Appendix I: Ontological Criteria for Evaluating the Semantics of Conceptual Modeling Grammars

To evaluate the ontological clarity and completeness of a conceptual modeling (CM) grammar, the grammar's constructs are mapped against the constructs in a benchmark ontology. Wand and Weber (1993, pp. 228–233) argue three conditions undermine *ontological clarity*:

- *Construct overload*: A single grammatical construct maps to two or more ontological constructs.
- *Construct excess*: A grammatical construct does not map to any ontological construct.
- *Construct redundancy*: Two or more grammatical constructs map to a single ontological construct.

Wand and Weber (1993, pp. 226–228) argue that *ontological completeness* is undermined when *construct deficit* exists—an ontological construct is not represented by any grammatical construct.

Our notion of ontological clarity differs in two ways from Wand and Weber's notion of ontological clarity. First, we do not see construct redundancy as undermining ontological clarity. Even though two or more grammatical constructs map to a single ontological construct, the meaning of each redundant construct is still clear. Whether the redundant constructs cause confusion among users of the grammar is an issue of pragmatics and not semantics.

Second, our concept of an always ontologically clear grammar is new. The traditional view of ontological clarity focuses on creating a mapping between individual constructs in a grammar and individual constructs in an ontological benchmark. An always-clear grammar goes further to consider the grammar's production rules. In an always-clear grammar, not only is each construct clear, but also the grammar's production rules will not allow unclear combinations of constructs in the grammar to be generated. We are not aware of any research on this property of grammars to date, but it could be viewed as an extension of the line of work begun by Evermann and Wand (2005, p. 148).

Note that ontological completeness applies only to grammars and not scripts. The reason is that we cannot know if a script includes all the grammatical constructs needed to represent all relevant ontological constructs in a domain without knowing which ontological constructs are relevant in the domain description. The latter is a pragmatic issue that lies outside the scope of ontological analysis.

Note, also, that the four ontological criteria for evaluating CM grammars are all independent of each other. The reason is that each criterion is based upon a different outcome of the mapping between the set of constructs in a CM grammar and the set of constructs in a benchmark ontology.

Appendix II: Formal Results

In this Supplementary Appendix, we provide a rigorous proof that the grammar ERM-R is always-precise. In the main paper, we used the lay term "precise" for what logicians mean by "complete" to avoid confusion with the latter term's more usual meaning in the conceptual modeling literature. However, we expect readers of this Appendix to be more familiar with the logician's terminology—and we refer here to work by logicians—so we adjust our vocabulary accordingly: wherever we refer to "completeness" in this Appendix, we mean (a) what logicians mean by "completeness" and (b) what we mean by "preciseness" in the main paper.

Syntax

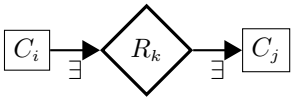
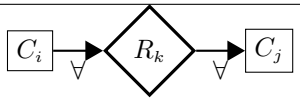
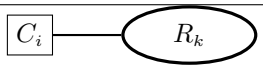
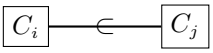
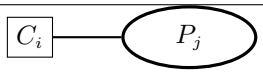
ERM-R is a visual, two-dimensional language. Nonetheless, we can translate scripts in ERM-R to a linear language (like English or FOL), which greatly simplifies the proofs to follow. We proceed by producing a linear "model-description" listing the elements of a visual ER diagram and their interconnections. This process loses none of the information contained in the ER diagram. Thus, we can use model-descriptions as substitutes for ERM-R scripts. Our scripts are always finite, so we can replace them with sentences giving instructions for constructing the grammar: "There are boxes labeled 'C_1' and 'C_2'; there is a line labeled 'R_1' from the box labeled 'C_1' to the box labeled 'C_2'; ...." By giving a syntax for the "language" of such translations—by giving rules determining what is and what is not an acceptable description of an ERM-R script—we effectively give a syntax for the original, untranslated language.

Table 1 gives a synopsis of our modified grammar ERM-R. The first column contains a list of visual constructs of ERM-R. The second column contains the translation of each construct into our language of model-descriptions. The third and fourth columns contain the informal and formal semantics intended for each construct.

