A Method for the Management of Exceptions in Multiple Inheritance Systems

C. Oussalah, M. Magnan and L. Torrès

L.E.R.I.
Parc Scientifique Georges Besse, 30000 Nimes France
E-mail: oussalah@eerie.fr Fax: 66.84.05.06

Abstract

In this paper, we are interested in multiple inheritance systems with exceptions in object-oriented languages. Two types of exceptions may be identified: exceptions by cancellation of an inheritance link and exceptions by cancellation of a property. For each type of exception, contradictions appear when there are simultaneously several paths of the inheritance graph which allow inheritance from an object or from a property to take place or not. If certain contradictions are commonly solved by masking, there are contradictions for which no common method of resolution exists. We propose a method for the management of these contradictions in object languages. This method is based on a computation of the complexity of inheritance paths which produce a contradiction.

1: Introduction

One of the main problems in knowledge representation is the representation and the management of exceptions [1] [2] [3] [4] [5]. Inheritance systems with exceptions are found just as well in object-oriented languages, in knowledge representation languages, in semantic networks and in object-oriented bases. Exceptions in inheritance systems are useful because they bring great flexibility to the knowledge representation. On the one hand, exceptions allow the amount of redundant information to be decreased and, on the other hand, they allow new information to be taken into account without questioning the pre-existing hierarchy.

We have identified two types of exceptions: exceptions by cancellation of an inheritance link and exceptions by cancellation of a property. The first type of exceptions does not allow inheritance from an object and from all of its ancestors to take place, whereas the second type indicates that an object does not have a property defined in its ancestors. For these two types of exceptions, the simultaneous presence of paths of the inheritance graph which allow inheritance from an object or a property to take place or not produces a contradiction which must be solved. Although common methods exist to solve certain contradictions, there are contradictions for which only partial solutions have been proposed. Exception management in inheritance systems therefore remains an open problem.

In this paper, we first present object-object and object-property exceptions. For each type of exception, we define the different types of contradictions which may appear and the resolution methods that exist.

We then propose a method for exception management in object languages.

Finally, we extend this method to the resolution of multiple inheritance conflicts for which there is no entirely satisfactory solution.

2: Exceptions

Research in knowledge representation has shown that it is necessary to be able to reason on incomplete knowledge, to take into account new knowledge and to treat problems of typicality and atypicality [6]. In order to do this, we have identified two types of exceptions:

1) exceptions by cancellation of an inheritance link which we call object-object exceptions
2) exceptions by cancellation of a property which we call object-property exceptions.

2.1: Object-object exceptions

2.1.1: Definition: Exceptions do not allow inheritance from an object to take place. The main effect of exception is to cause the inheritance relation to lose its transitivity. When an object does not inherit from another object, it does not inherit from this object nor from all the ancestors of this object.

- Notion of inheritability set:
  The inheritability set of an object is the set of objects from which it inherits. When there are no exceptions, this set regroups all the objects which belong to the hierarchy of the object [7]. When exceptions are added, it is necessary to be able to determine the set of inherited objects.
- Notion of inheritability path:
The notion of inheritability path is associated with the notion of inheritability set [8]. A path of the inheritance graph corresponds to a sequence of links of which only the last link may be negative. A path of the inheritance graph is an inheritability path if it has no negative transitivity link\(^1\). An inheritability path is said to be negative if its last link is negative, otherwise it is said to be positive.

If we consider the example in Figure 1, there is only one inheritability path between Clyde and gray: Clyde \texttt{royal elephant} \texttt{gray}. The path Clyde \texttt{royal elephant} \texttt{elephant} \texttt{gray} is a path of the inheritance graph but is not an inheritability path because it has a negative transitivity link: \texttt{royal elephant} \texttt{gray}.

A positive inheritability path between \(x\) and \(y\) is necessary in order for \(x\) to inherit from \(y\).

\textbf{Notations:} 
A positive inheritability path between \(x\) and \(y\) is noted \([x, \ldots, y, +]\), the sign + indicating that the last link of the path is an is-a link.

A negative inheritability path between \(x\) and \(y\) is noted \([x, \ldots, y, -]\), the sign - indicating that the last link of the path is an is-not-a link.

\textbf{Notation of contradictions:} 
Contradictions appear when there are simultaneously at least one positive and one negative inheritability path between \(x\) and \(y\). Three types of contradictions may be identified (cf. Figure 2).

