13th Workshop on Logic Programming (WLP'98), October 6–8, 1998, Vienna, Austria.
Technical Report 1843-1998-10, Institut für Informationssysteme, Technical University Vienna

# Nonmonotonic Inheritance in Object-Oriented Deductive Database Languages

Wolfgang May Paul-Th. Kandzia
{may|kandzia}@informatik.uni-freiburg.de

Institut für Informatik, Universität Freiburg,
Am Flughafen 17, D-79110 Freiburg, Germany

**Abstract**

Deductive object-oriented frameworks integrate logic rules and inheritance. There, specific problems arise: Due to the combination of deduction and inheritance, (a) deduction can take place depending on inherited facts, thus raising indirect conflicts, and (b) also the class hierarchy and -membership is subject to deduction. From this point of view, we investigate the application of the extension semantics of Default Logic to deductive object-oriented database languages. By restricting the problem to Horn programs and a special type of defaults tailored to the semantics of inheritance, a forward-chaining construction of extensions is possible. This construction is compared with a solution as implemented in the F-Logic system Florid which is based on a combination of classical deductive fixpoints and an inheritance-trigger mechanism. The paper is a condensed version of [MK98].

## 1 Introduction

In deductive object-oriented database languages, a class hierarchy and nonmonotonic inheritance is used for modeling an application domain. Facts can be derived either by classical deduction, or by *inheritance*: Assume that an object $o$ is an instance of a class $c$, and it is known that a "typical" instance of $c$ has a property $p$. Then, if it can consistently be assumed that $p$ holds for $o$, it is added to the model.

The combination of deductive rules with inheritance is significantly more complex than pure deduction or pure inheritance concepts (e.g., Description Logics), where efficient implementations exist. In this work, we study the restricted case where defeasible reasoning is only concerned with inheritance. This combination is particularly of interest in deductive object-oriented databases.

**Related Work.** In the AI community, several frameworks for nonmonotonic reasoning have been presented which implement a notion of *defaults* (for an overview, see [GHR94, Bre91]).

Nonmonotonic reasoning is integrated into logic programming with negation; such programs are evaluated wrt. well-founded semantics [VGRS88] or stable semantics [GL88, BF91], see also *extended logic programs* [Prz91, GL90]; for an overview, see [Dix95]. *Circumscription* [Lif94] uses the same syntax as first-order logic, augmented with a special predicate abnormal. In *Default Logic* [Rei80, Poo94, MT93], defeasible reasoning is expressed by *defaults*, extending the first order syntax. Default Logic is presented in more detail in Section 3. *Inheritance Networks* [Tou86, Hor94] provide a comprehensive framework for specifying *typical* or *atypical* properties. An approach to inheritance in frame systems based on circumscription is presented in [Bre87]. As a *semantic* approach, *preferential models* [Sho88, KLM90, Mak94] provide a very general formalization of nonmonotonic reasoning. Except inheritance nets, the above approaches are based on first-order syntax. By representing classes as predicates, a derived class-membership is supported.

In the deductive database community, nonmonotonic features (except strict negation) are still very rare. The paradigm of deductive object-oriented database languages conceptually includes nonmonotonic inheritance, but this is not actually integrated into the existing languages and implementations. Here, *structural* inheritance denotes a refining, but not fully overriding inheritance on the *signature* level. In contrast, *value inheritance* denotes the concept of nonmonotonic inheritance known from AI.

The early object-oriented logics focussed on complex objects, but still lacked a class-hierarchy or inheritance. A class hierarchy with only structural inheritance has been introduced in LOGRES [CC$^+$90], IQL [AK92], and ROL [Liu96]. Nonmonotonic value inheritance can be found in Gulog [DT95]. The semantics is only defined for a very restricted class of "well-defined" programs; the class hierarchy and class membership are static.

F-Logic [KLW95] supports nonmonotonic value inheritance with overriding together with a class hierarchy which can be defined by rules. The combination of a $T_P$-like operator with a trigger mechanism for handling nonmonotonic inheritance in the F-Logic system FLORID [FH$^+$97, FLO] is investigated in Section 5.

The paper is structured as follows: Section 2 introduces the syntax and semantics of F-Logic used throughout the paper and illustrates the problem arising from the combination of inheritance and deduction. In Section 3, Default Logic is introduced, a characterization of inheritance by defaults is given, and the global semantics of Default theories via *extensions* is investigated. In Section 4, we adapt the results to Default theories consisting of a Horn program and the special "Horn-like" defaults which characterize inheritance, resulting in a Herbrand-style representation of extensions. In Section 5, we present the semi-declarative semantics based on logical deduction and inheritance triggers which is defined and implemented for F-Logic. Section 6 shows the relation between the presented concepts wrt. the problem of inheritance and shows the equivalence and correctness of the F-Logic solution. Proofs can be found in [MK98].

## 2 F-Logic: Language and Basic Concepts

This work has been motivated by the problem of integrating non-monotonic *value inheritance* into the deductive object-oriented database language F-Logic (cf. [KLW95]). Here, a short summary of the relevant part of the F-Logic syntax and semantics is given (we omit parameterized and multivalued methods).

- The alphabet consists of a set $\mathcal{F}$ of *object constructors*, playing the role of function symbols, a set $\mathcal{V}$ of variables, and several auxiliary symbols. Object constructors are denoted by lowercase letters and variables by uppercase ones.
- *id-terms* are composed from object constructors and variables. They are interpreted as elements of the universe.

In the sequel, let $O$, $C$, $D$, $M$, and $V$ denote id-terms.

