Linköping Electronic Articles in
Computer and Information Science
Vol. n(1999): nr nn

Object-Oriented First-Order Logic

Eyal Amir

Department of Computer Science,
Stanford University,
Stanford, CA 94305-9020, USA
eyal.amir@cs.stanford.edu,

Linköping University Electronic Press
Linköping, Sweden

http://www.ep.liu.se/ea/cis/1999/nnn/

Published on December 20, 1999 by
Linköping University Electronic Press
581 83 Linköping, Sweden

Linköping Electronic Articles in
Computer and Information Science
ISSN 1401-9841
Series editor: Erik Sandewall

Typeset by the author using LaTeX
Formatted using étendu *style*

Recommended citation:
<Author>. <Title>. Linköping Electronic Articles in Computer and Information Science, Vol. n(1999): nr nn. http://www.ep.liu.se/ea/cis/1999/nnn/. *December 20, 1999.*

This URL will also contain a link to the author's home page.

Abstract

We wish to improve modeling in first-order logic (FOL) by using object-oriented tools. To this end, we describe Object-Oriented First-Order Logic (OOFOL), which is a simple extension of FOL. It does not surpass the expressive power of FOL, but it provides object-oriented tools to the designer.

In OOFOL, objects are theories that are connected via interface vocabularies to other objects, classes are used to provide a reusable logical template, and inheritance is used to adapt classes to specialized tasks. We show that this logic has a simple semantics provided by FOL. A variant of the logic that allows links between the objects to be unidirectional is also examined. This variant is called Directed Object-Oriented First-Order Logic (Directed OOFOL). We show that such a system can be given semantics using Circumscription.

Our new tools facilitate the object-oriented design of theories. We demonstrate this through a few applications taken from model-based reasoning, theories of action and cognitive robotics. These examples also demonstrate the use of the object-oriented methodology and tools for the construction of first-order theories. We conclude with proposed specialized inference algorithms for these logics.

1 Introduction

This paper is about modeling in First-Order Logic (FOL) and the tools needed for it. FOL provides a richness of representation, but does not necessarily make change and reuse easy. The difficulty of building and reusing first-order theories is visible in the obstacles encountered in such areas as Commonsense Reasoning, Knowledge Reuse and Formal Verification. On the other hand, the Object-Oriented paradigm has proved to be successful at facilitating software engineering, database engineering and knowledge engineering. (e.g., [27], [35], [4], [29], [25]).

In this paper, we propose adding object-oriented tools to FOL. We show that our extension does not increase the expressive power of FOL by giving it a simple semantics using FOL. A second extension that also allows for directionality of inference is shown to be semantically characterizable using Circumscription. In these logics, classes are theories equipped with interfaces, objects are copies of their classes with links to other objects, and inheritance is used to define classes that specialize their parents. We show the existence of efficient inference algorithms for both logics.

We show that the extensions we provide are suitable and sufficient for several application areas. We do so by building example theories for physical systems (following the model-based reasoning approach), hierarchical domain models, theories of action and modeling agents.

The combination of object-oriented ideas and logical knowledge representation is not new. Frame systems (e.g., KRYPTON/KL-ONE [4]), Context Systems (e.g., CycL [29]), Object-Oriented Prolog dialects (e.g., Prolog++ [38]) and (more remotely) logics describing object-oriented programs and databases (e.g., [24], [27], [31]) are examples. Still, frame systems do not have the theoretical framework or tools to support object-oriented design in FOL. Most remarkably, frame systems either have no semantic difference between a FOL axiom associated with one frame's ABox or another frame's ABox (e.g., Ontolingua [19], [13]), or there is such a difference, but the semantics is complicated (context-supported CycL). Our work is directly applicable to these systems.

We expand on this comparison at end of this paper. Before that, Sections 2, 3, 4 and 5 describe the motivation and structures of our approach, the syntax and semantics, several applications, and entailment methods, respectively. Throughout this paper, we treat logic as an engineering tool and view it through the eyes of an engineer. This approach influences our design decisions, as will be evident.

2 An Object-Oriented Approach to Logic

The intuition behind our approach to Object-Oriented logic is that a theory can be presented as a collection of theories communicating with one another via *interface vocabularies*. From this point of view, *a theory is a graph of smaller theories* (the graph is possibly directed). If the interface vocabularies are small, there is a significant performance improvement over ordinary FOL theories because of the algorithms that can be applied. Also, the directionality of such interfaces allows us to restrict inconsistencies from influencing other parts of the theory. Modifications in the internals of these theories have only local impact, an impact that is circumscribed by the interface vocabularies.

We describe the structure of objects through a simple example. The *Cup* object, depicted in Figure 1, describes some of the characteristics of

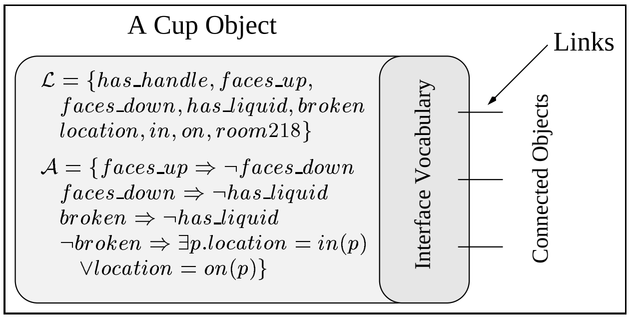

Figure 1: A Cup object.

a typical cup. As with other objects, it has a vocabulary, a set of axioms, an interface vocabulary and a set of links to other objects. In its simple form (non-directional), a link specifies an equality between two interface-vocabulary symbols in two respective objects.

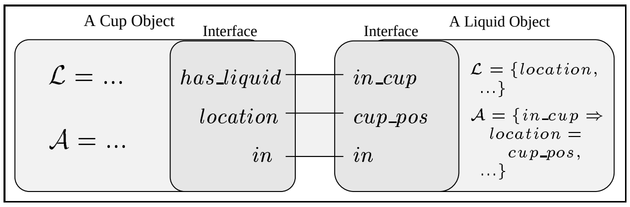

Figure 2: Connecting Cup and Liquid objects.

In Figure 2 two linked objects are presented. Some of the interface symbols of the *Cup* object are linked to the interface symbols of the *Liquid* object. The link specifies that *has_liquid* in the object *Cup* has the same semantics as *in_cup* in the object *Liquid*. The same goes with the pairs $location, cup\_pos$ and in, in. From a first-order logic perspective it is as if we added the equality/equivalence axioms $Cup.has\_liquid \equiv Liquid.in\_cup$, $Cup.location = Liquid.cup\_pos$ and $\forall x(Cup.in(x) = Liquid.in(x))$.[1]

From an automated reasoning perspective, describing a theory as a set of objects allows the following style of reasoning: All objects generate consequences in parallel, sending results to those other objects interested in them. These results are then incorporated into the theorem-proving process of the receiving objects. Axioms sent from object to object are restricted to the communication vocabulary, but the language is full FOL. For example, look again at Figure 2. If an object *Liquid* has a link to the *Cup1* object that includes $has\_liquid, location, in$ (on the side of *Cup1*), then a sentence like $has\_liquid \Rightarrow \exists p\ location = in(p)$ will be translated and sent to *Liquid* in the form $in\_cup \Rightarrow \exists x\ cup\_pos = in(x)$.

The use of fixed interfaces allows the encapsulation of the internals of objects. This way, we can add or replace internal axioms with higher-grain axioms, describing the type of liquid, the heat, the degrees of tilt of the cup or the material of which the cup is made. Since we can expect the interface to remain the same, we can verify the change by examining

[1] All of this machinery can be seen as a special case of similar techniques used in context theories such as [16]. More on that in Section 6.

changes to results in the interface, paying less attention to other internal results (possibly applying the techniques of [23]).

To allow reuse of theories and following object-oriented design tradition, we define a *class* to be a "template" of objects. A class can be thought of as an object without specified links. Alternatively, an object is simply a copy of its class with links to other objects. We allow different uses of two objects of the same class in the same theory (linked to possibly different objects).

The Covered_Cup Class

$$\mathcal{L}_{Covered\_Cup} = \underline{\mathcal{L}_{Cup}} \cup \{cover\_on, has\_liquid'\}$$

$$\mathcal{A}_{Covered\_Cup} = \underline{(\mathcal{A}_{Cup})_{[has\_liquid/has\_liquid']}} \cup \{has\_liquid' \Rightarrow has\_liquid, \; faces\_down \wedge \neg cover\_on \Rightarrow \neg has\_liquid \; broken \Rightarrow \neg has\_liquid\}$$

$$\mathcal{L}^{inter}_{Cov\_Cup} = \underline{\mathcal{L}^{inter}_{Cup}}$$

Figure 3: A subclass of the **Cup** class.

Inheritance is yet another object-oriented tool used in many KR systems. It is sometimes reasoned about using nonmonotonic reasoning systems (e.g., [32], [36], [41]). In contrast, we implement inheritance of a class from its parent class outside the logic, as we are interested in it as an engineering tool.