We give a syntax for ERM-R by giving a syntax for model-descriptions in Backus-Naur form (BNF).

$$
\begin{aligned}
< \text{description} > ::= \; & < \text{description} > ; < \text{description} > \mid \\
& < \text{class} > \rightarrow_{\exists} < \text{relation} > < \text{class} > \mid \\
& < \text{class} > \rightarrow_{\forall} < \text{relation} > < \text{class} > \mid \\
& < \text{class} > \subseteq < \text{class} > \mid < \text{class} > < \text{property} > \\
& < \text{class} > < \text{relation} >
\end{aligned}
$$

Table 1 Vocabulary elements of ERM-R.

Visual construct	Linear description	Informal semantics	Formal semantics
	$C_i \rightarrow_\exists R_k C_j$	All members of C_i bear R_k to at least one member of C_j, and vice versa.	If $x \in C_i$ then $\exists y \in C_j$ with $R_k xy$; if $y \in C_j$ then $\exists x \in C_i$ with $R_k xy$
	$C_i \rightarrow_\forall R_k C_j$	All members of C_i bear R_k to all members of C_j.	If $x \in C_i$ and $y \in C_j$ then $R_k xy$
	$C_i R_k$	Everything in C_i bears R_k to itself.	If $x \in C_i$ then $R_k xx$
	$C_i \subseteq C_j$	C_i is a subset of C_j.	$C_i \subseteq C_j$ (If $x \in C_i$ then $x \in C_j$)
	$C_i P_j$	Everything in C_i has P_j.	If $x \in C_i$ then $P_j x$

$$< \text{class} > ::= C_1 | C_2 | \ldots | C_n$$

$$< \text{relation} > ::= R_1 | R_2 | \ldots | R_m$$

$$< \text{property} > ::= P_1 | P_2 | \ldots | P_l$$

We design the syntax here for compactness rather than readability. Some brief comments on the syntax are in order. Semicolons conjoin component sentences of a description. When two classes are connected by a relation, we need to know what the relation is, its direction, and whether it is a ∀-type or ∃-type arrow. ("$\rightarrow_\exists$" or "$\rightarrow_\forall$"). Our description syntax does not allow sentences of the form, "There is a box labeled 'C_1'." This is because we assume that all classes will have some attributes—either a connection to another class via subclasshood or a relation arrow, or some intrinsic attribute of its members. Therefore, all classes will appear in one of those sentences in a model's description, making sentences of the form, "There is a box labeled 'C_1'," superfluous. From a syntactic perspective, there are no further constraints on the scripts (diagrams) we can produce: any combination of attributes, relations, and subclass relations is acceptable.

We can now define some syntactic terms that will be useful in giving a semantics below. We say that a class C_i is a *subclass* of a class C_j, and C_j is a *superclass* of C_i, and write $C_i \subseteq C_j$, iff the model-description contains, for some (finite) sequence of classes $C_1, \ldots, C_n$, the sentences "$C_i \subseteq C_1$," "$C_1 \subseteq C_2$," ..., and "$C_n \subseteq C_j$." By convention, we consider each class to be a subclass (and a superclass) of itself. We say that C_i is a *proper subclass* of C_j, and write $C_i \subset C_j$, iff C_i is a subclass of C_j and C_j is not a subclass of C_i. If C_i is a subclass of C_j but not a proper subclass, then we write $C_i = C_j$. Say a class is *basic* iff it has no (non-empty) proper subclasses and *non-basic* otherwise. Note that it is possible to have basic classes C_i and

C_j with $i \neq j$ but with $C_i = C_j$. Finally, say a class C_j is an *immediate subclass* of a class C_i iff $C_j \subset C_i$ and there is no class C_k such that $C_j \subset C_k \subset C_i$.

To deal conveniently with symmetric relations, we also add some technically superfluous sentences to our model-descriptions. If R_k is symmetric, and the model-description contains a sentence "$C_i \rightarrow_\exists R_k C_j$," then we add the sentence "$C_j \rightarrow_\exists R_k C_i$". Likewise, if the model-description contains a sentence "$C_i \rightarrow_\forall R_k C_j$," then we add the sentence "$C_j \rightarrow_\forall R_k C_i$". These additions have no material impact on the semantics of the original conceptual model, but they simplify the statement of the semantics in the next section.