- An *is-a atom* is an expression of the form $O$ isa $C$ (object $O$ is a member of class $C$), or $C :: D$ (class $C$ is a subclass of class $D$).
- The following are *object atoms*:
  - $O[M \rightarrow V]$: applying the *scalar* method $M$ to $O$ results in $V$,
  - $C[M \bullet\!\!\rightarrow V]$: $C$ provides the *inheritable scalar* method $M$. For a member $O$ isa $C$, inheritance results in $O[M \rightarrow V]$; for a subclass $C' :: C$, inheritance results in $C'[M \bullet\!\!\rightarrow V]$.
- *Formulas*, *rules*, and *programs* are defined as usual.

Note that F-Logic does not distinguish between classes, methods, and objects which uniformly are denoted by id-terms; also variables can occur at arbitrary positions of an atom.

The semantics of F-Logic extends the semantics of first-order predicate logic. Formulas are interpreted over a semantic structure. We restrict our discussion to Herbrand-interpretations where the universe consists of ground id-terms. An *H-structure* is a set of ground F-Logic atoms describing an object world, thus it has to satisfy several *closure axioms* related to general object-oriented properties:

**Definition 1 (Closure Axioms)** A set $\mathcal{H}$ of ground atoms is an *H-structure* if the following conditions hold for arbitrary ground id-terms $u, u_0, u_1, \ldots$:

- $u :: u \in \mathcal{H}$ (subclass reflexivity),
- if $u_1 :: u_2 \in \mathcal{H}$ and $u_2 :: u_3 \in \mathcal{H}$ then $u_1 :: u_3 \in \mathcal{H}$ (subclass transitivity),
- if $u_1 :: u_2 \in \mathcal{H}$ and $u_2 :: u_1 \in \mathcal{H}$ then $u_1 = u_2 \in \mathcal{H}$ (subclass acyclicity),
- if $u_1$ isa $u_2 \in \mathcal{H}$ and $u_2 :: u_3 \in \mathcal{H}$ then $u_1$ isa $u_3 \in \mathcal{H}$ (transitivity),

$*$ there are no ground terms $u'$ and $u''$ such that $u[u_1 \leadsto u'] \in \mathcal{H}$ and $u[u_2 \leadsto u''] \in \mathcal{H}$, where $\leadsto$ is $\rightarrow$ or $\bullet\!\!\rightarrow$ (uniqueness of scalar methods).

For a set $M$ of ground atoms, $\mathcal{C}\ell(M)$ denotes the closure of $M$ wrt. the above axioms, $\mathcal{C}\ell(M) = \perp$ if the constraint $(*)$ is violated in $M$.

By $\mathrm{Th}_{FL}(F)$, we denote the F-Logic theory of a set $F$ of formulas which means the closure of $F$ wrt. first-order logic and the above closure axioms. □

Positive F-Logic programs are evaluated bottom-up by a $T_P$-like operator including $\mathcal{C}\ell$, providing a minimal model semantics [KLW95]:

**Definition 2 (Deductive Fixpoint)**
For an F-Logic program $P$ and an H-structure $\mathcal{H}$,

$$\begin{aligned}
T_P(\mathcal{H}) &:= \mathcal{H} \cup \{h \mid (h \leftarrow b_1, \ldots, b_n) \text{ is a ground instance of some rule of } P \\
&\qquad\qquad \text{and } b_i \in \mathcal{H} \text{ for all } i = 1, \ldots, n\}\ , \\
T_P^0(\mathcal{H}) &:= \mathcal{C}\ell(\mathcal{H})\ , \\
T_P^{i+1}(\mathcal{H}) &:= \mathcal{C}\ell(T_P(T_P^i(\mathcal{H})))\ , \\
T_P^\omega(\mathcal{H}) &:= \begin{cases} \lim_{i\to\infty} T_P^i(\mathcal{H}) & \text{if the sequence } T_P^0(\mathcal{H}), T_P^1(\mathcal{H}), \ldots \text{ converges,} \\ \bot \ \text{otherwise.} \end{cases}
\end{aligned}$$

Note that $\mathcal{C}\ell(\mathcal{H}) = \bot$ can also lead to the result $\bot$. □

The above $T_P$-operator does not deal with inheritance. In [KLW95], *inheritance-canonic* models are defined, based on *inheritance triggers* which extend the above fixpoint semantics with some procedural flavor. This definition is further investigated in Section 5.
First, we give some introductory examples which show that logical deduction in combination with inheritance can lead to semantical difficulties and even contradictions already in simple settings:

**Example 1 (Nixon Diamond)** Consider the program

$$P = \{\mathsf{quaker[policy{\bullet\!\!\to}pacifist]},\ \mathsf{republican[policy{\bullet\!\!\to}hawk]},\ \mathsf{r\_nixon\ isa\ quaker},\ \mathsf{r\_nixon\ isa\ republican}\}.$$

This shows the problem of multiple inheritance in its original form, without additional logic rules: nothing can be derived by classical deduction. Both policies can be argued to be inherited. Each of them can be inherited without any problem, making r_nixon[policy→_] defined, "blocking" the other. □

**Example 2 (Nixon Family)**

$$P = \{\mathsf{r\_nixon\ isa\ republican},\ \mathsf{republican[policy{\bullet\!\!\to}hawk]},\ \mathsf{mrs\_nixon[policy{\to}pacifist]},\ \mathsf{mrs\_nixon[husband{\to}r\_nixon]},\ \mathsf{W[policy{\to}P]} \leftarrow \mathsf{W[husband{\to}O]}\land\mathsf{O[policy{\to}P]}\}\ .$$

Here, although there is no *direct* conflict when inheriting r_nixon[policy→hawk], the logical consequences require mrs_nixon[policy→hawk], leading to an inconsistency. Thus, a "responsible" semantics must not inherit in this situation, though leaving the policy of r_nixon undefined. □