\textbf{Type 1}: The first type concerns a contradictory redundancy because one of these two paths passes over a transitivity link of the inheritance relation. All vertices of the inheritance graph may be compared by the inheritance relation.

\textbf{Type 2}: This second type of contradiction is a generalization of the previous case [4]. Here, the transitivity link is interrupted by the node \(w\). All the objects are no longer comparable but there are at least two objects which can be compared. We have the following result:

Positive and negative inheritability paths are in the form of \([x, \ldots, y, +]\) and \([x, \ldots, z, y, -]\); \(v\) and \(z\) can be compared by the inheritance relation.

\textbf{Type 3}: The third type is an ambiguity. For this type of contradiction, all objects of the contradictory inheritability paths (except the origin and the extremity) are not comparable:

\(\forall v \in [x, \ldots, y, +]\) and \(\forall z \in [x, \ldots, y, -]\); \(v\) and \(z\) cannot be compared by the inheritance relation.

\textbf{2.1.2: Resolution of contradictions of types 1 and 2:} These two types of contradictions may be solved identically because, in both cases, positive and negative inheritability paths have respectively the following form \([x, \ldots, y, +]\) and \([x, \ldots, z, y, -]\) with \(v\) and \(z\) comparable by the inheritance relation.

Figures 3a and 3b are examples of contradictions of types 1 and 2.

The resolution of these two types of contradictions depends on whether a semantic network or an object-oriented language is involved.

In the field of semantic networks, these contradictions are solved by off-path preemption [4] which is an improvement of on-path preemption or inferential distance proposed by Touretzky [5] [8] [9].

A negative inheritability path \([x, \ldots, z, y, -]\) takes preemption over a positive inheritability path \([x, \ldots, v, y, +]\) if and only if \(z\) is a specialization of \(v\). The principle of this algorithm is therefore to give predominance to the more specific information.

In the example in Figure 3a, Clyde inherits the property of not being gray from royal elephant and the

---

\(^1\) A transitivity link joins two objects of an inheritance path which are not consecutive.
property of being gray from elephant. As royal elephant is-a elephant, Clyde is not gray.

The example in Figure 3b was proposed by Sandewall [4]. Clyde is not gray if a royal elephant and is gray if an African elephant. African elephants are not explicitly gray, they inherit this property from elephant. As royal elephant is-a elephant, Clyde is not gray.

The inheritance link x - y (resp. x - y) is translated by x having the property "I am a y" (resp. "I am not a y"). Inheritance is therefore a property which can be inherited, masked or present conflicts [7]. As there are objects which may be compared by the inheritance relation for the two types of contradictions, masking may therefore be applied on inheritability properties.

Let us consider Figures 4a and 4b which represent the same inheritance graphs of those given in Figures 3a and 3b but where each node is noted by the inheritability properties it has. In both cases, the property 'gray' in royal elephant overrides the property gray in elephant: Clyde is not gray.

2.1.3: Resolution of contradictions of type 3:

The best known example of object-object contradictions is called the Nixon diamond (cf. Figure 5).

Unlike contradictions of types 1 and 2, no common method exists to solve object-object contradictions. For this type of contradiction, we obtain the result: \( \forall x \in \{ x \} \) and \( \forall z \in \{ y \} \); \( x \) and \( z \) cannot be compared by the inheritance relation. Therefore, none of the previous techniques allows the resolution of object-object contradictions because they are based on the comparison of objects of the paths. Object-object contradictions constitute the crucial problem of managing object-object exceptions.

For this type of contradiction, two attitudes may be identified: the skeptical attitude and the credulous attitude [5].

The skeptical attitude consists in drawing no conclusions in the presence of object-object contradictions. For example, no conclusion is given for Nixon's pacifism, as long as the contradiction is not removed by adding additional information [3].

On the other hand, the credulous attitude tries to draw as many conclusions as possible. Two strategies are therefore possible:
- All conclusions are generated and isolated from one another [8].
- A conclusion is chosen arbitrarily either in giving priority to the affirmation or in giving priority to the negation.

2.2: Object-property exceptions

2.2.1: Definition:

Object-property exceptions indicate that an object does not have a property defined in its ancestors. The object is then said to be atypical [6].

An object-property exception in x on the property P is noted as except(P) and means that x does not have property P.

An exception except(P) in x can be seen as a special value given to the property P. This value of property P defined in x therefore overrides the values of P defined in the ancestors of x but may also be overridden by the values of P defined in the descendants of x. This allows an object-property exception to be cancelled.

