and  sort   NPOB < NPED
            } given TP_NPOB;
    TP_PED :
            {TP_NOSUM_ETERNAL[sort PED]
            and  free type PED ::= sort POB | sort M | sort F
            } given TP_M;
    TP_ED :
            {TEMPORARY_PARTHOOD with s ↦ ED
            and  esort  AS
                 free type ED ::= sort PED | sort NPED | sort AS
            } given TP_PED, TP_NPED;
    ...
    CEP_PD :
            {{CLASSICAL_EXTENSIONAL_PARTHOOD and  sort   s}
            with s ↦ PD
            then free type PD ::= sort EV | sort STV
            } given TP_ED, EV, STV;
    PARTICIP : PARTICIPATION given CEP_PD;
    MEREOLOGY_AND_TEMPORALPARTPD :
            MEREOLOGY_AND_TEMPORALPART given PARTICIP
    result MEREOLOGY_AND_TEMPORALPARTPD
end
```

Figure 8: Architectural specification for mereology.

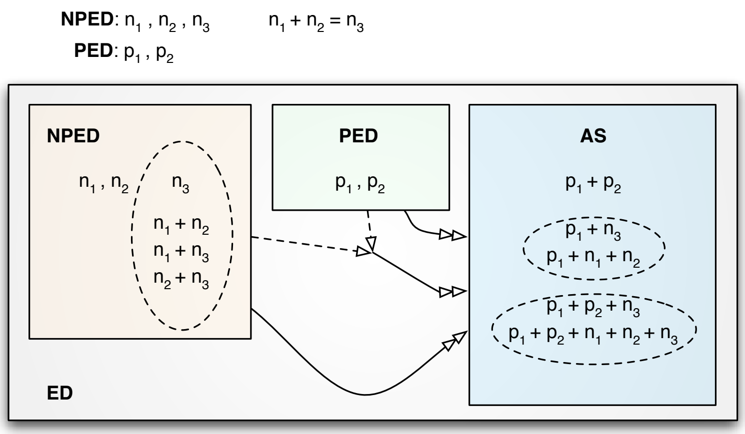



Figure 7: Reduced powerset construction for the temporal sum on sort ED.

## Strengthening Specifications

Another lesson learned from the subtle interactions introduced by the specification 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 DEPENDENCE, we could rely on stronger assumptions for the interpretation of 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 $F : SP \to SP_1$ for some $SP_1$. That is, specifications of formal parameter and actual parameter match exactly.

**Theorem 1.** *Let $ASP$ be an exactly matching architectural specification, and $SP$, $SP'$ be structured specifications such that $SP' \models SP$ (where $\models$ means logical entailment, i.e. every model of $SP'$ is also a model of $SP$). Let $ASP'$ be obtained by replacing every occurrence of specification $SP$ in $ASP$ by $SP'$. Then the consistency of $ASP'$ implies that of $ASP$.*

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 parametrised units.

An example is again TEMPORARY_PARTHOOD, or more precisely, its subtheory TP_NOSUM. We have strengthened this specification to TP_NOSUM_ETERNAL, which requires that the binary predicate $PRE(x, t)$ ('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:

$$
\begin{aligned}
\textbf{pred}\ \ & SD(x : MOB; y : APO) \\
& \Leftrightarrow (\exists\, t : T \bullet PRE(x, t)) \\
& \quad \wedge \forall\, t : T \bullet PRE(x, t) \Rightarrow PRE(y, t);
\end{aligned}
$$

SD introduces dependencies between different sorts, here MOB and APO, regarding temporal presence. Intuitively, $SD(x, y)$ 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 and the predicate SD, and is employed for the theories of DEPENDENCE as well as TEMPORARY_PARTHOOD. Given the eternal object in APO we can give the following definitions:

$$
\begin{gathered}
\forall x : MOB; y : APO.SD(x, y) \iff \forall t : T.PRE(y, t) \\
\forall x : MOB; t : T.PRE(x, t) \iff true
\end{gathered}
$$

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 $T''$ 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 38 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, 37 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 temporary parthood.