**Example 3** Consider the following classical example:

$$P = \{\mathsf{bird[fly{\bullet\!\!\to}true;\ laying\_eggs{\bullet\!\!\to}true]},\ \mathsf{penguin[fly{\bullet\!\!\to}false]},\ \mathsf{penguin :: bird},\ \mathsf{tweety\ isa\ penguin}\}\ .$$

With the above definition, $\mathcal{C}\ell(P) = P \cup \{\mathsf{tweety\ isa\ bird}\}$. Here, tweety should inherit tweety[fly→false] from penguin, not tweety[fly→true] from bird: the potential inheritance of from bird is *preempted* by the intermediate class penguin. On the other hand, [laying_eggs$\bullet\!\!\rightarrow$true] should be inherited from bird to penguin and then to tweety[laying_eggs$\rightarrow$true]. $\square$

In the report [MK98], it is shown that a "hard-coding" into logic rules is not appropriate. In the following we show how defeasible inheritance can be integrated with the classical logic programming idea underlying F-Logic and similar deductive database languages by a solution derived from the semantics of Default Logic.

## 3 Default Logic and Inheritance

In *Default Logic* [Rei80, Poo94, MT93], defeasible reasoning is expressed by *defaults*: a default $d = \alpha : \beta_1, \ldots, \beta_n \;/\; w$ consists of a *precondition* $p(d) = \alpha$, a *justification* $J(d) = \beta = \{\beta_1, \ldots, \beta_n\}$ and a *consequence* $c(d) = w$; for a set $D$ of defaults, $J(D) = \bigcup_{d \in D} J(d)$, analogous $c(D)$. Given $\alpha$, if $\beta$ can be assumed consistently, one can conclude $w$. If $\beta$ is true, the default $\alpha{:}\beta/w$ is equivalent to the logic rule $w \leftarrow \alpha$ as long as only consistent interpretations are considered. A *default theory* is a pair $\Delta = (D, F)$ where $D$ is a set of defaults and $F$ is a set of formulas.

For characterizing inheritance, only a special form of defaults is needed, called *semi-normal defaults*; i.e., of the form $\alpha(\bar{x}){:}\beta(\bar{x})/w(\bar{x})$ where $\alpha(\bar{x})$ is a conjunction of atoms, $w(\bar{x})$ is also an atomic formula, and $\forall \bar{x} : \beta(\bar{x}) \rightarrow w(\bar{x})$ holds.

**Definition 3** For a given F-Logic program $P$, by $\Delta_P$ we denote the Horn default theory $(D_{inh}, P)$, where

$$D_{inh} := \left\{ \frac{O \;\mathsf{isa}\; C, C[M \bullet\!\!\rightarrow V] \;:\; \neg\exists C'(O \;\mathsf{isa}\; C' \wedge C' :: C) \;,\; O[M \rightarrow V]}{O[M \rightarrow V]} \;, \right.$$
$$\left. \frac{SC :: C, C[M \bullet\!\!\rightarrow V] \;:\; \neg\exists C'(SC :: C' \wedge C' :: C) \;,\; SC[M \bullet\!\!\rightarrow V]}{SC[M \bullet\!\!\rightarrow V]} \right\}$$

$\square$

### 3.1 Extensions

The semantics of a default theory is defined in terms of *extensions*. In the following, for a set $S$ of formulas, let $\mathrm{Th}(S)$ denote the theory of $S$.

**Definition 4 (Extension; based on [Poo94])** Let $\Delta = (D, F)$ be a default theory. For sets $S$, $T$ of formulas, let

$$\begin{aligned} GD(S,T,D) := \{d \mid\; & d \text{ is an instance of a default in } D,\ \mathrm{Th}(T) \models p(d) \text{ , and} \\ & \mathrm{Th}(S \cup \{\beta\}) \text{ is consistent for every } \beta \in J(d)\} \end{aligned}$$

(*generating defaults*). Then, for all sequences $S_0 = F, S_1, S_2, \ldots$ of sets of formulas s.t. $S = (\bigcup_{i=0}^{\infty} S_i)$ and

$$S_{i+1} = S_i \cup C_i \quad \text{where } C_i = c(GD(S, S_i, D)) \;,$$

Th($S$) is an *extension* of $\Delta$. Since $S$ is needed later on, we call it an *extension base* of $\Delta$. □

**Remark** $S = F \cup \bigcup_{i=0}^{\infty} C_i$ and there is no default applicable in $S$. □

In [Mak94], this is termed a *quasi-inductive* definition: in the step $i \to i{+}1$, all $\beta_j$ are required to be consistent with $\mathrm{Th}(S) = \mathrm{Th}(\bigcup_{i=0}^{\infty} S_i)$, thus, assumptions about *future* stages are made (note that in contrast, the evaluation of $\alpha$ does not use $S$). Note that, depending on which assumptions are made, there can be several *different* extensions (cf. Ex. 3).

## 3.2 Forward Chaining Evaluation

Motivated by the fixpoint semantics for positive logic programs, the evaluation of logic programs with inheritance should also be based on a forward-chaining approach, i.e. without having to guess $S$ first. From Def. 4, a forward-chaining, *inflationary* strategy can be defined by replacing "$\mathrm{Th}(S \cup \{\beta\})$ is consistent" with "$\mathrm{Th}(S_i \cup \{\beta\})$ is consistent", i.e. evaluating defaults against the current belief set (in contrast to Def. 4, in every step, we allow the application of exactly one default which is sufficient in this setting; cf. [MK98, Sec. 8]).