Figure 3 describes **Covered_Cup**, a subclass of **Cup**; the inherited part is underlined. A subclass inherits *all the axioms* and vocabulary of its superclass, but may add symbols to the vocabulary and the interface vocabulary and may also add axioms. To enable the subsumption of axioms, we allow symbols used in the superclass to be replaced by new symbols (i.e., the subsumption of axioms is done by subsuming symbols[2]). In our example, the symbol has_liquid is substituted by has_liquid' in all of $\mathcal{A}_{Cup}$ to allow for a new definition for has_liquid.

This kind of inheritance is similar to inheritance in Object-Oriented programming languages, where the interface of the subclass must include the interface of the superclass, but may implement some of the internals differently. This definition allows the specialization of a class by adding more detail in its subclasses.

3 Object-Oriented FOL

In what follows, we use some notations that are common in object-oriented programming languages, namely, $Obj.c$ (or $N.c$) to refer to the symbol c in the object Obj (with the name N). For simplicity of exposition, we shall not define classes and objects recursively (i.e., classes whose theories are themselves OOFOL theories). A recursive definition may sometimes be useful, but it requires only a simple technical step and makes our notations significantly more dense. Thus, this definition is omitted, but we note that all the results that follow hold for this more general case.

[2]This enables the reuse of superclass results in the subclass.

First, we define our language: classes and objects, interfaces, links and theories. Unless otherwise mentioned, all languages mentioned are full first-order.

A *Class* C is a tuple $\langle \mathcal{L}_C, \mathcal{A}_C, \mathcal{I}_C \rangle$ corresponding to an FOL vocabulary ($\mathcal{L}_C$), a set of FOL axioms ($\mathcal{A}_C$) in this language, and $\mathcal{I}_C \subset \mathcal{L}_C$, the interface vocabulary of C.

An *OOFOL Theory* T is a set of statements

$$Object(C, N, L),$$

(defining an object) where C is a class reference (a class description as above, or a reference to such a description), N is a unique name for the object (typically a string of symbols) and L is a linking relation $L = \{(P_i, N_i, Q_i)\}_{i \leq n}$ that specifies the links between this object and other objects in the theory. Here P_i is a symbol in the interface vocabulary of C, N_i is a name of another object in T, and Q_i is an interface symbol of this other object. This way, an object is simply "declared" to be in the theory. We assume that no statement in T refers to a name N_i (in the linking relation) for which no corresponding object exists in T.

3.1 FOL Semantics

Unlike object-oriented approaches to logic that use context (see Section 6), we take advantage of the limited expressivity of our language (we cannot quantify over objects) and the limited interaction among our objects to give a simple FOL semantics to OOFOL theories.

We define the semantics by making objects have mutually exclusive vocabularies and viewing the links as equality/equivalence assertions. For an OOFOL theory T, define $\mathcal{L}(T) = \{N.P \mid Object(C, N, L) \in T,\ P \in \mathcal{L}_C\}$. This language gives a unique vocabulary for each object in T. We apply a simple translation into this vocabulary (replacing P by $N.P$ in the axioms of the object N for each symbol P in its vocabulary) and add the equality/equivalence assertions:

$$\begin{aligned}\widetilde{T} = \{&(\mathcal{A}_C)_{[P/N.P \mid P \in \mathcal{L}_C]} \mid Object(C, N, L) \in T\} \cup \\ &\{\forall \overrightarrow{x} (N.P(\overrightarrow{x}) \equiv N'.P'(\overrightarrow{x})) \mid Object(C, N, L) \in T,\ L(P, N', P')\}\end{aligned} \tag{1}$$

where "$\equiv$" means "$=$" if P, P' are function symbols, and $\overrightarrow{x}$ may be of arity 0 (if P, P' are propositional or constant symbols).

An *OOFOL structure* in the language $\mathcal{L}(T)$ is structure of $\mathcal{L}(T)$ as an FOL language.

Definition 3.1 (FOL Semantics for OOFOL) *Let T be an OOFOL theory and let $\mathcal{M}$ be an OOFOL structure of $\mathcal{L}(T)$. A formula $\varphi \in \mathcal{L}$ is satisfied in $\mathcal{M}$, written $\mathcal{M} \models \varphi$, iff $\mathcal{M}$ satisfies φ in the FOL sense.*

$\mathcal{M}$ is a model of T, *written $\mathcal{M} \models T$ iff $\mathcal{M} \models \widetilde{T}$ (in the FOL sense).*

In other words, the semantics of an OOFOL theory as an undirected graph of connected theories has the links state equality/equivalence among symbols and otherwise has the theory behave like an FOL theory.

EXAMPLE   If T is the OOFOL theory containing the two connected objects from Figure 2, then

$$\begin{aligned}T = \{&Object(\mathbf{Cup}, \mathfrak{Cup1}, \{(has\_liquid, \mathfrak{Liquid1}, in\_cup), \\ &\quad (location, \mathfrak{Liquid1}, cup\_pos), (in, \mathfrak{Liquid1}, in)\}), \\ &Object(\mathbf{Liquid}, \mathfrak{Liquid1}, \{\})\}\end{aligned}$$

The equality/equivalence sentences that we add in defining $\widetilde{T}$ and T's semantics are:

$$\begin{array}{l} Cup1.has\_liquid \equiv Liquid1.in\_cup \\ Cup1.location = Liquid1.cup\_pos \\ \forall x(Cup1.in(x) = Liquid1.in(x)). \end{array}$$

Notice that the definition of Liquid1 above did not include any links, because the links were already specified for $Cup1$. ■

3.2 Directional Semantics

The previous section gave semantics that is sufficient to represent an object-oriented system that has no direction associated with the links in the objects graph. From a procedural perspective, this approach corresponds to allowing influence to flow in both directions of a link. In Figure 2, if we proved $Cup1.has\_liquid$, then we can infer $Liquid1.in\_cup$ and vice versa.

Sometimes, we would like to have the influence go in only one direction. For example, we may want to allow different accuracies of reasoning in two connected objects O_1, O_2. *Accuracy* here means that although something of interest to O_1 may come up in O_2, we prefer to have O_1 ignore it. Abstractions and approximations are examples of such differences in accuracy. This ability to ignore influence is of particular importance in time-dependent tasks.

To give directional semantics to OOFOL theories, we convert the equivalence/equality axioms that we added in Section 3.1 into defaults. This is done simply by adding *abnormality predicates* in front of each link equality/equivalence. We later *Circumscribe* these abnormalities. The circumscription will be done for each pair of linked objects separately. Thus, we write the corresponding formula to (1) for each pair separately:

$$\begin{array}{l} \widetilde{\mathcal{A}}(O_1,O_2) = \{(\mathcal{A}_{C_i})_{[P/N_i.P|P\in\mathcal{L}_{C_i}]} \mid i=1,2\}\cup \\ \quad \{\forall \overrightarrow{x}(Ab_{N_1.P_1,N_2.P_2}(\overrightarrow{x}) \Rightarrow N_1.P_1(\overrightarrow{x}) \equiv N_2.P_2(\overrightarrow{x})) \mid L_2(P_2,N_1,P_1)\} \end{array} \quad (2)$$

where $O_i = Object(C_i, N_i, L_i)$.

We use Circumscription [32] for our semantics. Circumscription is a method for nonmonotonic reasoning in which, given a set of axioms T, the extensions of some predicate symbols $\overrightarrow{P}$ are minimized while some other symbols $\overrightarrow{Q}$ are allowed to vary their semantics. This is achieved by replacing T with the second-order formula (or FOL schema) $Circ[T;P;Q]$:

$$T \wedge \forall p,q[T(p,q) \wedge p \leq P \Rightarrow p = P].$$

In the following semantics, to have influence flow in only one direction, we circumscribe the abnormality predicates Ab_{O_1,O_2} (the set of all abnormality predicates in $\widetilde{\mathcal{A}}(O_1,O_2)$) in the axioms of the object-pair, $\widetilde{\mathcal{A}}(O_1,O_2)$. We allow only the symbols of O_2 to vary, thus allowing influence to flow from O_1 to O_2, but not vice versa.

Definition 3.2 (Directional Semantics) *We say that* $\mathcal{M}$ is a Directed OOFOL model of an OOFOL theory T, *written* $M \models_D T$, *iff*

$$M \models \bigwedge_{O_1,O_2\in T} Circ[\widetilde{\mathcal{A}}(O_1,O_2); Ab_{O_1,O_2}; \mathcal{L}(O_2)]$$

We define $M \models_D \varphi$ and $T \models_D \varphi$, for a formula $\varphi \in \mathcal{L}(T)$, as usual.

Roughly speaking, in this definition, the links of O_2 serve as *inputs* and the links of O_1 serve as *outputs*. This distinction is made precise by allowing the circumscription to vary only $\mathcal{L}(O_2)$, holding $\mathcal{L}(O_1)$.

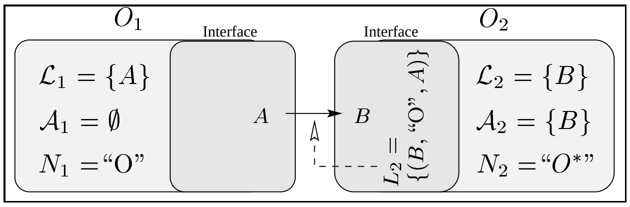

Figure 4: Influence flows from O_1 to O_2.