We stress that the choice of the details of the models can be made independently for each of the 38 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 the disjoint union of PED, NPED and AS, the interpretations of PED and NPED are already given, and only AS is newly interpreted (see Fig. 7).[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 instantiating 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 encountered a problem which could be considered a design flaw in DOLCE by some since it restricts possible refinements; importantly, such issues would probably not have been found with monolithic methods since they do not affect 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.


### Acknowledgements.

Work on this paper has been supported by the Vigoni program of the DAAD, by the DFG-funded collaborative research center SFB/TR 8 Spatial Cognition and by the German Federal Ministry of Education and Research (Project 01 IW 07002 FormalSafe). We thank John Bateman, Joana Hois, Claudio Masolo, and Stefano Borgo for discussing DOLCE, Christian Maeder for implementation work around the Heterogeneous Tool Set, and especially Dominik Lücke for earlier work on the consistency problem of DOLCE.


[6]All formal specifications, including DOLCE itself, its architectural specification, and the 38 model constructions can be found at the following URL `http://www.dfki.de/sks/hets/dolce`

## References


Astesiano, E.; Bidoit, M.; Kirchner, H.; Krieg-Brückner, B.; Mosses, P. D.; Sannella, D.; and Tarlecki, A. 2002. CASL: The Common Algebraic Specification Language. *Theoretical Computer Science* 286(2):153–196.

Baumgartner, P., and Tinelli, C. 2003. The Model Evolution Calculus. In Baader, F., ed., *CADE-19*, volume 2741 of *LNAI*, 350–364. Springer.

Baumgartner, P.; Fuchs, A.; and Tinelli, C. 2004. Darwin: A Theorem Prover for the Model Evolution Calculus. In Schulz, S.; Sutcliffe, G.; and Tammet, T., eds., *ESFOR-04, IJCAR Workshop*, ENTCS.

Bidoit, M.; Sannella, D.; and Tarlecki, A. 2002. Architectural specifications in CASL. *Formal Aspects of Computing* 13:252–273.

Gangemi, A.; Guarino, N.; Masolo, C.; Oltramari, A.; and Schneider, L. . 2002. Sweetening Ontologies with DOLCE. In Gómez-Pérez, A., and Benjamins, V. R., eds., *EKAW-02*, LNCS Vol. 2473, 166–181. Springer.

Grenon, P.; Smith, B.; and Goldberg, L. 2004. Biodynamic Ontology: Applying BFO in the Biomedical Domain. In Pisanelli, D. M., ed., *Ontologies in Medicine*. IOS. 20–38.

Herre, H. 2010. General Formal Ontology (GFO): A Foundational Ontology for Conceptual Modelling. In Poli, R.; Healy, M.; and Kameas, A., eds., *Theory and Applications of Ontology: Computer Applications*. Springer. 297–345.

Horrocks, I., and Voronkov, A. 2006. Reasoning support for expressive ontology languages using a theorem prover. In *FoIKS-06*, number 3861 in LNCS, 201–218. Springer.

Klin, B.; Hoffman, P.; Tarlecki, A.; Schröder, L.; and Mossakowski, T. 2001. Checking Amalgamability Conditions for CASL Architectural Specifications. In *MFCS-01*, 451–463. London, UK: Springer.

Kutz, O.; Lücke, D.; and Mossakowski, T. 2008. Modular Construction of Models—Towards a Consistency Proof for the Foundational Ontology DOLCE. In *First International Workshop on Foundations of Computer Science as Logic-Related, (ICTAC-08)*.

Masolo, C.; Borgo, S.; Gangemi, A.; Guarino, N.; and Oltramari, A. 2003. Ontology Library—WonderWeb Deliverable D18. Technical report, Laboratory For Applied Ontology - ISTC-CNR, Trento, Italy.

Mossakowski, T.; Maeder, C.; and Lüttich, K. 2007. The Heterogeneous Tool Set. In Grumberg, O., and Huth, M., eds., *TACAS-07*, volume 4424 of *LNCS*, 519–522.

Niles, I., and Pease, A. 2001. Towards a Standard Upper Ontology. In *FOIS-01*, 2–9. New York, NY, USA: ACM.