**Definition 5 (Inflationary extension)**
Let $\Delta = (D, F)$ be a default theory. For a theory $S$, let

$$\begin{aligned} GD(S, D) := \{d \mid\ & d \text{ is an instance of a default in } D,\ \mathrm{Th}(S) \models p(d)\ \text{, and} \\ & \mathrm{Th}(S \cup \{\beta\}) \text{ is consistent for every } \beta \in J(d)\}\ . \end{aligned}$$

Let $AD_0 = \emptyset$ and $S_0{=}F, S_1, S_2, \ldots, S_\eta$ be a sequence of sets of formulas s.t.

$$S_{i+1} = S_i \cup \{c(d_i)\}\ ,\ AD_{i+1} = AD_i \cup \{d_i\} \ \text{ where } d_i \in GD(S_i, D)\ ,$$

and $GD(S_\eta, D) = \emptyset$. Then, with $S = (\bigcup_{i=0}^{\eta} S_i)$, Th($S$) is called an *inflationary extension* of $\Delta$; we call $S$ an *inflationary extension base* of $\Delta$. □

This approach is, e.g., investigated in [MT93, Section 3.7, Def. 3.61]. As shown there, the above method is complete, but not sound: it generates theories which are no extensions.

**Proposition 1 (Extensions vs. Inflationary Extensions)**
*Let $\Delta = (D, F)$ be a Default theory.*

1. *Every extension $S$ of $\Delta$ is also an inflationary extension of $\Delta$, and*
2. *Let $T$ be an inflationary extension base computed by the algorithm given in Def. 5. If for every $\beta \in J(AD_\eta)$, $\beta$ is consistent with Th($T$), then Th($T$) is an extension of $\Delta$.* □

**Proof** cf. [MT93, Cor. 3.71 and Th. 3.65]. ■

The strategy is inflationary in the sense that a default which has been once applied is not undone (which would require to undo also all its logical consequences) if in a later step one of its *justifications* turns out to be wrong which is exactly the tested criterion in (2) of the above proposition.

This problem can be solved by either (i) forbidding the application of defaults whose justifications will be falsified later, or (ii) forbidding the application of a default whose logical consequences would falsify the justifications of another default which has been applied earlier. The notion of extensions includes (i) whereas (ii) is much easier to implement:

**Definition 6 (Cautious inflationary extension)**
Let $\Delta = (D, F)$ be a default theory. For a theory $S$, let

$$\begin{aligned} GD_{caut}(S, D, AD) := \{d \mid\ & d \text{ is an instance of a default in } D,\ \mathrm{Th}(S) \models p(d)\ , \\ & \text{and } \mathrm{Th}(S \cup c(d) \cup \beta) \text{ is consistent} \\ & \text{for every } \beta \in J(AD \cup \{d\})\}\ . \end{aligned}$$

Let $AD_0 = \emptyset$ and $S_0{=}F, S_1, S_2, \ldots, S_\eta$ be a sequence of sets of formulas s.t.

$$S_{i+1} = S_i \cup \{c(d_i)\}\ ,\ AD_{i+1} = AD_i \cup \{d_i\} \ \text{ where } d_i \in GD_{caut}(S_i, D, AD_i)\ ,$$

and $GD_{caut}(S_\eta, D, AD_\eta) = \emptyset$. Then, with $S = (\bigcup_{i=0}^{\eta} S_i)$, $\mathrm{Th}(S)$ is called a *cautious inflationary extension* of $\Delta$; we call $S$ a *cautious inflationary extension base* of $\Delta$. □

**Remark** Note that now there can be applicable defaults in $S$ (which would falsify a justification of a previously applied default). □

**Proposition 2 (Cautious Inflationary vs. Inflationary Extensions)**
*Let $\Delta = (D, F)$ be a default theory. Then,*

- *Every cautious inflationary extension $S$ of $\Delta$ can be extended to an inflationary extension. If $GD(S, D) = \emptyset$, then $S$ is an inflationary extension.*
- *If an inflationary extension $S$ satisfies Prop. 1(2) then $S$ is also a cautious inflationary extension.* □

**Proof** The computation sequences for cautious inflationary extensions given in Def. 6 are prefixes of computation sequences for inflationary extensions given in Def. 5. Thus, by applying further defaults in $D$, thereby falsifying justifications of previously applied defaults, an inflationary extension can be computed. ■

**Proposition 3 (Extensions vs. Cautious Inflationary Extensions)**
*Given a default theory $\Delta = (D, F)$, a cautious inflationary extension $S$ of $\Delta$ is an extension of $\Delta$ if $GD(S, D) = \emptyset$.* □

**Proof** By Prop. 2, every cautious inflationary extension $S$ s.t. $GD(S, D) = \emptyset$ is an inflationary extension. Since every cautious inflationary extension satisfies the additional criterion stated in Prop. 1, it is then an extension of $\Delta$. ■

## 4 The Horn Case

Given a default theory $(D, P)$ which consists of a set $P$ of Horn formulas and a set $D$ of semi-normal defaults, both in Definitions 4 and 5, every $S_i$ and the resulting base $S = P \cup \bigcup_{i=0}^{\infty} C_i$ is Horn. Thus, the semantics can equivalently be given in a Herbrand style similar to minimal models in logic programming. In the following, we consider the case of F-Logic programs and restrict ourselves to finite extensions.