Semantics

We give a semantics for ERM-R by giving a translation of any given model in ERM-R to a finitely axiomatized theory of FO^2. Because, as a fragment of FOL, FO^2 has a well-defined semantics (see, e.g., Shoenfield 1967), this translation amounts to a semantics for ERM-R.

Given a conceptual modeling script M in ERM-R, we build its translation into FOL, T_M, that we sometimes call "the (first-order) theory of M," as follows. The language $\mathcal{L}$ of the theory has as its nonlogical symbols finitely many unary predicates $C_1, C_2, \ldots$, and $P_1, P_2, \ldots$, and finitely many binary predicates $R_1, R_2, \ldots$; these are, respectively, exactly those class labels, attribute labels, and relation labels that occur in M. In general, $C_i x$ is to be intepreted as saying that x is a member of class C_i; $P_j x$ as saying that x has property P_j; and $R_k xy$ as saying that x bears relation R_k to y. A guiding principle in building T_M is that *the absence of a claim is a claim of absence*: for example, if a box C_1 does not have a circle labeled P_1 (where P_1 does occur somewhere in the script), then we interpret the script as claiming that the members of the class C_1 do not possess the property P_1. We now build the axioms of the theory T_M.

We have an axiom of exhaustivity, saying that no objects exist outside of the classes depicted in the model. It is enough to say that no objects exist outside the basic classes. So, where $C_{i_1}, \ldots, C_{i_j}$ are all of the basic classes, the exhaustivity axiom is:

$$\vdash_{T_M} C_{i_1} x \vee \ldots \vee C_{i_j} x$$

Next we add axioms of non-triviality, saying that the classes depicted are non-empty. Again, it is enough to say that the basic classes are non-empty. For each basic class C_i, we add:

$$\vdash_{T_M} \exists x C_i x$$

We also need axioms to say which of the basic classes are disjoint. For each pair of basic classes C_i and C_j with $i \neq j$, we add either a class equivalence or a class exclusion axiom. If $C_i = C_j$, then we add a class equivalence axiom:

$$\vdash_{T_M} C_i x \leftrightarrow C_j x$$

Otherwise, we add a class exclusion axiom:

$$\vdash_{T_M} \neg(C_i x \wedge C_j x)$$

Likewise, we need relation exclusion axioms, to capture the conventional constraint on relations. For each pair of distinct relation symbols R_i and R_j, we add both of the following:

$$\vdash_{T_M} \neg(R_i xy \wedge R_j xy)$$

$$\vdash_{T_M} \neg(R_i xy \wedge R_j yx)$$

Similarly, according to whether R_k is symmetric or asymmetric, we add either

$$\vdash_{T_M} R_k xy \rightarrow R_k yx$$

or

$$\vdash_{T_M} R_k xy \rightarrow \neg R_k yx$$

Now, we characterize non-basic classes by listing the subclasses of which they are composed. If C_i is non-basic, and its immediate subclasses are $C_{j_1}, \ldots, C_{j_k}$, then we add the following composition axiom:

$$\vdash_{T_M} C_i x \leftrightarrow C_{j_1} x \vee \ldots \vee C_{j_k} x$$

Thus, a non-basic class is characterized by the basic subclasses of which it is ultimately composed.

Finally, we add characterization axioms, giving the properties attributed to members of each basic class. For each basic class C_i, the characterization axiom is:

$$\vdash_{T_M} C_i x \rightarrow \Phi_i x \wedge \neg \Psi_i x,$$

where $\Phi_i x$ and $\Psi_i x$ are to be defined as follows.

To define the formulas $\Phi_i x$ and $\Psi_i x$ that do the real work of characterizing a basic class C_i, we must look at the relations and properties attributed by the model to members of C_i. $\Phi_i x$ will be a conjunction listing the properties that members of C_i must have, and $\Psi_i x$ will be a disjunction listing the properties that members of C_i lack. We build $\Phi_i x$ and $\Psi_i x$ by going through all the nonlogical symbols of $\mathcal{L}$ other than class symbols and checking whether and how they apply to the class C_i in the model.

For each predicate symbol P_k, if the model-description contains a sentence "$C_j P_k$," where C_j is any superclass of C_i, then add $P_k x$ as a conjunct of $\Phi_i x$; otherwise, add the same formula as a disjunct of $\Psi_i x$.