In Figure 6, the class v does not have the property P because the exception in x is more specific than the declaration of P in y. In Figure 7, the class x which
declares property P overrides the object-property exception defined in y and therefore v inherits P from x.

In the presence of object-property exceptions, the problem consists in determining the set of properties which are inherited by an object.

2.2.2: Problematics of object-property exceptions: The problematics of object-property exceptions is linked to multiple inheritance. Contradictions appear when there are simultaneously several positive inheritability paths [x, ..., z, +I] and [x, ..., y, +] where z excepts property P and y possesses property P.

Two classes of contradictions may be distinguished. The first is the class of contradictions which can be solved by masking, i.e. the objects which possess and except the property are comparable by the inheritance relation. In the example in Figure 8, the exception on property P defined in z is more specific than the declaration of P in y; x does not have property P.

![Figure 8](image1)

![Figure 9](image2)

The second class of contradictions cannot be solved by masking; objects which possess and except the property are indeed incomparable by the inheritance relation (cf. Figure 9). We call this type of contradiction an object-property contradiction.

The management of object-property contradictions constitutes the main problem of the management of object-property exceptions. As for object-object exceptions, some give priority to the affirmation or, on the contrary, to the negation.

3: A method for the management of exceptions

We present a method for the management of multiple inheritance systems with object-object and object-property exceptions. As we have seen before, problems for the management of these two types of exceptions are object-object and object-property contradictions.

3.1: Bipolar, nonmonotonic and heterogeneous multiple inheritance systems

According to the classification given by Touretzky [5], the inheritance systems that we consider are bipolar, nonmonotonic and heterogeneous multiple inheritance systems.

These systems are bipolar because inheritance links are either positive (is-a links) or negative (is-not-a links). Negative links are useful for the representation of object-object exceptions.

These links may also be either strict links, i.e. they cannot be cancelled, or defeasible links, i.e. they may be excepted [2] [11]. The system is therefore said to be heterogeneous. Strict links represent universal truths while defeasible links represent generally accepted truths.

Four types of inheritance links may therefore be identified:
- strict is-a link: \[ A \rightarrow B : A \text{ is always a } B \]
- strict is-not-a link: \[ A \nrightarrow B : A \text{ is never a } B \]
- defeasible is-a link: \[ A + B : \text{ Normally } A \text{ is a } B \]
- defeasible is-not-a link: \[ A \n+ B : \text{ Normally } A \text{ is not a } B \]

Concerning object-property exceptions, they are represented by the property except(P) which indicates that an object does not have property P.

3.2: Model for the resolution of contradictions

Our problem consists in solving for each object of the inheritance graph object-object and object-property contradictions which appear at the level of this object.

3.2.1: Stages of the model: For each object of the inheritance graph, we must:
- determine the inheritability set (stage 1)
- determine the set of inherited properties (stage 2).

It is necessary to determine the inheritability set before determining the properties inherited. Indeed, properties inherited by an object cannot be researched if the objects from which it inherits are not known.

- stage 1: The inheritability set of an object represents the set of all objects from which it inherits, i.e. the ancestors of this object. To determine the inheritability set, all contradictions which appear at the level of this object must be solved. If a common method exists to solve contradictions of types 1 and 2, such is not the case for contradictions of type 3, i.e. object-object contradictions. The main problem of this step lies in the resolution of this type of contradiction.
- stage 2: An object inherits all the properties from its ancestors except those which are excepted. If there are contradictions concerning the inheritance of a property, they must be solved in order to be able to define the set of properties effectively inherited. As we have seen, there are two classes of contradictions: contradictions solved by

---

2 A class x has property P:
- if x possesses P, i.e. P is declared in x
- or x inherits P, i.e. an ancestor of x possesses P.
A class x does not have property P if x does not possess and does not inherit P.

---
masking and object-property contradictions. In this step, object-property contradictions must be solved.

Thus, to determine the inheritability set of an object, it is necessary to have a mechanism for the resolution of object-object contradictions, and to determine the set of properties inherited, it is necessary to have a mechanism for the resolution of object-property contradictions.

3.2.2: Resolution of object-object and object-property contradictions: At each stage of the model, object-object contradictions or object-property contradictions must be solved.

Our resolution method is the same whether object-object contradictions or object-property contradictions are involved. This is possible because, in all cases, the problem is due to the simultaneous presence of different paths involving conflicting conclusions.