**Definition 7** Given an F-Logic program $P$ and an extension base $S$ of $\Delta_P$, $\mathcal{H} := T_S^{\omega}(\emptyset)$ is called the *H-extension* of $P$ to $S$ (analogous for *inflationary H-extensions* and *cautious inflationary H-extensions*).[1] $\square$

The forward-chaining approach of Def. 5 can also be used for computing the inflationary H-extensions of a program without manipulating sets of formulas:

**Proposition 4** *Let $P$ be an F-Logic program, $\Delta_P$ its default theory according to Def. 3. For an H-structure $\mathcal{H}$ and a semi-normal default theory $\Delta = (D, F)$, let*

$$GD(\mathcal{H}, \Delta) := \{d \mid d \textit{ is a ground instance of a default in } \Delta,\ p(d) \subseteq \mathcal{H}\ ,\ \textit{and } Th_{FL}(F \cup \mathcal{H} \cup \{\beta\}) \textit{ is consistent for every } \beta \in J(d)\}\ .$$

*Let $\mathcal{H}_0, \mathcal{H}_1, \ldots, \mathcal{H}_\eta$ be a sequence of H-structures s.t. $\mathcal{H}_0 = T_P^{\omega}(\emptyset)$ and*

$$\mathcal{H}_{i+1} = T_P^{\omega}(\mathcal{H}_i \cup \{c(d_i)\})\ ,\ AD_{i+1} = AD_i \cup \{d_i\} \quad \textit{where } d_i \in GD(\mathcal{H}_i, \Delta_P)\ ,$$

*and $GD(H_\eta, \Delta) = \emptyset$. If $\mathcal{H} := \bigcup_{i=0}^{\eta} \mathcal{H}_i \neq \bot$, then $\mathcal{H}$ is an inflationary H-extension of $P$. Moreover, every inflationary H-extension can be computed by such a sequence.* $\square$

**Proof** see [MK98]. $\blacksquare$

The criterion given in Prop. 1(2) carries over to H-extensions:

**Proposition 5** *Let $\mathcal{H}$ be an inflationary H-extension computed by the above algorithm. If for every $\beta \in J(AD_\eta)$, $\beta$ is consistent with $\mathcal{H}$, then $\mathcal{H}$ is an H-extension of $\Delta$.* $\square$

Since only defaults of the form given in $D_{inh}$ are used, the only case where a justification can be annulled in later steps is when an intermediate class is inserted. We come back to this issue later. First, we describe the trigger-based inheritance mechanism extending the semantics defined for F-Logic [KLW95] which is implemented in the FLORID system.

## 5 Inheritance via Inheritance Triggers

The deductive part of F-Logic programs is evaluated wrt. an inflationary fixpoint semantics (cf. Def. 2), additionally, user-defined stratification is

[1]note that by Def. 2, $T_P^{\omega}$ includes the closure $\mathcal{C}\ell$.

supported. Non-monotonic inheritance is implemented via a trigger mechanism in a *deduction precedes inheritance* manner: The evaluation of a program is defined by alternatingly computing a classical deductive fixpoint and carrying out a specified amount of inheritance. The strategy is formally characterized as follows, based on *inheritance triggers*:

**Definition 8 (Inheritance Triggers)** Let $\mathcal{H}$ be an H-structure.

- An *inheritance trigger* in $\mathcal{H}$ is a pair $(o\sharp c, m{\bullet\!\!\to} v)$ such that $(o\sharp c) \in \mathcal{H}$ and $c[m{\bullet\!\!\to} v] \in \mathcal{H}$, and there is no $o \neq c' \neq c$ s.t. $\{o\sharp c', c' :: c\} \subseteq \mathcal{H}$ ($\sharp$ being either $\mathsf{isa}$ or $::$ ).
- An inheritance trigger $(o \mathsf{~isa~} c, m{\bullet\!\!\to} v)$ or $(c' :: c, m{\bullet\!\!\to} v)$ is *active* in $\mathcal{H}$ if there is no $v'$ s.t. $o[m{\to} v'] \in \mathcal{H}$ or $c'[m{\bullet\!\!\to} v'] \in \mathcal{H}$, respectively.
- $\mathbf{T}(\mathcal{H})$ denotes the set of active inheritance triggers in $\mathcal{H}$.
- An inheritance trigger $(o \mathsf{~isa~} c, m{\bullet\!\!\to} v)$ or $(c' :: c, m{\bullet\!\!\to} v)$ is *blocked* in $\mathcal{H}$ if $o[m{\to} v'] \in \mathcal{H}$ or $c'[m{\bullet\!\!\to} v'] \in \mathcal{H}$, respectively, for some $v' \neq v$.

Note that this definition depends only on $\mathcal{H}$, not on a program. □

The value of a method is inherited from a class to an object or a subclass only if no other value for this method can be derived for the object or the subclass, respectively. Hence, inheritance is done *after* classical deduction, leading to an alternating sequence of (deductive) fixpoint computation and inheritance steps.

**Definition 9 (Firing a Trigger)** For an H-structure $\mathcal{H}$ and an active trigger $t = (o \mathsf{~isa~} c, m{\bullet\!\!\to} v)$ or $t = (c' :: c, m{\bullet\!\!\to} v)$, the H-structure after firing $t$, $t(\mathcal{H})$, is defined as $\mathcal{H} \cup \{o[m{\to} v]\}$ or $\mathcal{H} \cup \{c'[m{\bullet\!\!\to} v]\}$, respectively.
In accordance to [KLW95], for an H-structure $\mathcal{H}$ and an active trigger $t$, $\mathcal{I}_P^t(\mathcal{H}) := T_P^\omega(t(\mathcal{H}))$ denotes the *one step inheritance transformation.* □