EXAMPLE Look at Figure 4. We are going to show that influence between the objects O_1 and O_2 flows only in the arrow's direction. As before, the link $L_2(B, "O", A)$ roughly says "link my B to O's A". Proving $O_2.B$ does not allow us to prove $O_1.A$. Formally,

$$Circ[B \wedge (\neg ab \Rightarrow (B \iff A)); ab; B] \not\models A.$$

This is because the only models (in the propositional language $\{A, B\}$) that satisfy $B \wedge (\neg ab \Rightarrow (B \iff A))$ are $M_1 = \{\neg A, B, ab\}, M_2 = \{A, B, \neg ab\}$, $M_3 = \{A, B, ab\}$. M_1, M_2 are not comparable by the preference relation of our circumscription because they differ on A (which is held constant in our circumscription). M_2 is preferred to M_3 and thus we have two minimal models of our circumscription, one of which does not satisfy A.

3.3 Inheritance

Unlike nonmonotonic inheritance (as in [37], [24]), inheritance in our system is treated outside the logic, having the engineer specify its exact use. We wish to allow three differences between a class and its parent: (1) the child's vocabulary and interface vocabulary are expansions of the parent's; (2) the child class inherits all the axioms of the parent, subject to the replacement of non-logical symbols (in all of the axioms in the same way) with some of the new symbols in the child's language; and (3) the child class may have additional axioms.

Formally, for a set of axioms A and non-logical symbols c, c', $A_{[c/c']}$ is the set of axioms A with every occurrence of the symbol c replaced by an occurrence of the symbol c'. $A_{[c_1/c'_1,\ldots,c_n/c'_n]}$ is similarly defined. A class $C = \langle \mathcal{L}, \mathcal{A}, \mathcal{I} \rangle$ is a *subclass* of $C' = \langle \mathcal{L}', \mathcal{A}', \mathcal{I}' \rangle$ if $\mathcal{L}' \subseteq \mathcal{L}$, $\mathcal{A}'_{[c_1/c'_1,\ldots,c_n/c'_n]} \subseteq \mathcal{A}$ $(n \geq 0)$ and $\mathcal{I}' \subseteq \mathcal{I}$. A detailed example is provided in Section 4.1 on the next page.

4 Applications

The last section provided the basic tools suggested by the object-oriented paradigm: classes, objects, interface and inheritance. In this section we show that the tools we provided are suitable for object-oriented design in FOL. The examples that we provide demonstrate the way object-oriented design can be used in building first-order theories.

4.1 Model-Based Reasoning

Below, we give a fragment of a theory describing the functionality of a car. We use qualitative proportionalities (see [14]) to represent the relationships among different variables of the car (in our case, these are monotonically increasing and monotonically decreasing functions). Figure 5 describes some of the classes for this domain and the theory fragment.

class Qualitative_Proportionalities {
axioms: $\forall xy(x > y \Rightarrow rel\_pos(x) > rel\_pos(y))$
$\forall xy(x > y \Rightarrow rel\_neg(x) < rel\_neg(y))$ **}**

class Brakes : Qualitative_Proportionalities {
interface: $apply\_force(x)$
axioms: $\forall xy(pedal\_pressure(x) \wedge oil\_pressure(y) \Rightarrow$
$apply\_force(rel\_pos(x) * rel\_pos(y)))$
$pedal\_pressure(0) \Rightarrow apply\_force(0),\ oil\_pressure(70)$ **}**

class ABS : Brakes {
interface: $apply\_force(x), abs(x)$
axioms: $\forall x(old\_apply\_force(x) \wedge x \geq ABS\_step \Rightarrow apply\_force(abs(x)))$
inherit: $[apply\_force/old\_apply\_force]$ **}**

class Tires : Qualitative_Proportionalities {
interface: $bad\_traction(wheel), apply\_force(x),\ engine\_force(x), brake\_force(x)$
axioms: $\forall xy(brake\_force(x) \wedge engine\_force(y) \Rightarrow$
$apply\_force(rel\_pos(y) - rel\_pos(x)))$
$\forall wheel(\neg balanced(wheel) \vee old(wheel) \Rightarrow bad\_traction(wheel))$
$\forall wheel(balanced(wheel) \wedge new(wheel))$
$\forall x(new(x) \iff \neg old(x))$ **}**

class Car : Qualitative_Proportionalities {
interface: $tire\_force(x), bad\_traction(wheel)$
axioms: $\exists w.bad\_traction(w) \wedge tire\_traction(t) \Rightarrow t < Good,$
$\forall t(\forall w.\neg bad\_traction(w)) \wedge tire\_traction(t) \Rightarrow t = Good$
$\forall fts(tire\_force(f) \wedge tire\_traction(t) \wedge speed(s) \Rightarrow$
$stop\_time(rel\_neg(s) * rel\_pos(f) * rel\_neg(t)))$ **}**

$$T \begin{cases} Object(\mathbf{ABS}, \mathfrak{Brakes1}, \{\}). \\ Object(\mathbf{Tires}, \mathfrak{Tires1}, \{(brakes\_force, \mathfrak{Brakes1}, apply\_force)\}). \\ Object(\mathbf{Car}, \mathfrak{Car1}, \{(tire\_force, \mathfrak{Tires1}, apply\_force), \\ \quad (bad\_traction, \mathfrak{Tires1}, bad\_traction)\}). \end{cases}$$

Figure 5: Some of the classes in the CAR theory and a theory fragment using these classes.

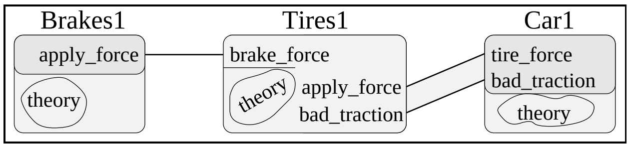

Figure 6: Diagrammatic structure of the CAR theory fragment.

The notation *class C1 : C2* indicates that $C1$ is a subclass of $C2$. For example, we have some knowledge about ordinary brakes systems, but we also have some specialized knowledge about anti-lock lock brakes systems (ABS). The class **ABS** is a subclass of **Brakes** in which we inherit all the axioms of **Brakes**, replacing the symbol $apply\_force$ with old_apply_force (in those inherited axioms). In this subclass, we also add one more axiom and add an interface symbol abs. The object named $\mathfrak{Brakes1}$ is declared of class

ABS, but is used as an instance of class **Brakes**. The theory including the three objects $\mathfrak{Brakes1}$, $\mathfrak{Tires1}$ and $\mathfrak{Car1}$ is semantically/diagrammatically depicted in Figure 6.

Using this theory, we can prove that increasing the pressure on the brakes will make the car stop faster, if we keep the rest constant (e.g., not increase the pressure on the accelerator). Notice that we chose undirected semantics (FOL semantics) because influence may flow in both directions between the objects: A tire explosion may influence the force applied on the brakes.

4.2 Hierarchical Reasoning

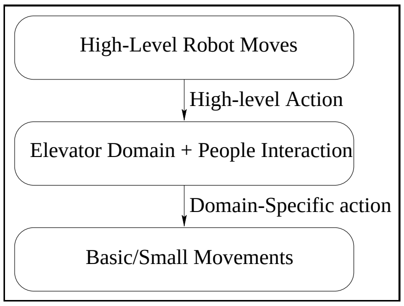

Figure 7: A hierarchical decomposition.

Suppose that a robot is given the assignment of navigating from one office to another. Since the robot's moves are miniscule and are supposed to be run and monitored in real-time, reasoning about the entire plan can be extremely expensive. Adding the requirement that the robot should reason about the behavior of other objects/agents (e.g., the elevator, doors, etc) makes brute-force reasoning intractable and knowledge modeling difficult.

With Directed OOFOL one can represent and reason about several layers of abstraction concurrently (similar to Brooks' Subsumption architecture [5]), leading to the desired behavior at the low-level sensors/actuators. To reason about high-level actions, the robot does not need to reason about the different ways these actions may be realized by lower layers. Figure 7 describes the theory. A more detailed description of a use of a similar theory is given in [1].

4.3 Action Theories: PMON and Multiple Agents

The temporal logic PMON [39], [22] uses the *features and fluents* language $\mathcal{L}(\mathcal{FL})$ and has a theory divided into several subtheories: Γ_{OBS} (observations), Γ_{SCD} (schedule of events), Γ_{UNA} (unique names axioms), Γ_T (axiomatization of time) and Γ_{NCG} (no-change axiom).

Different axioms are allowed in each category. We refer the reader to [22] for a relatively recent work on the subject. [22] showed that the semantics of PMON can be defined by $\Gamma_{NCG} \wedge \Gamma_{OBS} \wedge \Gamma_{SCD} \wedge \Gamma_{UNA} \wedge \Gamma_T \wedge$ $Circ[\Gamma_{SCD}(Occlude); Occlude;]$.

Thus, we can represent PMON using the Directed OOFOL theory displayed schematically in Figure 8. In this figure, OCC is an object including the axiom $\forall x.Occlude(x)$. The rest of the objects represent their respective theories.

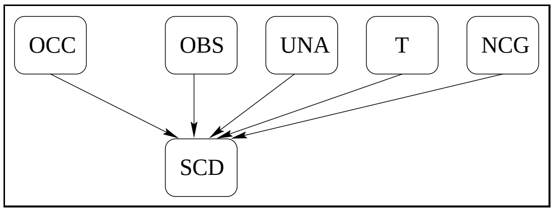

Figure 8: PMON in Directed OOFOL.