For each relation symbol R_k, we have several cases to check. If the model-description contains a sentence "$C_j R_k$," where C_j is any superclass of C_i, then add $R_k xx$ as a conjunct of $\Phi_i x$; otherwise, add the same formula as a disjunct of $\Psi_i x$.

If the model-description contains a sentence "$C_j \rightarrow_\forall R_k C_g$," where C_j is any superclass of C_i, then add $\forall y(C_l y \rightarrow R_k xy)$ as a conjunct of $\Phi_i x$; otherwise, add the same formula as a disjunct of $\Psi_i x$. If the model-description contains a sentence "$C_g \rightarrow_\forall R_k C_j$," where C_j is any superclass of C_i, then add $\forall y(C_g y \rightarrow R_k yx)$ as a conjunct of $\Phi_i x$; otherwise, add the same formula as a disjunct of $\Psi_i x$.

If the model-description contains a sentence "$C_j \rightarrow_\exists R_k C_g$," where C_j is any superclass of C_i, and it does not contain any sentence "$C_j \rightarrow_\forall R_k C_h$," where C_h is a superclass of C_g, then add $\exists y(C_g y \wedge R_k xy)$ and $\exists y(C_g y \wedge \neg R_k xy)$ as conjuncts of $\Phi_i x$; otherwise, add the first formula as a disjunct of $\Psi_i x$.

If the model-description contains a sentence "$C_g \rightarrow_\exists R_k C_j$," where C_j is any superclass of C_i, and it does not contain any sentence "$C_h \rightarrow_\forall R_k C_j$," where C_h is a superclass of C_g, then add $\exists y(C_g y \wedge R_k yx)$ and $\exists y(C_g y \wedge \neg R_k yx)$ as conjuncts of $\Phi_i x$; otherwise, add the first formula as a disjunct of $\Psi_i x$.

This completes the definition of $\Phi_i x$ and $\Psi_i x$, and thereby also completes the definition of T_M.

Always-preciseness

To prove the following result, we extend T_M to a new theory, T'_M, as follows. Take an arbitrary basic class—say, C_1. We get T'_M by adding a constant c to the language of T_M, and the axiom $\vdash_{T'_M} C_1 c$, which we call the axiom for c. We will deal with T'_M in what follows rather than with T_M. This is a legitimate move because,

for any sentence ϕ in the language of T_M (i.e., any sentence in T'_M not involving the new constant c), $\vdash_{T_M} \phi$ iff $\vdash_{T'_M} \phi$. In other words, T'_M is a *conservative extension* of T_M.

LEMMA 1. *T'_M is a conservative extension of T_M.*

Proof. By the non-triviality axiom for C_1, we have $\vdash_{T_M} \exists x C_1 x$. The lemma therefore follows by a theorem on functional extensions (Shoenfield 1967, pp. 55–56). □

THEOREM 1 **(Always-Preciseness)**. *Any consistent conceptual modeling script in ERM-R is precise: for any script M, the theory T_M is complete in FO^2.*

Proof. By Shoenfield (1967, p. 83, Lemmas 2 and 3), it suffices to show that every variable-free formula of T'_M is decidable in T'_M and that every simply existential formula (formula of the form $\exists x A$, with A open) is equivalent in T'_M to an open formula. FO^2 can be axiomatized to recover classical consequence (Henkin 1967)—i.e., if Γ is a set of formulas of FO^2 and ϕ is a formula of FO^2, and ϕ is derivable from Γ in FOL, then ϕ is derivable from Γ in FO^2. Therefore, in showing that each simply existential formula of FO^2 is equivalent in T'_M to an open formula of FO^2, we will not worry about intermediate formulas in FOL but not in FO^2 (e.g., through changes of bound variables).

The only variable-free formulas in $\mathcal{L}'$, the language of T'_M, are boolean combinations of formulas of the form $P_k c$, $C_k c$, and $R_k cc$. Therefore, it is enough to show that each such atomic formula is decidable in T'_M. First, each $C_k c$ is true if C_k is a superclass of C_1 by the axiom for c and the composition and class equivalence axioms. If C_k is not a superclass of C_1, then $C_k c$ is false by the composition and class exclusion axioms. Once we have the truth values of $C_k c$, we can determine whether $P_k c$ is true by the characterization and composition axioms.