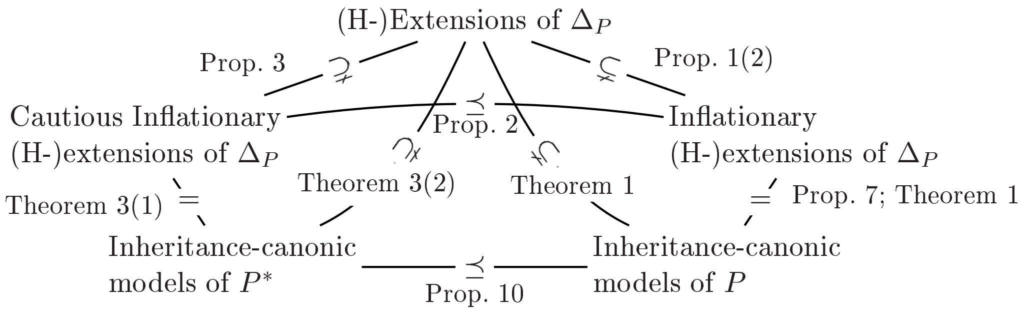
**Proposition 6 (Correctness of one-step-inheritance)** *Let $P$ be a program and $\mathcal{H}$ an H-structure which is a model of $P$ (i.e., $\mathcal{H} \models h \leftarrow b$ for every rule in $P$). For every $t \in \mathbf{T}(\mathcal{H})$, if $\mathcal{I}_P^t(\mathcal{H}) = T_P^\omega(t(\mathcal{H}))$ is consistent, then it is also a model of $P$.* □

Note that the notion of a model of an F-Logic program does not require closure wrt. inheritance (e.g., in Ex. 2 there exists no model which is closed wrt. inheritance).
In [KLW95], *inheritance-canonic* models of F-Logic programs are defined, here we reformulate the definition for finite computations:

**Definition 10 (Inheritance-Canonic Model)** (Finite variant)
For an F-Logic program $P$, a sequence $\mathcal{M}_0, \mathcal{M}_1, \ldots, \mathcal{M}_n$ of H-structures is an *$\mathcal{I}_P$-sequence* if $\mathcal{M}_0 = T_P^\omega(\emptyset)$ and for all $i$, there is a $t_i \in \mathbf{T}(\mathcal{M}_i)$ s.t. $\mathcal{M}_{i+1} = \mathcal{I}_P^{t_i}(\mathcal{M}_i)$.
An H-structure $\mathcal{M}$ is an *inheritance-canonic* model of $P$ if there is an $\mathcal{I}_P$-sequence $\mathcal{M}_0, \mathcal{M}_1, \ldots, \mathcal{M} \neq \perp$ s.t. $\mathcal{M}$ has no active triggers. □

## 6 Comparison

In this section, the relationships between the concepts of extensions, inflationary (H-)extensions, and inheritance-canonic H-structures are investigated, and criteria for isolating one class from the other are given.
In anticipation of the results of this section, these concepts compare as follows:


(H-)Extensions of $\Delta_P$
Prop. 3 $\supsetneq$
Prop. 1(2) $\subsetneq$
Cautious Inflationary (H-)extensions of $\Delta_P$
$\preceq$ Prop. 2
Inflationary (H-)extensions of $\Delta_P$
$\not\supseteq$ Theorem 3(2)
$\not\subseteq$ Theorem 1
Theorem 3(1) $=$
$=$ Prop. 7; Theorem 1
Inheritance-canonic models of $P^*$
$\preceq$ Prop. 10
Inheritance-canonic models of $P$

$M_1 \preceq M_2$ denotes that every structure/theory in $M_1$ can be extended to one in $M_2$. Proofs can be found in [MK98].

### 6.1 Inheritance-Canonic Models and Inflationary H-Extensions

The computation of inheritance-canonic models implements the process described in Prop. 4 for computing inflationary H-extensions:

**Proposition 7** *Let $P$ be an F-Logic program and $\Delta_P$ the corresponding default theory. Then the following sets coincide:*

- *the set of $\mathcal{I}_P$-sequences (cf. Def. 10) $\mathcal{M}_0, \mathcal{M}_1, \ldots, \mathcal{M}_n$ s.t. $\mathcal{M}_n \neq \perp$, and*
- *the set of prefixes $\mathcal{H}_0, \mathcal{H}_1, \ldots, \mathcal{H}_n$ of sequences of H-structures as described in Prop. 4 (computation of inflationary H-extensions).* □

**Corollary 1** *Let $P$ be an F-Logic program. Then, every (consistent) inheritance-canonic model of $P$ is an inflationary H-extension of $P$.* □

The inclusion in the other direction, i.e., that every inflationary H-extension is an inheritance-canonic model of $P$, does *not* hold since the stopping criterion is different in both approaches: The consistency check before inheriting is omitted in the definition of inheritance-canonic models.

**Definition 11** Let $\mathcal{S}_\mathcal{I}(P)$ be the set of H-structures $\mathcal{H}$ s.t. there exists an $\mathcal{I}_P$-sequence $\mathcal{M}_0, \mathcal{M}_1, \ldots, \mathcal{H}$, and $\mathcal{I}_P^t(\mathcal{H}) = \perp$ for every $t \in \mathbf{T}(\mathcal{H})$. □

**Theorem 1 ($\mathcal{I}_P$-sequences and inflationary H-Extensions)**

- *$\mathcal{S}_\mathcal{I}(P)$ is the set of inflationary H-extensions of $P$.*
- *An H-structure $\mathcal{H} \in \mathcal{S}_\mathcal{I}(P)$ is an H-extension of $P$ if and only if there is an $\mathcal{I}_P$-sequence $\mathcal{M}_0, \mathcal{M}_1, \ldots, \mathcal{H}$ which satisfies Prop. 5.* □

## 6.2 Cautious Inflationary Extensions for Inheritance

The relationship between extensions and inflationary extensions has been clarified by Prop. 1(2), giving a criterion for identifying inflationary extensions which are no extensions: an inflationary extension $S$ is an extension if every justification of every default which is applied in the computation of $S$ is consistent with $\mathrm{Th}(S)$. By the concept of cautious inflationary extensions this property has been enforced allowing a forward-chaining construction.