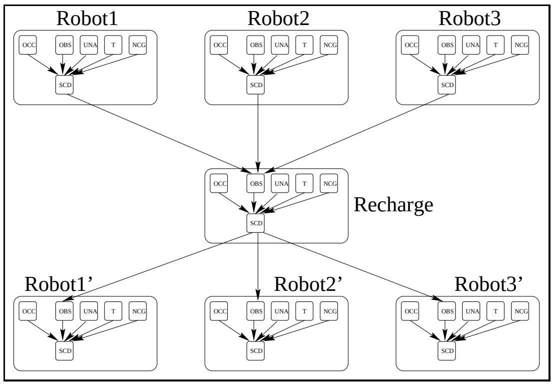

Figure 9: Multiple Agents.

When we want to reason about several agents, decomposing the domain according to the different agents and their interactions can yield a significant gain in modeling effort and reasoning efficiency.

For example, if three robots should each perform a different task, but interact every hour to consider battery-recharging, we can create a class *Robot* that includes the action theory for that robot and a class *Recharge* that includes the action theory for the multiple robots involved. Each object *Rob1, Rob2, Rob3, Rob1', Rob2', Rob3'* is of class *Robot*. An action schedule can be given to each of the robots, and the results of executing these schedules are fed into the *Recharge* object (of class *Recharge*). The results of the recharging process are then fed into the separate schedules, and the robots may continue in their separate tasks. The complete picture is depicted in Figure 9.

5 Algorithms

Algorithms for inference in OOFOL and Directional OOFOL may reduce the theories to the appropriate semantics and perform inference on the result. We are more interested in algorithms that can take advantage of the special structure of our theories. We present such an algorithm (each semantics with its own variant) for the special case of totally-ordered theories. The general case can be reduced to this one using the methods of [2].

5.1 Totally Ordered Theories

We say that an OOFOL theory T is *totally ordered* if the objects in T can be sorted into a chain $\langle Obj_1, ..., Obj_n \rangle$ such that every object Obj_i in the chain is directionally linked (or simply *linked* in the FOL-semantics case) to its predecessor Obj_{i-1} and there are no other links (i.e., no "loops").

For totally ordered theories, we can give equivalent semantics to our Directional Semantics using Prioritized Circumscription. Define $flat(T) \stackrel{def}{=} \cup_{O_i,O_j \in T}(\widetilde{\mathcal{A}}(O_i,O_j))$.

Theorem 5.1 (Equivalent Directional Semantics) *Let* $T = \{Obj_i\}_{i \leq n}$ *be a totally ordered OOFOL theory.* $\mathcal{M}$ is a model of T *iff it is a model of*

$$\bigwedge_{i \leq n} Circ[flat(T); Ab_i; Ab_{i+1}, ..., Ab_n, \mathcal{L}(Obj_{i+1}), ..., \mathcal{L}(Obj_n)]$$

where $Ab_i = Ab_{O_i,O_{i+1}}$.

PROOF See Appendix A.1.

Note that the formula above can be written as a single Prioritized Circumscription formula after the symbols in $\mathcal{L}(Obj_{i+1}), ..., \mathcal{L}(Obj_n)$ are eliminated from the varied part [7].

5.2 Message-Passing

Applying our original intuition from Section 2, the following algorithm performs inference in one object, passes results to the appropriate objects and repeats the process in subsequent objects. Various inference methods can be used to prove a given query in a given object, but for simplicity we assume forward reasoning and a *totally ordered* theory T. We describe the different variances needed for each of the semantics after describing the algorithm.

For two objects $Obj = Object(C, N, L), Obj' = Object(C', N', L')$, let $\mathcal{L}_{Obj,Obj'}$ be the language that includes the set of "linking symbols" of Obj, Obj', $\{P \mid L'(Q, N, P) \text{ for some } Q\}$ with equality. For every sentence φ in $\mathcal{L}_{Obj,Obj'}$, we say that the *translation of* φ *from* Obj *to* Obj' is the sentence resulting from following L' in replacing every symbol $N.P$ in φ by $N'.Q$ such that $L'(Q, N, P)$. $\mathcal{L}_=$ is the first-order language with the relation "=" and no other relation, function or constant symbols. Figure 10 describes our message-passing algorithm. Theorem 5.2 validates our approach for the FOL semantics.

PROCEDURE MESSAGE-PASSING
Given a totally ordered OOFOL theory $T = \{Obj_i\}_{i \leq m}$ and a query Q whose symbols are from $\mathcal{L}(Obj_n)$ (for some $n \leq m$),

1. Concurrently perform consequence finding for each of the objects $Obj_i \in T$ with $i \leq m$.
2. For each $i \leq n$, when we prove a formula $\varphi \in \mathcal{L}_{Obj_i,Obj_{i+1}}$, add the translation of φ from Obj_i to Obj_{i+1} to the set of axioms of Obj_{i+1}.
3. For each $n < i \leq m$, when we prove $Obj_i \models \varphi$, for $\varphi \in \mathcal{L}_=$, add φ to the set of axioms of Obj_n.
4. If we proved Q in Obj_n, return YES.

Figure 10: An algorithm for proving Q from a (Directed) OOFOL theory T by message-passing.

Theorem 5.2 (Completeness and Soundness for OOFOL) *Let $T = \{Obj_i\}_{i \leq n}$ be a totally ordered OOFOL theory. Let $\varphi \in \mathcal{L}(Obj_n)$ be a sentence. $T \models \varphi$ if and only if the MESSAGE-PASSING algorithm outputs YES on φ given T.*

PROOF Follows immediately from a similar theorem in [2]. ■

Theorems 5.3 and 5.5 validate our approach to the Directional Semantics.

Theorem 5.3 *Let $T = \{Obj_i\}_{i \leq m}$ be a totally ordered OOFOL theory with $Obj_i = \langle C_i, N_i, L_i \rangle$. Let $n \leq m$, $\varphi \in \mathcal{L}(Obj_n)$ be a sentence and $E = \{\psi \in \mathcal{L}_= \mid \exists i > n \; Obj_i \models \psi\}$. If $T \models_D \varphi$ and the circumscriptions involved in the semantics of T are* smooth[3]*, then there is $a \leq n$ and a sequence of sentences $\varphi_a, \ldots, \varphi_n$ such that $E_n \cup \{\varphi_n\} \models \varphi$, $Obj_a \models \varphi_a$ and every $a \leq \forall i < n$ $\varphi_i \in \mathcal{L}_{Obj_i, Obj_{i+1}}$ such that $Obj_{i+1} \models \varphi_i' \Rightarrow \varphi_{i+1}$, where φ_i' is the translation of φ_i from Obj_i to Obj_{i+1}.*

PROOF See Appendix A.2.

Corollary 5.4 (Completeness for Directional Sem.) *Let $T = \{Obj_i\}_{i \leq m}$ be a totally ordered OOFOL theory and $\varphi \in \mathcal{L}(Obj_n)$ be a sentence ($n \leq m$). If $T \models_D \varphi$, then MESSAGE-PASSING will output YES.*

For soundness, we need to assume that influence does not "try" to flow back (i.e., if $O_{i+1} \models \varphi'$ for some $\varphi \in \mathcal{L}_{O_i, O_{i+1}}$, then $T \setminus \{O_j \mid j \leq i\} \models \varphi$).

Theorem 5.5 (Soundness for Directional Sem.) *Assume that there are $\varphi_k, \ldots, \varphi_l$ such that $k \leq l$ and $O_k \models \varphi_k$ and $k \leq \forall i < l$, $\varphi_i \in L_{O_i, O_{i+1}}$ and $O_i \models \varphi_i' \Rightarrow \varphi_{i+1}$, where φ_i' is the translation of φ_i from O_i to O_{i+1}. Then, $T \models \varphi_l$.*

PROOF Since there is no attempt for influence back, we know by lemma A.4 that $T \models \forall x. \neg ab(x)$ for all the abnormality predicates ab in our system. We get that the soundness follows from that of entailment in FOL. ■

There are two main differences between the uses of MESSAGE-PASSING for the two semantics: (1) for FOL Semantics we can ask queries only from the last object in the sequence, Obj_m; and (2) MESSAGE-PASSING is not sound for Directional Semantics if influence "tries" to flow back, in the sense described above. This is because when there is such influence, the semantics sanctions that it may block influence flow forward (i.e., some of the ab's become TRUE), while there is no such block in our algorithm (we always transfer axioms forward).

It is illuminating to notice the close relationship between our assumption above and the soundness of inference. If we weaken the assumption to merely state that T is consistent with $\forall x. \neg ab(x)$, we immediately find a counter example to soundness. Take $O_1 \models P$ and $O_2 \models \neg P' \vee \neg Q'$, where P, P' are linked by ab_P and Q, Q' are linked by ab_Q. Our algorithm will pass P from O_1 to O_2, and we will conclude $\neg Q'$ in O_2. On the other hand, according to our directional semantics, $\neg Q'$ does not follow from $T = \{O_1, O_2\}$, since there is a minimal model $\mathcal{M}$ of T in which $\mathcal{M} \models P \wedge Q \wedge \neg P' \wedge Q'$.