Finally, take $R_k cc$. We have either $\vdash_{T'_M} \phi$ or $\vdash_{T'_M} \neg\phi$ according to whether the model-description contains a sentence "$C_j R_k$," where C_j is any superclass of C_1, by the axiom for c, the composition axioms, and the characterization axioms.

Now we show that every simply existential formula $\exists x A$ is equivalent in T'_M to an open formula. First, we assume that A is in disjunctive normal form (i.e., A is a disjunction of conjunctions of literals). Because the existential quantifier distributes over disjunction, we can assume without loss of generality that A is a conjunction of literals. We can further assume that every conjunct of A contains an occurrence of x, because $\exists x(B \wedge C)$ is equivalent to $\exists x B \wedge C$ if x does not occur in C. Now, we will show that $\exists x A$ is equivalent in

T'_M to a boolean combination of formulas in the set $\chi = \{C_i y | 1 \leq i \leq l\}$. (Once we know what classes y can belong to, we know everything there is to know about what other properties it can have.)

A is equivalent to a conjunction of the formulas A_x and A_y, where, A_x is the conjunction of those literals in A in which no variable other than x occurs, and A_y is the conjunction of those literals in A in which x and y both occur. We will build a boolean combination of formulas in χ by using A_x to restrict our attention to a subset of the original conceptual model, and then using that subset plus A_y to state a constraint on y.

To begin, focus on A_x. We will use this to determine the classes to which x might belong. We do this by showing that A_x (i.e., any conjunction of literals in each of which x occurs, but no other variable occurs) is equivalent to a disjunction $C_{i_1}x \vee \ldots \vee C_{i_j}x$. We proceed by induction on the number of literals in A_x.

First, suppose A_x is a literal. Then there are five possibilities: A_x is $C_k x$, $P_k x$, $R_k xx$, $R_k xc$, $R_k cx$, or the negation of one of these formulas. The first case is trivial. In the remaining cases, the exhaustivity, characterization, composition, class exclusion, and class equivalence axioms, plus the axiom for c, give us our result.

Now, for induction, assume that the claim holds for any formula of length k or less ($k \geq 1$), and that A_x contains $k+1$ literals. Then A_x is equivalent to $B \wedge D$, where B is a literal, and D is of length k. Then D is equivalent to a disjunction $C_{i_1}x \vee \ldots \vee C_{i_j}x$ by the induction hypothesis, and B to a disjunction $C_{g_1} \vee \ldots \vee C_{g_h}$. So A_x is equivalent to $(C_{g_1} \vee \ldots \vee C_{g_h}) \wedge (C_{i_1}x \vee \ldots \vee C_{i_j}x)$, which is equivalent to $(C_{g_1} \wedge C_{i_1}x) \vee \ldots \vee (C_{g_h} \wedge C_{i_1}) \vee \ldots \vee (C_{g_h} \wedge C_{i_j}x)$. But any formula of the form $C_g x \wedge C_i x$ will be equivalent either to $\perp$ by the composition and class exclusion axioms, or to $C_g x$ by the composition and class equivalence axioms.

So we can conclude that $\exists x A$ is equivalent in T'_M to $\exists x((C_{g_1}x \vee \ldots \vee C_{g_h}x) \wedge A_y)$. This in turn is equivalent to $\exists x(C_{g_1}x \wedge A_y) \vee \ldots \vee \exists x(C_{g_h}x \wedge A_y)$, so it is enough to show that $\exists x A$ is equivalent to a boolean combination of formulas in χ when A_x is an atomic formula of the form $C_i x$.

Now consider A_y. This is a conjunction of literals, each of which contains an occurrence of x and an occurrence of y. Therefore, each literal must be either $R_k xy$, $R_k yx$, or their negations. Note, first, that by the relation exclusion axioms, $R_k xy$ entails $\neg R_g xy$ and $\neg R_g yx$ for $g \neq k$; furthermore, either $R_k xy$ entails $R_k yx$ or it entails $\neg R_k yx$. Therefore, we may assume that A_y is either a single unnegated atomic formula $R_k xy$ or $R_k yx$, or it is a conjunction of negated atomic formulas. In the latter case, we can regard A_y as a

conjunction $B_y \wedge D_y$, where B_y is a conjunction of formulas of the form $\neg R_k xy$, and D_y is a conjunction of formulas of the form $\neg R_k yx$.