Voronkov, A. 2006. Inconsistencies in Ontologies. In *JELIA-06*, 19.

Weber, T. 2005. Bounded model generation for Isabelle/HOL. volume 125 of *ENTCS*, 103–116.

Weidenbach, C.; Brahm, U.; Hillenbrand, T.; Keen, E.; Theobalt, C.; and Topic, D. 2002. SPASS version 2.0. In Voronkov, A., ed., *Automated Deduction, CADE-18*, volume 2392 of *LNCS Vol.* , 275–279. Springer.

# Appendix. Architectural Specification

**library** ONTOLOGY/DOLCE/DOLCEMODEL **version** 1.0

**from** ONTOLOGY/DOLCE/DOLCESIMPL_ESORT **get**
TAXONOMY, PARTIAL_ORDER, EXT_PARTIAL_ORDER,
EXT_OVERLAP_or_CONNECTION,
CLASSICAL_EXTENSIONAL_PARTHOOD,
CLASSICAL_EXTENSIONAL_PARTHOOD_NO,
TEMPORARY_PARTHOOD_ETERNAL, TIME_MEREOLOGY,
UNARY_TEMPORAL_DISSECTIVE, BEING_PRESENT,
MEREOLOGY, MEREOLOGY_AND_TEMPORALPART,
BINARY_PRESENT, BINARY_TEMPORAL_DISSECTIVE,
TEMPORARY_PARTIAL_ORDER,
TEMPORARY_STRICT_PARTIAL_ORDER,
TEMPORARY_PARTHOOD_NO, TEMPORARY_PARTHOOD,
TP_NOSUM_ETERNAL, TEMPORARY_MEREOLOGY,
CONSTITUTION_SPEC,
CONSTANTLY_GENERICALLY_CONSTITUTED, CONSTITUTION,
PARTICIPATION, DIRECT_QUALITY_SPEC, DIRECT_QUALITY,
IMMEDIATE_QUALE_SPEC, IMMEDIATE_QUALE,
TEMPORARY_QUALE_SPEC, TEMPORARY_QUALE,
SPECIFIC_DEPENDENCE, MUTUAL_SPECIFIC_DEPENDENCE,
ONESIDE_SPECIFIC_DEPENDENCE, GENERIC_DEPENDENCE,
MUTUAL_GENERIC_DEPENDENCE,
ONESIDE_GENERIC_DEPENDENCE, DEPENDENCE,
STRONGLY_NON_EMPTY, CUMULATIVE, ANTI_CUMULATIVE,
HOMEOMEROUS, ANTI_HOMEOMEROUS, ATOMIC,
ANTI_ATOMIC, PREDOLCE, DOLCE

**arch spec** PARTHOOD_MODEL =
**units** TM : *Time_Mereology*;
TP_SC : TEMPORARY_PARTHOOD_ETERNAL[**sort** *SC*] **given** TM;
TP_SAG : TEMPORARY_PARTHOOD_ETERNAL[**sort** *SAG*] **given** TM;
TP_NASO :
{TEMPORARY_PARTHOOD **with** *s* ↦ *NASO*
**and** ONESIDE_GENERIC_DEPENDENCE
**with** *s1* ↦ *NASO*, *s2* ↦ *SC*
}
**given** TP_SC;
TP_APO :
{TEMPORARY_PARTHOOD_ETERNAL[**sort** *APO*]
**and** ONESIDE_GENERIC_DEPENDENCE
**with** *s1* ↦ *SAG*, *s2* ↦ *APO*
}
**given** TP_SAG;
TP_F : TEMPORARY_PARTHOOD_ETERNAL[**sort** *F*] **given** TM;
TP_NAPO :
{TEMPORARY_PARTHOOD **with** *s* ↦ *NAPO*
**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*
**and** ONESIDE_SPECIFIC_DEPENDENCE
**with** *s1* ↦ *MOB*, *s2* ↦ *APO*
}
**given** TP_APO;
TP_SOB :
{TP_NOSUM_ETERNAL[**sort** *SOB*]
**and** **free type** *SOB* ::= *sort ASO* | *sort NASO*
}
**given** TP_NASO, TP_ASO;
TP_POB :
{TP_NOSUM_ETERNAL[**sort** *POB*]
**and** **free type** *POB* ::= *sort APO* | *sort NAPO*
}