[3]See [26] for the definition.

5.3 Branches and Cycles

To generalize over the totally ordered case, one approach is to rewrite every structure into a reduced totally ordered OOFOL theory by aggregating groups of objects into single objects (a different approach is described in [2]).

Assume that we have an arbitrary OOFOL theory $T = \langle Obj_i \rangle_{i \in \mathcal{I}}$ and we wish to prove $T \models \varphi$, where $\varphi \in \mathcal{L}(Obj_i)$ for some $i \in I$. All we need to do is to unify objects such that we are left with a new theory T' that is totally ordered. This unification will most likely result in an increased *bandwidth* for the links among the new objects, and this is the price we must pay for such an algorithm.

To see how this happens, see Figure 11. Here we see that, to use the MESSAGE-PASSING method to prove $O4.p \vee O4.q$, we cannot simply pass messages from $O1$ to $O2, O3$ to $O4$; rather, we have to unify O_2, O_3 into O_2' (on the right side of the figure).

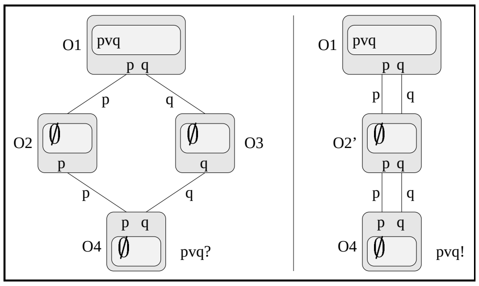

Figure 11: An example of joining nodes.

One may think that we can generalize MESSAGE-PASSING to work for a circular Directed OOFOL theory by simply continuing to transfer formulas from one object to the next until we prove our goal. Unfortunately, there are instances in which MESSAGE-PASSING is not complete for such theories.

6 Other Object-Oriented KR Systems

The seemingly closest approach to ours is the work on object-oriented Prolog (O-O Prolog) (e.g., [40] and [38]). Obviously, full FOL expressivity is different from Horn clauses with the Closed World Assumption. More interestingly, Prolog and FOL raise different challenges for object-oriented approaches. O-O Prolog structures are built using a programming perspective, whereas OOFOL structures are built emphasizing a representation perspective. Also, the program flow in O-O Prolog is identical to the program flow in ordinary Prolog. In contrast, the object-oriented structures in OOFOL present guidelines for *theorem proving flow* which do not exist in FOL. The object structure gives heuristic information to the theorem prover on how to aggregate computation and what information should flow from object to object. Such information is not used in O-O Prologs.

Another close approach is Frame Systems. Currently, there is no adequate object-oriented approach for full FOL in Frame systems. KRYPTON / KL-ONE [4], one of the early Frame systems, had an A-Box part that allowed arbitrary FOL sentences to be stated. This structure was completely flat and had nothing of the object structure that we gave to our OOFOL.

Ontolingua [19], [13] which is one of the most expressive Frame systems to date, allows different sentences to be associated with each frame. Unlike OOFOL, there is no difference between a sentence put in one frame and the same sentence put in a different frame in Ontolingua [12]. In particular, there is no "scope" in which the axiom is active/inactive. The situation is similar in other frame systems such as [8] (for a recent survey see [15]). Frame systems also traditionally use only a restricted form of the axiom inheritance used in this paper (for the A-Box part of these systems). *Overriding* is achieved (if at all) using nonmonotonic techniques.

The field of *description logics* (e.g., [3], [10]) grew out of the interest in generalizing frame systems and giving them clear semantics and tractable algorithms. Subsets of FOL that are computationally tractable are discussed in the context of describing objects. In contrast, our approach to describing objects is not restricted to relations that include this object and those related to it. We follow the object-oriented approach devised by programming languages in which objects serve as landmarks around which one builds the application/theory. This results in a different approach to object-oriented design than that applied in description logics, together with unlimited first-order logic expressivity.

Compared to the approaches for formalizing context (e.g., McCarthy's [33], CycL [29, 21, 20, 28] and QLC [6]), our logic allows only prescribed connections between the different objects (contexts usually have no restrictions), and instead of appealing to modalities for representing the different theories (or other special-purpose semantics as in [17], [16], [18] (using local-models semantics for the undirected case and proof-theoretic semantics for the directed case)), we use a simple FOL semantics for the undirected case and Circumscriptive/Preferential semantics for the directional case. Our semantics also has one universe of elements, whereas contexts are usually considered to have distinct domains of elements. For OOFOL we devised much more efficient general-purpose algorithms than those available for context systems (see [29], [2]).

In sum, no other approach to date describes the application of object-oriented design in first-order logic. In addition, to date there is no other approach to logic that supplies the tools needed for object-oriented design without abandoning the original first-order logic semantics.

Other knowledge-representation language that are related to object-oriented designs (e.g., [24], [25], [27], [31]) have contributed some intuitions to this work, but are not directly relevant here.

7 Conclusions

Object-Oriented Design is a proven method in software engineering. Its benefits range from ease of change and composition, aggregation of related information into the same place, information encapsulation and hiding. These benefits make software construction a much less tedious process, allowing software to be larger and more robust.

We presented an object-oriented approach to First-Order Logic. We showed that our approach can be captured by a simple FOL semantics in the undirected case and a simple circumscriptive semantics in the directional case. These semantics are especially simple compared to previous approaches to decomposing logical theories into small fragments. We demonstrated several applications of our logics and showed how object-oriented design can be facilitated by the tools we provided. Finally, we proposed efficient inference algorithms for both logics.

Compared to unstructured FOL, the object-oriented constructs encourage the knowledge engineer to aggregate knowledge for a particular use in one place, hiding most of the details of the implementation from the outside world by the provision of minimal interfaces. These interfaces limit the impact of internal-object changes on the rest of the theory. To allow for knowledge reuse, information integration and better knowledge engineering, FOL needs to be used with a structured methodolgy. In this work we demonstrated one such methodology and provided the tools needed for its application.

Currently, we are applying the work discussed here to an object-oriented approach to situation calculus [34].

References

[1] Eyal Amir and Pedrito Maynard-Reid. Logic-based subsumption architecture. In *Proc. Sixteenth International Joint Conference on Artificial Intelligence (IJCAI '99)*, 1999.

[2] Eyal Amir and Sheila McIlraith. Paritition-based logical reasoning. In *Principles of Knowledge Representation and Reasoning: Proc. Seventh Int'l Conference (KR '2000)*, 2000. To appear. Also available at http://www-formal.stanford.edu/eyal/papers/oo-reason.ps.

[3] Alex Borgida. On the relative expressiveness of description logics and predicate logics. *Artificial Intelligence*, 82(1-2):353–367, 1996.

[4] Ronald J. Brachman, Richard E. Fikes, and Hector J. Levesque. KRYPTON: A functional approach to knowledge representation. *Computer*, 16(10):67–73, October 1983.

[5] Rodney A. Brooks. A robust layered control system for a mobile robot. *IEEE Journal of Robotics and Automation*, RA-2(1):14–23, March 1986.

[6] Saša Buvač. Quantificational logic of context. In *Proc. Natl Conf. on Artificial Intelligence (AAAI '96)*, 1996.

[7] M. Cadoli, T. Eiter, and G. Gottlob. An efficient method for eliminating varying predicates from a circumscription. *Artificial Intelligence*, 54(3):397–410, April 1992.

[8] Peter Clark and Bruce Porter. Building concept representations from reusable components. In *Proc. Nat'l Conf. on Artificial Intelligence (AAAI '97)*, pages 369–376, 1997.

[9] William Craig. Linear reasoning. a new form of the herbrand-gentzen theorem. *Journal of Symbolic Logic*, 22:250–268, 1957.

[10] Francesco M. Donini, Maurizio Lenzerini, Daniele Nardi, and Werner Nutt. The complexity of concept lanauges. In *KR91 Proceedings*, 1991.

[11] Herbert B. Enderton. *A Mathematical Introduction to Logic*. Academic Press, 1972.

[12] Richard Fikes. Personal communication, 1997.

[13] Richard Fikes, Adam Farquhar, and James Rice. Tools for assembling modular ontologies in ontolingua. In *Proc. Nat'l Conf. on Artificial Intelligence (AAAI '97)*, pages 436–441. AAAI Press, 1997.

[14] Ken Forbus. Qualitative reasoning. In *CRC Handbook of Computer Science and Engineering*. CRC, 1996.

[15] Natalya Fridman-Noy and Carole D. Hafner. The state of the art in ontology design. *AI Magazine*, 18(3):53–74, 1997.

[16] Chiara Ghidini and Luciano Serafini. Distributed first order logic. In *Proceedings of Frontiers of Combining Systems (FroCoS'98)*, Amsterdam, October 1998. Also IRST-Technical Report #9804-02.

[17] Fausto Giunchiglia and Chiara Ghidini. Local models semantics, or contextual reasoning = locality + compatibility. In *Principles of Knowledge Representation and Reasoning: Proc. Sixth Int'l Conference (KR '98)*, 1998.

[18] Fausto Giunchiglia and Luciano Serafini. Multilanguage hierarchical logics, or: how we can do without modal logics. *Artificial Intelligence*, 65(1):29–70, 1994.