For object-object contradictions (respectively object-property contradictions), these paths may or may not allow inheritance from an object (respectively from a property) to take place. An object-object contradiction is characterized by the fact that one of the two inheritability paths is positive and the other is negative. In the case of object-property contradictions, all inheritability paths concerned are positive. The difference between the two paths is that one of the objects at the end of the path possesses the property whereas the other excepts it.

Our resolution method is based on a comparison of these paths.

When the contradiction is isolated, the resolution is performed in two steps:
- computation of the complexity of the paths involved
- comparison of the paths, i.e., comparison of the complexities of the paths.

Complexity of a path: The complexity of a path $\sigma$ is the pair $(\text{deg} \sigma, \chi)$ where $\chi$ depends on the type of contradiction we want to solve.

If this is an object-object contradiction:

$$\chi = \begin{cases} 2 & \text{if the last link of } \sigma \text{ is a positive strict link} \\ 1 & \text{if the last link of } \sigma \text{ is a negative strict link} \\ -1 & \text{if the last link of } \sigma \text{ is a positive defeasible link} \\ -2 & \text{if the last link of } \sigma \text{ is a negative defeasible link} \end{cases}$$

If this is an object-property contradiction:

$$\chi = \begin{cases} 2 & \text{if the last link of } \sigma \text{ is a positive strict link and } \sigma \text{ allows inheritance of the property} \\ 1 & \text{if the last link of } \sigma \text{ is a strict link and } \sigma \text{ does not allow inheritance of the property} \\ -1 & \text{if the last link of } \sigma \text{ is a defeasible link and } \sigma \text{ allows inheritance of the property} \\ -2 & \text{if the last link of } \sigma \text{ is a defeasible link and } \sigma \text{ does not allow inheritance of the property} \end{cases}$$

The value of $\chi$ depends on the type of contradiction concerned. If it is an object-object contradiction, this value translates, if the last link of the path is strict or defeasible, as positive or negative. If the last link is a strict link, the value of $\chi$ is greater than if it is a defeasible link.

Moreover, for the links having the same type, $\chi$ is greater if the link is positive.

When it is an object-property contradiction, all the paths end with a positive link. The value of $\chi$ varies according to the type of the last link and whether the path allows inheritance of the property to take place or not. If the last link is a strict link, the value of $\chi$ is greater than if it is a defeasible link. For the links having the same type, the value of $\chi$ is greater if the path allows inheritance of the property to take place.

In the example in Figure 10, an object-object contradiction concerning the birthplace of Hermann must be solved. The complexity of the path $a \rightarrow s \rightarrow r \rightarrow p$ is $(0, 2)$ because the degree of the path is 0 and the last link of the path is a positive strict link. The complexity of the path $a \rightarrow s \rightarrow q \rightarrow p$ is $(0, -2)$ because the degree of this path is 0 and the last link of this path is a negative defeasible link.

The example in Figure 11 represents an object-property contradiction on property P. The inheritability paths which bring about the contradiction are $\sigma 1 = x \rightarrow u \rightarrow v$ and $\sigma 2 = x \rightarrow y$; objects at the end of the paths are those which possess and except property P.
The complexity of \( \sigma_1 \) is \((0,-2)\) because the degree of \( \sigma_1 \) is 0 (there is one defeasible link and one strict link), the last link of the path is a defeasible link and the path does not allow inheritance of property P to take place. The complexity of \( \sigma_2 \) is \((-1,-1)\) because the degree of \( \sigma_2 \) is -1, the last link of the path is a defeasible link and the path allows inheritance of property P to take place. The path allows inheritance of property P to take place. The complexity of \( \sigma_1 \) is \((0,-2)\) because the degree of \( \sigma_1 \) is 0 (there is one defeasible link and one strict link), the last link of the path is a defeasible link and the path does not allow inheritance of property P to take place. The complexity of \( \sigma_2 \) is \((-1,-1)\) because the degree of \( \sigma_2 \) is -1, the last link of the path is a defeasible link and the path allows inheritance of property P to take place.

![Figure 10](image1)

![Figure 11](image2)

**Comparison of two paths**

Resolution of object-object contradictions and object-property contradictions are carried out in two stages. In the first stage, the complexity of the paths concerned is computed. The second stage consists in comparing the complexity of these paths to solve the contradiction.