**given** TP_MOB, TP_NAPO;
TP_NPOB :
{TP_NOSUM_ETERNAL[**sort** *NPOB*]
**and** **free type** *NPOB* ::= *sort SOB* | *sort MOB*
}
**given** TP_SOB, TP_MOB;
TP_M : {TEMPORARY_PARTHOOD **with** *s* ↦ *M*} **given** TP_POB;
TP_NPED :
{TP_NOSUM_ETERNAL[**sort** *NPED*]
**and** **sort** *NPOB* < *NPED*
}
**given** TP_NPOB;
TP_PED :
{TP_NOSUM_ETERNAL[**sort** *PED*]
**and** **free type** *PED* ::= *sort POB* | *sort M* | *sort F*
}
**given** TP_M;
TP_ED :
{TEMPORARY_PARTHOOD **with** *s* ↦ *ED*
**and** **esort** *AS*
**free type** *ED* ::= *sort PED* | *sort NPED* | *sort AS*
}
**given** TP_PED, TP_NPED;
ACC :
{CLASSICAL_EXTENSIONAL_PARTHOOD_NO **with** *s* ↦ *ACC*
};
ACH :
{CLASSICAL_EXTENSIONAL_PARTHOOD_NO **with** *s* ↦ *ACH*
};
EV :
{CLASSICAL_EXTENSIONAL_PARTHOOD_NO **with** *s* ↦ *EV*
**then free type** *EV* ::= *sort ACC* | *sort ACH*
}
**given** ACC, ACH;
ST :
{CLASSICAL_EXTENSIONAL_PARTHOOD_NO **with** *s* ↦ *ST*};
PRO :
{CLASSICAL_EXTENSIONAL_PARTHOOD_NO **with** *s* ↦ *PRO*};
STV :
{CLASSICAL_EXTENSIONAL_PARTHOOD_NO **with** *s* ↦ *STV*
**then free type** *STV* ::= *sort ST* | *sort PRO*
}
**given** ST, PRO;
CEP_PD :
{{CLASSICAL_EXTENSIONAL_PARTHOOD **and** **sort** *s*}
**with** *s* ↦ *PD*
**then free type** *PD* ::= *sort EV* | *sort STV*
}
**given** TP_ED, EV, STV;
PARTICIP : *Participation* **given** CEP_PD;
MEREOLOGY_AND_TEMPORALPARTPD :
*Mereology_and_TemporalPart* **given** PARTICIP

```
    result MEREOLOGY_AND_TEMPORALPARTPD
end

arch spec CONSTITUTION_MODEL =
    units PARTHOODM : arch spec Parthood_Model;
    TP_ED = PARTHOODM; TP_NPED = PARTHOODM;
    DEPENDENCEAQNPED :
    {MUTUAL_SPECIFIC_DEPENDENCE
    with s1 ↦ AQ, s2 ↦ NPED
    and  DIRECT_QUALITY_SPEC with s1 ↦ AQ, s2 ↦ NPED
    }
    given TP_ED, TP_NPED;
    CONSTITUTIONPD : {CONSTITUTION_SPEC with s ↦ PD}
    given PARTHOODM;
    CONSTITUTIONPED :
    {CONSTANTLY_GENERICALLY_CONSTITUTED
    with s ↦ PED, s1 ↦ NAPO, s2 ↦ M
    and  CONSTANTLY_GENERICALLY_CONSTITUTED
         with s ↦ PED, s1 ↦ APO, s2 ↦ NAPO
    }
    given PARTHOODM;
    CONSTITUTIONNPED :
    {CONSTANTLY_GENERICALLY_CONSTITUTED
    with s ↦ NPED, s1 ↦ SC, s2 ↦ SAG
    }
    given PARTHOODM, DEPENDENCEAQNPED;
    CONSTITUTIONED : {CONSTITUTION_SPEC with s ↦ ED}
    given TP_ED;
    CONSTITUTION =
    CONSTITUTIONPD
    and CONSTITUTIONPED
    and CONSTITUTIONNPED
    and CONSTITUTIONED
    result CONSTITUTION
end

arch spec PREDOLCE_MODEL =