[19] Thomas R. Gruber. A translation approach to portable ontology specifications. *Knowledge Aquisition*, 5(2):199–220, 1993.

[20] R. V. Guha. *Contexts: A Formalization and Some Applications*. PhD thesis, Stanford University, 1991. Also published as technical report STAN-CS-91-1399-Thesis, and MCC Technical Report Number ACT-CYC-423-91.

[21] R. V. Guha and Douglas B. Lenat. Cyc: A midterm report. *AI Magazine*, 11(3):33–59, 1990.

[22] Joakim Gustafsson and Patrick Doherty. Embracing occlusion in specifying the indirect effects of actions. In *Proc. 5th Intl Conf. on Knowledge Representation and Reasoning (KR '96)*, pages 87–98, 1996.

[23] Katzumi Inoue. Linear resolution for consequence finding. *Artificial Intelligence*, 56(2-3):301–353, August 1992.

[24] Michael Kifer, Georg Lausen, and James Wu. Logical foundations of object-oriented and frame-based languages. *Journal of the ACM*, 42(4):741–843, July 1995.

[25] Daphne Koller and Avi Pfeffer. Object-oriented Bayesian networks. In *Proc. of the 13th Conf. on Uncertainty in Artificial Intelligence (UAI-97)*, pages 302–313. Morgan Kaufmann, 1997.

[26] S. Kraus, D. Lehmann, and M. Magidor. Nonmonotonic Reasoning, Preferential Models and Cumulative Logics. *Artificial Intelligence*, 44(1):167–207, 1990.

[27] Georg Lausen and Gottfried Vossen. *Models and Languages of Object-Oriented Databases*. Addison-Wesley, 1998.

[28] Douglas B. Lenat. Cyc: A large-scale investment in knowledge infrastructure. *Communications of the ACM*, 38(11):33–38, 1995.

[29] Douglas B. Lenat and R. V. Guha. *Building Large Knowledge-Based Systems: Representation and Inference in the CYC Project*. Addison-Wesley, 1990.

[30] Vladimir Lifschitz. Circumscription. In J.A.Robinson D.M. Gabbay, C.J.Hogger, editor, *Handbook of Logic in Artificial Intelligence and Logic Programming, Volume 3: Nonmonotonic Reasoning and Uncertain Reasoning*. Oxford University Press, 1993.

[31] D. Maier. A logic for objects. In J. Minker, editor, *Workshop on Foundations of Deductive Databases and Logic Programming*, pages 6–26, 1986.

[32] J. McCarthy. Applications of circumscription to formalizing common-sense knowledge. *Artificial Intelligence*, 28:89–118, 1986.

[33] John McCarthy. Notes on Formalizing Context. In *IJCAI-93*, 1993. Available on http://www-formal.stanford.edu/jmc/.

[34] John McCarthy and Patrick J. Hayes. Some Philosophical Problems from the Standpoint of Artificial Intelligence. In B. Meltzer and D. Michie, editors, *Machine Intelligence 4*, pages 463–502. Edinburgh University Press, 1969.

[35] Bertrand Meyer. *Object-Oriented Software Construction*. Prentice-Hall, 2nd edition, 1997.

[36] Leora Morgenstern. Inheriting well-formed formulae in a formula-agumented semantic network. In Luigia Carlucci Aiello, Jon Doyle, and Stuart Shapiro, editors, *Proc. 5th Intl Conf. on Knowledge Representation and Reasoning (KR '96)*, pages 268–279, San Francisco, November 5–8 1996. Morgan Kaufmann.

[37] Leora Morgenstern. Inheritance comes of age: Applying nonmonotonic techniques to problems in industry. *Artificial Intelligence*, 103(1–2):237–271, 1998.

[38] Chris Moss. *Prolog++, The Power of Object-Oriented and Logic Programming*. Addison-Wesley, 1994.

[39] Erik Sandewall. *Features and Fluents*. Oxford University Press, 1994.

[40] Ehud Shapiro and A. Takeuchi. Object oriented programming in concurrent prolog. *New Generation Computing*, 1:25–48, 1983.

[41] Richmond Thomason, Jeff Horty, and D. S. Touretzky. A Calculus for Inheritance in Monotonic Semantic Nets. Research Note CMU-CS 86-138, Carnegie Mellon, 1986.

A Proofs

A.1 Equivalent Semantics for Directional OOFOL

The proof relies on the following lemma. It says that we can weaken the circumscribed theory or the set of varied constant symbols without losing models.

Lemma A.1 *Let A be a FOL theory and P, Z, Z' be vectors of non-logical symbols (P has only predicates). For a structure M, if $M \models Circ[A; P; Z, Z']$, then $M \models Circ[A; P; Z]$.*

Proof Let $M \models Circ[A; P; Z, Z']$. Then M is a $\leq^{P;Z,Z'}$-minimal model satisfying A ($M \leq^{P;Z,Z'} M' \iff (|M| = |M'| \wedge \forall C \notin P \cup Z \cup Z', C^M = C^{M'} \wedge P^M \subseteq P^{M'})$). If $M' <^{P,Z;Z'} M$ such that $M' \models A$, then $|M'| = |M|$, $P^M \subsetneq P^{M'}$ and for every constant symbol C not in P, Z, Z', $C^M = C^{M'}$. Thus, in particular, for every constant symbol C not in P, Z, $C^M = C^{M'}$. Therefore $M' <^{P,Z} M$ and we have contradiction to M being $\leq^{P;Z,Z'}$-minimal model satisfying A. ■

The converse is not true, as it entails monotonicity of circumscription. For our specific case, though, we can get a version of the opposite direction. Let $\mathcal{A}(Obj) = \widetilde{\mathcal{A}}(Obj, Obj)$ and $\mathcal{L}(Obj) = \mathcal{L}(\mathcal{A}(Obj))$.

Our theorem can be restated as follows.

Theorem A.2 *Let T be a totally ordered OOFOL theory. $\mathcal{M}$ is a model of*

$$\bigwedge_{i\leq n} Circ[\widetilde{\mathcal{A}}(O_i, O_{i+1}); Ab_i; \mathcal{L}(O_{i+1})]$$

if and only if it is a model of

$$\textstyle\bigwedge_{i\leq n} Circ[flat(T); Ab_i; Ab_{i+1}, ..., Ab_n, \mathcal{L}(O_{i+1}), ..., \mathcal{L}(O_n)].$$

PROOF The backward direction follows directly from lemma A.1 and noticing that $Circ[flat(T); Ab_i; \mathcal{L}(O_{i+1})] \equiv Circ[\widetilde{\mathcal{A}}(O_i, O_{i+1}); Ab_i; \mathcal{L}(O_{i+1})] \wedge flat(T)$ (See formula (3.2) in [30]).

For the forward direction, assume that

$$\begin{array}{l} M \models \bigwedge_{j\leq n} Circ[\widetilde{\mathcal{A}}(O_j, O_{j+1}); Ab_j; \mathcal{L}(O_{j+1})], \text{ and} \\ M \not\models Circ[flat(T); Ab_i; Ab_{i+1}, ..., Ab_n, \mathcal{L}(O_{i+1}), ..., \mathcal{L}(O_n)] \end{array}$$

for some $i \leq n$. Let i be the first (smallest) such index. For brevity, let us write Z for $Ab_{i+1}, ..., Ab_n, \mathcal{L}(Obj_{i+1}), ..., \mathcal{L}(Obj_n)$. $M \models flat(T)$, because $flat(T) = \bigcup_i \widetilde{\mathcal{A}}(O_i, O_{i+1})$. Thus, there is a model $M' <^{Ab_i;Z} M$ such that $M' \models flat(T)$.

Take M'' such that $|M''| = |M|$ (i.e., they have the same universe), the interpretation of all the symbols of $\mathcal{L}(T)$ other than $\mathcal{L}(O_{i+1}), Ab_i$ is identical to M and the interpretation of $\mathcal{L}(O_{i+1}), Ab_i$ is as in M' (this is sound since $|M'| = |M|$ by the conditions of $M' \leq^{Ab_i;Z} M$).

$M'' <^{Ab_i;Ab_{i+1},\mathcal{L}(O_{i+1})} M$, because $|M| = |M''|$, $C^M = C^{M''}$ for every constant symbol C not in $\mathcal{L}(O_{i+1})$ (since this is how we set the interpretation for these symbols in M'') and $Ab_i^{M''} \subsetneq Ab_i^M$ (since $M' <^{Ab_i;Z} M$). To see that $M'' \models \widetilde{\mathcal{A}}(O_i, O_{i+1})$, notice that M'' is identical to M' on $\mathcal{L}(\widetilde{\mathcal{A}}(O_i, O_{i+1}))$.

Thus, M is not a model of $Circ[\widetilde{\mathcal{A}}(O_i, O_{i+1}); Ab_i; \mathcal{L}(O_{i+1})]$, contradicting our assumption. ■

A.2 Completeness Proof for Directional OOFOL

To prove the completeness theorem for MESSAGE-PASSING (Theorem 5.3), we need several lemmas.