For defaults of the form $D_{inh}$ occuring in the default theory of an F-Logic program, the only justification which can be invalidated by later steps is the non-existence of an intermediate class. Thus, the inflationary semantics differs from the "real" semantics only when *after* inheritance, the existence of an intermediate class is derived.

**Proposition 8 (Static Class Hierarchy)** *For an F-Logic program $P$ with a static class hierarchy, i.e. no* isa *-atom or* :: *-atom occurs in any non-fact rule head, the set of extensions of $\Delta_P$ and the set of inflationary extensions of $\Delta_P$ coincide.* □

In presence of a non-static class hierarchy, the above effect can be termed as *postemption*[2]. Cautious inflationary computations can be enforced by augmenting the consequence of the defaults by their justifications. For implementing inheritance, with every instance of inheritance, the class hierarchy at this point is fixed by forbidding the introduction of an intermediate class. This is accomplished by the following modification of the inheritance default schema (analogous for subclass inheritance), blocking the later introduction of an intermediate class, obtaining a *normal default* $\alpha{:}\beta/\beta$:

$$D^*_{inh} := \frac{O \text{ isa } C, C[M \bullet\!\!\to V] \ : \ \neg\exists C'(O \text{ isa } C' \wedge C' :: C)\ ,\ O[M \to V]}{\neg\exists C'(O \text{ isa } C' \wedge C' :: C)\ ,\ O[M \to V]}$$

For an F-Logic program $P$, let $D^*_P$ be defined like $D_P$ with $D^*_{inh}$ instead of $D_{inh}$. Normal defaults guarantee the following:

**Proposition 9** *For an F-Logic program $P$, every inflationary extension of $\Delta^*_P$ is also an extension of $\Delta^*_P$.* □

**Theorem 2** *Given an F-Logic program $P$, there is a mapping $\phi$ from the extensions of $\Delta_P$ to the (inflationary) extensions of $\Delta^*_P$ such that $\phi(S)$ augments $S$ exactly by the explicit knowledge about the absence of intermediate classes in some places of the class hierarchy.* □

Note that the consequences in $D^*_{inh}$ are no longer sets of atoms. Thus, $D^*_{inh}$ cannot be directly translated to H-extensions and inheritance-canonic models. In the following section, these notions are integrated by extending the program appropriately.

[2]in contrast to preemption, where inheritance is not applied due to an already known intermediate class.

## 6.3 Inheritance-Canonic Models and Cautious Extensions

The intended semantics of an F-Logic program $P$ are the H-extensions of $\Delta_P$. Thus, the set of $\mathcal{I}_P$-sequences has to be restricted to sequences where no postemption occurs, resulting in H-extensions or at least in cautious inflationary H-extensions.
The effect of cautious computations can be implemented by adding a rule

$$r(t) \ := \ \mathsf{inconsistent} \leftarrow \mathsf{o} \sharp \ \mathsf{C}, \ \mathsf{C} :: \mathsf{c}, \ \mathsf{not} \ (\mathsf{c}{=}\mathsf{C}).$$

as an *integrity constraint* to the program whenever an inheritance trigger $t = (o\sharp c, m {\bullet\!\!\to} v)$ is fired. In subsequent inheritance steps, this rule derives an inconsistency whenever an intermediate class would be derived. This requires only a slight modification in the concept of $\mathcal{I}_P$-sequences:

**Definition 12**
For an F-Logic program $P$, a sequence $\mathcal{M}_0, \mathcal{M}_1, \ldots, \mathcal{M}_n$ of H-structures is an $\mathcal{I}_P^*$*-sequence* if $\mathcal{M}_0 = T_P^\omega(\emptyset)$ and for all $i$, there is a $t_i \in \mathbf{T}(\mathcal{M}_i)$ s.t. $\mathcal{M}_{i+1} = \mathcal{I}_{P_{i+1}}^{t_i}(\mathcal{M}_i) \neq \perp$ where $P_0 = P$ and $P_{i+1} = P_i \cup r(t_i)$.
Let $S_{\mathcal{I}}^*(P)$ be the set of H-structures $\mathcal{H}$ such that there exists an $\mathcal{I}_P^*$-sequence $\mathcal{M}_0, \mathcal{M}_1, \ldots, \mathcal{M}_n = \mathcal{H}$, and $\mathcal{I}_{P_n \cup r(t)}^t(\mathcal{H}) = \perp$ for every $t \in \mathbf{T}(\mathcal{H})$. □

**Proposition 10 ($\mathcal{I}_P$- and $\mathcal{I}_P^*$-sequences)** *Let $P$ be an F-Logic program.*
*1. for every $\mathcal{I}_P^*$-sequence $\mathcal{M}_0, \mathcal{M}_1, \ldots, \mathcal{M}_n$, every $\mathcal{M}_i$ is a model of $P$.*
*2. every $\mathcal{I}_P^*$-sequence satisfies Prop. 5.*
*3. every $\mathcal{I}_P^*$-sequence is a prefix of an $\mathcal{I}_P$-sequence.* □

**Theorem 3 ($\mathcal{I}_P^*$-sequences and cautious H-Extensions)**
*Let $P$ be an F-Logic program. Then,*
*1. $\mathcal{S}_{\mathcal{I}}^*(P)$ is the set of cautious inflationary H-extensions of $P$.*
*2. for every $\mathcal{H}$ in $\mathcal{S}_{\mathcal{I}}^*(P)$, there is a $\mathcal{H}'$ in $\mathcal{S}_{\mathcal{I}}(P)$ s.t. $\mathcal{H} \subseteq \mathcal{H}'$.*
*3. for every $\mathcal{H}$ in $\mathcal{S}_{\mathcal{I}}^*(P)$, if $\mathcal{I}_P^t(\mathcal{H}) = \perp$ for every $t \in \mathbf{T}(\mathcal{H})$, then $\mathcal{H}$ is an H-extension of $P$.* □

**Complexity** Thus, with the strategies of $\mathcal{I}_P$- and $\mathcal{I}_P^*$-sequences, the semantics of cautions and non-cautious *inflationary* extensions can be implemented in F-Logic in a forward-chaining way.