    units CONSTITUTION : arch spec Constitution_Model;
    TM = CONSTITUTION; TP_EDSAR = CONSTITUTION;
    TP_ED = TP_EDSAR; TP_PED = CONSTITUTION;
    PARTICIP = CONSTITUTION;
    DEPENDENCEAQNPED = CONSTITUTION;
    MEREOLOGYAR = CONSTITUTION;
    IMMEDIATEQUALE :
    {{IMMEDIATE_QUALE
    then sorts  TL < TQ;
                T < TR
    }
    and  DIRECT_QUALITY_SPEC with s1 ↦ TQ, s2 ↦ PD
    then esort  TL < TQ
         ∀ y : PD • ∃ x : TL • dqt(x, y)
    }
    given TM;
    BINTEMPDISS :
    {BINARY_TEMPORAL_DISSECTIVE with s1 ↦ S, s2 ↦ SL}
    given TP_EDSAR;
    DEPENDENCEPQPED :
    {{MUTUAL_SPECIFIC_DEPENDENCE
    with s1 ↦ PQ, s2 ↦ PED
    then sort   SL < PQ
    }
    and  DIRECT_QUALITY_SPEC with s1 ↦ PQ, s2 ↦ PED
    then esort  SL < PQ
         ∀ y : PED • ∃ x : SL • dqt(x, y)
    }
    given TP_PED, BINTEMPDISS;
    DEPENDENCETQPD :
    {MUTUAL_SPECIFIC_DEPENDENCE with s1 ↦ TQ, s2 ↦ PD}
    given PARTICIP, IMMEDIATEQUALE;
    BEINGPRESENTEDORDP :
    {BEING_PRESENT with s ↦ EDorPDorQ
    and  free type Q ::= sort TQ | sort PQ | sort AQ
         free type EDorPDorQ ::= sort Q | sort PD | sort ED
    }
    given DEPENDENCETQPD, DEPENDENCEPQPED,
    DEPENDENCEAQNPED
    TEMPQUALEPR :
    {TEMPORARY_QUALE_SPEC with s1 ↦ PR, s2 ↦ PQ
    then sort   S < PR
    }
    given BEINGPRESENTEDORDP, BINTEMPDISS;
    TEMPQUALEAR :
    {TEMPORARY_QUALE_SPEC with s1 ↦ AR, s2 ↦ AQ} given
    BEINGPRESENTEDORDP, BINTEMPDISS;
    TEMPQUALES :
    {TEMPORARY_QUALE_SPEC with s1 ↦ S, s2 ↦ SL} given
    BEINGPRESENTEDORDP, BINTEMPDISS;
    TEMPQUALE =
    TEMPQUALEPR and TEMPQUALEAR and TEMPQUALES;
    BINTEMPDISPR :
    {BINARY_TEMPORAL_DISSECTIVE with s1 ↦ PR, s2 ↦ PQ}
    given TEMPQUALEPR;
    CEP_S :
    {CLASSICAL_EXTENSIONAL_PARTHOOD with s ↦ S} given
    BINTEMPDISS;
    BINTEMPDISAR :
    {BINARY_TEMPORAL_DISSECTIVE with s1 ↦ AR, s2 ↦ AQ}
    given DEPENDENCEAQNPED, MEREOLOGYAR
    DIRECTQUALITY : Direct_Quality given
    BEINGPRESENTEDORDP, BINTEMPDISS, CONSTITUTION,
    TEMPQUALE, BINTEMPDISPR, BINTEMPDISAR, CEP_S
    result
    CONSTITUTION
    and DIRECTQUALITY
    and TEMPQUALE
    and BINTEMPDISPR
    and BINTEMPDISAR
    and CEP_S
end

arch spec DOLCE_MODEL =
    units PREDOLCE : arch spec PreDolce_Model;
    TAX : {TAXONOMY and  pred  K : ED × ED × T} given
    PREDOLCE
    result TAX
end

refinement DOLCE_REF =
        Dolce refined to arch spec Dolce_Model
end
```