Lemma A.3 *Let O_1, O_2 be two objects and $\mathcal{M}$ a model of $\widetilde{\mathcal{A}}(O_1, O_2)$. Assume that for every sentence $\phi \in \mathcal{L}_{O_1,O_2}$, $\mathcal{M} \models \phi$ iff $\mathcal{M} \models \phi'$, where ϕ' is the translation of ϕ from O_1 to O_2. Then, there is a model $\mathcal{N}$ such that for all $ab \in Ab_{O_1,O_2}$, $\mathcal{N} \models \forall x.\neg ab(x)$ and $\mathcal{N} \leq^{Ab_{O_1,O_2};\mathcal{L}(O_2)} \mathcal{M}$.*

PROOF Assume that $\mathcal{L}$ has a single sort, that the only link between O_1, O_2 is a single predicate P_1, P_2 of O_1, O_2, respectively, whose parameter is a single variable (we generalize over all of these assumptions at the end of the proof).

Let $||\mathcal{M}||$ denote the cardinality (finite or infinite) of the universe of $\mathcal{M}$ and, for a predicate P, let $|P|^{\mathcal{M}}$ denote the cardinality of $\{x \mid x \in |M|\ P^{\mathcal{M}}(x)\}$.

If $|P_1|^{\mathcal{M}} = |P_2|^{\mathcal{M}}$ and $|P_1^c|^{\mathcal{M}} = |P_2^c|^{\mathcal{M}}$ ($P^c(x) \iff \neg P(x)$), take f to be a bijection from $|\mathcal{M}|$ to itself such that f is a 1-1 and onto mapping of the elements of $P_1^{\mathcal{M}}$ on those of $P_2^{\mathcal{M}}$. Let $\mathcal{N}$ have the same universe and the same interpretation of $\mathcal{L}(O_1)$ as $\mathcal{M}$. For every relation R in $\mathcal{L}(O_2)$, let $R^{\mathcal{N}}(x_1,...,x_k) \iff R^{\mathcal{M}}(f(x_1),...,f(x_k))$. Finally, let $\forall x \in |\mathcal{N}|.\neg ab^{\mathcal{N}}(x)$. This model is easily seen to be the required one.

Otherwise, $|P_1|^{\mathcal{M}} \neq |P_2|^{\mathcal{M}}$ or $|P_1^c|^{\mathcal{M}} \neq |P_2^c|^{\mathcal{M}}$. If there is a model $\mathcal{M}'$ of O_2 such that $|\mathcal{M}| = |\mathcal{M}'|$ and the cardinalities of P_2, P_2^c match those of P_1, P_1^c as above, then we take $\mathcal{N}$ as in the previous case, setting $R^{\mathcal{N}}(x_1,...,x_k) \iff R^{\mathcal{M}'}(f(x_1),...,f(x_k))$, when f is a bijection as above between $|\mathcal{M}|$ and $|\mathcal{M}'|$ that is 1-1 and onto from $|P_1^{\mathcal{M}}|$ to $|P_2^{\mathcal{M}'}|$. Finally, let $\forall x \in |\mathcal{N}|.\neg ab^{\mathcal{N}}(x)$. This model is again easily seen to be the required one.

Otherwise, let $\kappa, \kappa', \kappa''$ stand for the cardinalities of $|\mathcal{M}|$, $(P_1)^{\mathcal{M}}$ and $(P_1^c)^{\mathcal{M}}$, respectively.

Assume that all of $\kappa, \kappa', \kappa''$ are finite. Then, since there is no model $\mathcal{M}'$ as above, there is a formula ξ_κ in $\mathcal{L}(O_2)$ that stands for "there are κ many elements", for each of $\kappa, \kappa', \kappa''$. Then, $O_2 \models \xi_\kappa \Rightarrow (\neg\xi_{\kappa'}(P_2) \vee \neg\xi_{\kappa''}(P_2^c))$. This formula is exactly such a ϕ' for which ϕ (the O_1-version of ϕ') satisfies $O_1 \not\models \phi$ (because $\mathcal{M}$ is a counter example of ϕ). Contradicting our assumption that in $\mathcal{M}$ every sentence $\phi \in \mathcal{L}_{O_1,O_2}$ is equivalent to ϕ'.

If κ is infinite, then that means that every infinite model of O_2 satisfies $\psi = \neg\xi_{\kappa'}(P_2) \vee \neg\xi_{\kappa'}(P_2^c)$, where ξ'_κ in $\mathcal{L}(O_2)$ stands for "there are κ' many elements such that ..." If all the finite models of O_2 satisfy $\neg\psi$ then this is a contradiction to the fact that FOL cannot express "there are finitely-many elements" (by the Löwenheim-Skolem Theorem (see p.141 in [11])). Thus, there are finite cardinalities ν, ν', ν'' such that $O_2 \models \xi_{>\nu} \Rightarrow (\neg\xi_{>\nu'}(P_2) \vee \neg\xi_{\nu''}(P_2^c))$. But this is again a formula ϕ' for which ϕ satisfies $O_1 \not\models \phi$ ($\mathcal{M}$ is a counter example to ϕ). This is a contradiction to our assumption that in $\mathcal{M}$ every sentence $\phi \in \mathcal{L}_{O_1,O_2}$ is equivalent to ϕ'.

To generalize over the assumptions we made at the beginning of our proof, we need to first be able to extend the arity of the parameters of P_1, P_2 to n. This can be done using a new sort of elements that are vectors of such arity. Having the goal being stated in that sort and then "translated" to and from it to the different layers gives us the required reduction. To add more links (ab's between other nonlogical symbols), it is simple to consider the links as extending the arity of the single formula P_1. Finally, to add constant symbols and functions, notice that they are all representable using relations in the language. ■

The following seemingly innocent lemma is central to our theory as it assures that every ab that can not be proved FALSE has a provable reason. It is also particularly difficult to prove because for an arbitrary circumscribed formula it may be the case that ab has no explicit definition coming out of the circumscription.

Lemma A.4 *If $Circ[\widetilde{\mathcal{A}}(O_1,O_2); Ab_{O_1,O_2}; \mathcal{L}(O_2)] \not\models \forall x \neg ab(x)$ for some $ab \in Ab_{O_1,O_2}$, then there is $\phi \in \mathcal{L}_{O_1,O_2}$ such that $O_1 \not\models \phi$ and $O_2 \models \phi'$, where ϕ' is the translation of ϕ from O_1 to O_2.*

Furthermore, if O_1' is the result of adding all such ϕ's to the axioms of O_1, then $Circ[\widetilde{\mathcal{A}}(O_1',O_2); Ab_{O_1,O_2}; \mathcal{L}(O_2)] \models \forall x \neg ab(x)$.

Proof   Assume that for every $\phi \in \mathcal{L}_{O_1,O_2}$, $O_2 \models \phi' \implies O_1 \models \phi$. Let us take a model $\mathcal{M}$ of the circumscription in which $ab(a)$ is true for some assignment a. We show that there is a model that is preferred over $\mathcal{M}$ (thus reaching a contradiction).

If, for every sentence $\phi \in \mathcal{L}_{O_1,O_2}$, $\mathcal{M} \models \phi$ iff $\mathcal{M} \models \phi'$ (ϕ' translated as above), then Lemma A.3 guarantees the existence of a model $\mathcal{N}$ of $\widetilde{\mathcal{A}}(O_1, O_2)$ such that $\mathcal{N} \leq^{Ab_{O_1,O_2};\mathcal{L}(O_2)} \mathcal{M}$. Contradiction.

Take $\Phi = \{\phi \in \mathcal{L}_{O_1,O_2} \mid \mathcal{M} \models \phi\}$. If Φ' (the translation of Φ from O_1 to O_2) is consistent with O_2, then there is a model $\mathcal{M}'$ in which every sentence $\phi \in \mathcal{L}_{O_1,O_2}$ satisfies $\mathcal{M}' \models \phi$ iff $\mathcal{M}' \models \phi'$. Take such a model $\mathcal{M}'$ so that it also has the same universe as $\mathcal{M}$, the same interpretation for $\mathcal{L}(O_1)$ and $\mathcal{M}' \models \forall x.ab(x)$ (this is possible because the interpretations for $\mathcal{L}(O_1)$ and $\mathcal{L}(O_2)$ are *independent* given $\forall x.ab(x)$, and if there is no such model, then we found a sentence ϕ such that $O_1 \not\models \phi$ and $O_2 \models \phi'$ (similar argument to the one given in the proof of Lemma A.3)). $\mathcal{M}'$ satisfies the conditions of Lemma A.3, and thus there is a model $\mathcal{N}$ that satisfies $\forall x.\neg ab(x)$, keeps the same interpretation for $\mathcal{L}(O_1)$ and has the same universe of elements as $\mathcal{M}$. Contradiction.

If Φ' is inconsistent with $\mathcal{A}(O_2)$, then there is a finite subset of it, $\hat{\Phi}'$, that is inconsistent with $\mathcal{A}(O_2)$ (by compactness of FOL). Let $\phi' = \bigwedge \hat{\Phi}'$. Then, $O_2 \models \neg\phi'$, and $\neg\phi$ is the formula promised by our lemma.

For the second part of the lemma, let O_1' be the result of adding all those ϕ to O_1. By the first part, we get that $Circ[\widetilde{\mathcal{A}}(O_1', O_2); Ab_{O_1',O_2}; \mathcal{L}(O_2)] \models \forall x \neg ab(x)$ because otherwise, there is ϕ that O_1' does not entail (which means O_1 does not entail it) but which O_2 does entail, contradicting the way we built O_1'. ■

The following is Craig's Interpolation Theorem. It is key to any separation used in this paper.