By the characterization axiom for C_i (or by the composition axiom, and the characterization axioms for the component subclasses of C_i), we can show that $\exists x A$ is equivalent to $(C_{i_1} y \vee \ldots \vee C_{i_k} y)$, where the various C_{i_g} are specified as follows. We can use A_y to produce a list of classes to which y might belong. If A_y is of the form $R_k xy$, then the list will consist of those classes C_j such that the model-description contains "$C_a \rightarrow_\exists R_k C_b$" or "$C_a \rightarrow_\forall R_k C_b$," where C_a is a superclass of C_i (the class to which x belongs) and C_b is a superclass of C_j. In this case, $\Phi_i x$ and $\Psi_i x$ (or $\Phi_j x$ and $\Psi_j x$ for the basic subclasses of C_i) were constructed to entail that $\exists y(C_j y \wedge R_g xy)$; and $\Phi_j x$ and $\Psi_j x$ for all other classes C_j were constructed to entail the negation of this formula. The list is similar if A_y is of the form $R_k yx$. Suppose instead that A_y is of the form $B_y \wedge D_y$, as described at the end of the previous paragraph. Then we produce a list for B_y and a list for D_y, and take the intersection of the two lists. The list for B_y will consist of those classes C_j such that the model-description contains no sentence "$C_a \rightarrow_\forall R_{k_g} C_b$," where C_a is a superclass of C_i and C_b is a superclass of C_j, for $1 \leq g \leq h$. In this case, $\Phi_i x$ and $\Psi_i x$ (or $\Phi_j x$ and $\Psi_j x$ for the basic subclasses of C_i) were constructed to entail that $\exists y(C_j y \wedge \neg R_g xy)$; and $\Phi_j x$ and $\Psi_j x$ for all other classes C_j were constructed to entail the negation of this formula. The list for D_y is built similarly to the list for B_y. If either of these lists is empty, or if they do not overlap, then $\exists x A$ is equivalent in T'_M to $\perp$. $\square$

The following fact is not difficult to verify. Say a script M is inconsistent iff T_M is inconsistent. For convenience, write $Cl(A)$ for the $\subseteq$-closure of a class A (in a given script M): $Cl(A)$ is the smallest set of classes including A and every class in M which can be reached by a finite number of $\subseteq$-steps from A. For example, if M includes the sentences "$A \subseteq B$" and "$B \subseteq C$," then B and C are both members of $Cl(A)$.

PROPOSITION 1. *A script M is inconsistent iff at least one of the following conditions holds:*

1. *For some classes A and B, there are classes $C \in Cl(A)$ and $D \in Cl(B)$ such that M contains both "$A \rightarrow_\exists R_k B$" and "$C \rightarrow_\forall R_k D$" for some k.*

2. *For some classes A and B, there are classes $C_1, C_2 \in Cl(A)$ and $D_1, D_2 \in Cl(B)$ such that M contains both "$C_1 \rightarrow_\forall R_i D_1$" and "$C_2 \rightarrow_\forall R_j D_2$" for some $i \neq j$.*

Condition 1 entails that M is inconsistent because our semantics for "$A \rightarrow_\exists R_k B$" entails that each member of A bears R_k to some *but not all* members of B. Condition 2 entails that M is inconsistent because of our requirement that the relations R_k be mutually exclusive.

References

Evermann J, Wand Y (2005) Ontology based object-oriented domain modelling: Fundamental constructs. *Requirements Engineering* 10:146–60.

Henkin L (1967) *Logical Systems Containing a Finite Number of Symbols*, volume 21 of *Séminaire de Mathématiques Supérieures* (Montreal: Presses de l'Université de Montréal).

Shoenfield JR (1967) *Mathematical Logic* (Reading, MA: Addison-Wesley), reprinted 2000 by Association for Symbolic Logic/A.K. Peters, Natick, MA.

Wand Y, Weber R (1993) On the ontological expressiveness of information systems analysis and design grammars. *Journal of Information Systems* 3:217–237.