As long as no object creation takes place, every $T_P^\omega$ computation is polynomial. Since the number of potential triggers is also polynomial, an H-extension $\mathcal{H} \in \mathcal{S}_{\mathcal{I}}(P)$ (or $\mathcal{S}_{\mathcal{I}}^*(P)$) can be computed in polynomial time. With object creation, the computations can become infinite.

## 7 Conclusion

We have shown how inheritance can be integrated into a deductive object-oriented database language. By considering the Horn fragment (i.e. logic programming rules) and restricting the use of defaults to the object-oriented notion of inheritance, we could tailor the semantics to the requirements in this area. Given a program $P$, the presented algorithm computes those Herbrand-like structures which represent the extensions of the default theory corresponding to $P$.

## References

[AK92] S. Abiteboul and P. C. Kanellakis. Object Identity as a Query Language Primitive. In *Building an Object-Oriented Database System – The Story of $O_2$*, ch. 5, pp. 98–127. Morgan Kaufmann, 1992.

[BF91] N. Bidoit and C. Froidevaux. Negation by default and unstratifiable logic programs. *Theoretical Computer Science*, 78:85–112, 1991.

[Bre87] G. Brewka. The Logic of Inheritance in Frame Systems. In *Intl. Joint Conference on Artificial Intelligence*, pp. 483–488, 1987.

[Bre91] G. Brewka. *Nonmonotonic Reasoning: Logical Foundations of Commonsense.* Cambridge University Press, 1991.

[CC$^+$90] F. Cacace, S. Ceri, S. Crespi-Reghizzi, L. Tanca, and R. Zicari. Integrating Object-Oriented Data Modeling with a Rule-Based Programming Paradigm. In , *ACM SIGMOD*, pp. 225–236, 1990.

[Dix95] J. Dix. Semantics of Logic Programs: Their Intuitions and Formal Properties. In A. Fuhrmann and H. Rott, editors, *Logic, Action and Information.* de Gruyter, 1995.

[DT95] G. Dobbie and R. Topor. On the Declarative and Procedural Semantics of Deductive Object-Oriented Systems. *Journal of Intelligent Information Systems*, 4(2):193–219, 1995.

[FH$^+$97] J. Frohn, R. Himmeröder, P.-T. Kandzia, G. Lausen, and C. Schlepphorst. FLORID: A Prototype for F–Logic. In *Intl. Conference on Data Engineering (ICDE)*, 1997.

[FLO] The FLORID Home Page. `http://www.informatik.uni-freiburg.de/~dbis/florid/`.

[GHR94] D. M. Gabbay, C. J. Hogger, and J. A. Robinson. *Handbook of Logic in Artificial Intelligence and Logic Programming.* Oxford Science Publications, 1994.

[GL88] M. Gelfond and V. Lifschitz. The Stable Model Semantics for Logic Programming. In *Proc. ICLP*, pp. 1070–1080, 1988.

[GL90] M. Gelfond and V. Lifschitz. Logic Programs with Classical Negation. In *Proc. ICLP*, pp. 579–597, MIT Press, 1990.

[Hor94] J. F. Horty. Some direct Theories of Nonmonotonic Inheritance. In [GHR94].

[KLM90] S. Kraus, D. Lehmann, and M. Magidor. Nonmonotonic Reasoning, Preferential Models and Cumulative Logics. *Artificial Intelli-*

*gence*, 44:167–207, 1990.

[KLW95] M. Kifer, G. Lausen, and J. Wu. Logical Foundations of Object-Oriented and Frame-Based Languages. *Journal of the ACM*, 42(4):741–843, July 1995.

[Lif94] V. Lifschitz. Circumscription. In [GHR94].

[Liu96] M. Liu. ROL: A Deductive Object Base Language. *Information Systems*, 21(5):431–457, 1996.

[Mak94] D. Makinson. General Patterns in Nonmonotonic Reasoning. In [GHR94].

[MK98] W. May and P.-T. Kandzia. Nonmonotonic Inheritance in Object-Oriented Deductive Database Languages. Technical report, Universität Freiburg, Institut für Informatik, 1998. Available from `http://www.informatik.uni-freiburg.de/~dbis/Publications/98/Inheritance.html`.

[MT93] V. W. Marek and M. Truszczyński. *Nonmonotonic Logic*. Springer, 1993.

[Poo94] D. Poole. Default Logic. In [GHR94].

[Prz91] T. Przymusinski. Stable Semantics for Disjunctive Programs. *New Generation Computing*, (9):401–424, 1991.

[Rei80] R. Reiter. A Logic for Default Reasoning. *Artificial Intelligence*, 12(1,2):81–132, 1980.

[Sho88] Y. Shoham. *Reasoning about Change*. MIT Press, 1988.

[Tou86] D. Touretzky. *The Mathematics of Inheritance*. Morgan-Kaufmann, Los Altos, CA, 1986.

[VGRS88] A. Van Gelder, K. Ross, and J. Schlipf. Unfounded Sets and Well-Founded Semantics for General Logic Programs. In *Proc. ACM PODS*, pp. 221–230, 1988.