Theorem A.5 ([9]) *If $\alpha \vdash \beta$, then there is a formula γ involving only symbols common to both α and β, such that $\alpha \vdash \gamma$ and $\gamma \vdash \beta$.*

The following lemma is a single step in our completeness proof.

Lemma A.6 *Let $T = \{O_1, O_2\}$ be a totally ordered OOFOL theory, and φ be a sentence in $\mathcal{L}(O_2)$. If $T \models \varphi$, then $O_2 \models \varphi$, or there is a sentence $\psi \in \mathcal{L}_{O_1,O_2}$ such that $O_1 \models \psi$ and $O_2 \models \psi' \Rightarrow \varphi$.*

PROOF Assume $T \models \varphi$. Then, $Circ[\widetilde{\mathcal{A}}(O_1, O_2); Ab_{O_1,O_2}; \mathcal{L}(O_2)] \models \varphi$. If $O_2 \models \varphi$ then we are done. Otherwise, there is a model M of O_2 such that $M \models \neg\varphi$.

Assume first that the circumscription entails $\forall x.\neg ab(x)$. We use Craig's Interpolation Theorem (Theorem A.5), taking $\alpha = \mathcal{A}(O_1)$ and $\beta = (\widetilde{\mathcal{A}}(O_1, O_2) \setminus \mathcal{A}(O_1) \cup \{\forall x.\neg ab(x) \mid ab \in Ab_{O_1,O_2}\}) \Rightarrow \varphi$. Since $\alpha \models \beta$, there is a formula ψ in the language $\mathcal{L}_{O_1,O_2}$ (which is the intersection of the two languages $\mathcal{L}(\alpha), \mathcal{L}(\beta)$), such that $\alpha \models \psi$ and $\psi \models \beta$. By the deduction theorem for FOL, we get that $O_1 \models \psi$ and $\psi' \wedge \mathcal{A}(O_2) \models \varphi$, where ψ' is the translation of ψ from O_1 to O_2.

Now we deal with the case that $\forall x.\neg ab(x)$ is not provable. Take Φ to be the set of all ϕ guaranteed by lemma A.4. Add Φ, the to O_1 to create O_1'. Using the first part, we know that there is a sentence ψ as required for O_1', O_2. Thus, $\{\psi\} \cup \mathcal{A}(O_2) \models \varphi$.

Take Ψ to be the set of sentences proved by O_1 in the language $\mathcal{L}_{O_1,O_2}$. $\Psi \cup \Phi' \models \psi$ because $\mathcal{A}(O_1) \cup \Phi' \models \psi$ and, by Craig's Interpolation Theorem, there is α in the language $\mathcal{L}_{O_1,O_2}$ such that $O_1 \models \alpha$ and $\alpha \models \Phi' \Rightarrow \psi$, and necessarily $\Psi \models \alpha$ (the last sentence conclusion was stated for the finite Φ'.

For an infinite case, the compactness of FOL reduces our conclusion to the one given above).

Take Ψ' to be the translation of Ψ from O_1 to O_2. Then, $\Psi' \cup \mathcal{A}(O_2) \models \varphi$. Take ψ' to be the finite subset of Ψ' needed for the proof of φ. The corresponding $\psi \subset \Psi$ is the formula needed for our lemma. ■

The following lemma says that there is no inference back (from objects further in sequence to those before them). Let $\mathcal{L}_=$ be the language including no constant/function/relation symbols.

Lemma A.7 *Let $T = \{O_i\}_{i \leq m}$. Assume that all the circumscriptions done for T in Definition 3.2 have* smooth *preference relations (i.e., every model is either minimal or has a minimal model that is preferred over it). If $T \models \varphi$, $\varphi \in \mathcal{L}(O_k)$, then*

$$T \setminus \{O_j | j \geq k\} \cup \{\psi | \psi \in \mathcal{L}_= \; T \models \psi\} \models \varphi$$

PROOF Assume $T' = T \setminus \{O_j \mid j > i\}$ and $T' \not\models \varphi$. Then, there is a minimal model $\mathcal{M} \models T'$ such that $\mathcal{M} \not\models \varphi$ ($\mathcal{M} \models \neg\varphi$). Since $T \models \varphi$, $\mathcal{M}$ is not minimal according to one of the circumscriptions, with P minimized varying Q.

Since we assume that $\leq^{P;Q}$ is smooth, there is $\mathcal{M}'$ minimal such that $\mathcal{M}' \leq^{P;Q} \mathcal{M}$. But P, Q are not in the language $\mathcal{L}(O_i)$, because this is a circumscription of some $j > i$. Thus, $\mathcal{M}'$ and $\mathcal{M}$ agree on all the symbols of $\mathcal{L}(O_i)$, and $\mathcal{M}' \models \neg\varphi$. Contradiction. ■

The proof of the theorem follows.

PROOF Assume first that $n = m$. We prove the theorem by induction on the number of objects in the theory. For the cases $n = 1, 2$ we have Lemma A.6 above. Assume that the theorem is proved for all sizes smaller than n, and we will prove it for n.

Assume $T \models \varphi$. Lemma A.6 deals with the case that φ is entailed by $Circ[\widetilde{\mathcal{A}}(O_{n-1}, O_n); Ab_{O_{n-1},O_n}; \mathcal{L}(O_n)]$.

For the case this formula does not entail φ, assume that the last circumscription entails $\forall x.\neg ab(x)$. Then,

$$\begin{array}{l}(\bigwedge_{i<n-1} Circ[\widetilde{\mathcal{A}}(O_i, O_{i+1}); Ab_{O_i,O_{i+1}}; \mathcal{L}(O_{i+1})]) \wedge \\ Circ[\widetilde{\mathcal{A}}(O_{n-1}, O_n); Ab_{O_{n-1},O_n}; \mathcal{L}(O_n)] \models \varphi\end{array}$$

Let $\Phi_{ab} = \{\forall x.\neg ab(x) \mid ab \in Ab_{O_{n-1},O_n}\}$. The above conjunctive formula is equivalent to

$$\begin{array}{l}(\bigwedge_{i<n-1} Circ[\widetilde{\mathcal{A}}(O_i, O_{i+1}); Ab_{O_i,O_{i+1}}; \mathcal{L}(O_{i+1})]) \wedge \\ \quad (\widetilde{\mathcal{A}}(O_{n-1}, O_n) \cup \Phi_{ab}) \models \varphi\end{array}$$

Let $T' = \{O_i\}_{i<n-1} \cup \{O'_{n-1}\}$, where O'_{n-1} is the union of $\widetilde{\mathcal{A}}(O_{n-1}, O_n)$ and Φ_{ab}. By the induction hypothesis, there is the required chain of formulas $\varphi_a, ..., \varphi_{n-2}$ in the system T'. In particular, $T \models \varphi_{n-2}$ and

$$\{\varphi'_{n-2}\} \cup \widetilde{\mathcal{A}}(O_{n-1}, O_n) \cup \Phi_{ab} \models \varphi$$

Using Craig's Interpolation Theorem, there is ψ in $\mathcal{L}_{O_{n-1},O_n}$ such that $\varphi'_{n-2} \wedge \mathcal{A}(O_{n-1}) \models \psi$ and $\psi' \wedge \mathcal{A}(O_n) \models \varphi$. Putting $\varphi_{n-1} = \psi$ gives us the required induction step.

Now we deal with the case that $\forall x.\neg ab(x)$ is not provable for some $ab \in Ab_{O_{n-1},O_n}$. Take Φ to be the set of all ϕ guaranteed by Lemma A.4.

Add Φ to O_{n-1} to create O'_{n-1}. Using the first part, we know that there is a sequence of sentences as required for the changed system T' (replacing O_{n-1} by O'_{n-1}).

Take φ'_{n-1} to be the sentence that is guaranteed for O'_{n-1}, O_n. Thus, $\{\varphi'_{n-1}\} \cup A(O_2) \models \varphi$. Take Ψ to be the set of sentences proved by O_{n-1} in the language $\mathcal{L}_{O_{n-1},O_n}$. $\Psi \cup \Phi' \models \varphi'_{n-1}$ because $\mathcal{A}(O_{n-1}) \cup \Phi' \models \varphi'_{n-1}$ (by Craig's Interpolation Theorem, there is α in the language $\mathcal{L}_{O_{n-1},O_n}$ such that $O_{n-1} \models \alpha$ and $\alpha \models \Phi' \Rightarrow \varphi'_{n-1}$, and necessarily $\Psi \models \alpha$).

Take Ψ' to be the translation of Ψ from O_{n-1} to O_n. Then, $\Psi' \cup \mathcal{A}(O_n) \models \varphi_n$. Take φ'_{n-1} to be the finite subset of Ψ' needed for the proof of φ. The corresponding $\varphi_{n-1} \subset \Psi$ is the required formula for our induction step.

Lemma A.7 reduces the case of $n \neq m$ to the previous case ($n = m$). We only need to notice that $\{\psi \in \mathcal{L}_= \mid T \models \psi\} = \{\psi \in \mathcal{L}_= \mid \exists i \; Obj_i \models \psi\}$. ■