We define an order relation on these paths noted as \( \prec_c \). This relation allows path complexities to be compared and, therefore, contradictions to be solved. In order to compare the complexity of two paths, we first compare their degree. If the degrees are not equal, the path having the greater complexity is the path whose degree is greater.

If the degrees are equal, we compare the values of the parameter \( \gamma \); the path having the greater complexity is then the path whose value of \( \gamma \) is greater.

To solve the contradiction, the path having the greater complexity is chosen.

**Comparison of two paths**: \( \sigma_1 \) and \( \sigma_2 \) are two paths having respectively the following complexity \((d_1, \gamma_1)\) and \((d_2, \gamma_2)\). We define an order relation, noted as \( \prec_c \), which allows path complexities to be compared:

\[
\sigma_1 \prec_c \sigma_2 \text{ if } (d_1 \prec d_2) \text{ or } (d_1 = d_2 \text{ and } \gamma_1 \prec \gamma_2).
\]

This relation defines a total order on the paths. It is therefore always possible to compare two inheritability paths which bring about a contradiction. This method allows all object-object or object-property contradictions to be solved.

In the example in Figure 10, \( \sigma_1 = a \rightarrow s \rightarrow r \rightarrow p \) and \( \sigma_2 = a \rightarrow s \rightarrow q \rightarrow p \). We conclude \( \sigma_2 \prec \sigma_1 \) because \( \text{deg} \sigma_2 = \text{deg} \sigma_1 \) and \( \gamma_2 < \gamma_1 \). The inheritability path which is chosen is \( \sigma_1 \); we obtain the same result as that of Horty. If the strict link \( r \rightarrow p \) is replaced by the defeasible link \( r \rightarrow p \), Horty draws no conclusion because \( \sigma_1 \) and \( \sigma_2 \) end with a defeasible link [11]. With our method, we conclude that \( a \text{-not-a}\ p \) because, in this case, \( \text{deg} \sigma_1 < \text{deg} \sigma_2 \).

Let us reconsider the Nixon diamond (cf. Figure 5). If we call \( \sigma_1 \) the path Nixon \( \longrightarrow \) Republican \( \longrightarrow \) pacifist and \( \sigma_2 \) the path Nixon \( \longrightarrow \) Quaker \( \longrightarrow \) pacifist, we have \( \text{deg} \sigma_1 = \text{deg} \sigma_2 = -2 \). As the degrees are equal, we compare the second parameter of the complexities. \( \gamma_1 = -2 \) because the last link is a negative defeasible link and \( \gamma_2 = -1 \) because the last link is a positive defeasible link.

We conclude \( \sigma_1 <_c \sigma_2 \) because \( \gamma_1 < \gamma_2 \). The inheritability path which is chosen is \( \sigma_2 \): we conclude that Nixon is pacifist.

In the example of object-property contradiction in Figure 11, \( \sigma_1 = x \rightarrow u \rightarrow v \) with the complexity \((0, -2)\) and \( \sigma_2 = x \rightarrow u \rightarrow y \) with the complexity \((-1, -1)\). If we compare the complexity of these two paths, we obtain \( \sigma_2 <_c \sigma_1 \) because \( \text{deg} \sigma_2 < \text{deg} \sigma_1 \): the inheritability path which is chosen is \( \sigma_1 \), i.e. A does not have property P.

**3.2.3: Why this choice?**

The method that we propose involves heuristics based on the computation of the complexity of inheritability paths. The paths involving the contradiction are ordered according to their complexity and the path which has the greatest complexity is chosen.

- In the case where the degrees are different, the path with the greatest complexity is the path with the greatest degree. The degree of a path is all the smaller because the path contains positive or negative defeasible links. A defeasible link is a link that can be excepted. Therefore, the more defeasible links there are, the less certain our conclusion becomes. Unlike defeasible links, strict links cannot be cancelled, i.e. cannot be excepted. The number of strict links does not have an influence on our certainty concerning the conclusion. It is for this reason that they are only counted once.

- If the degrees of the paths are equal, we choose the path which has the greatest value of \( \gamma \). The parameter \( \gamma \) is not computed in the same way depending on whether it is an object-object contradiction or an object-property contradiction that must be solved. If an object-object contradiction must be solved, the value of the parameter \( \gamma \) translates, if the last link of the path is strict or defeasible, as positive or negative. The characteristic of an object-object contradiction is that one of the two paths involved in the contradiction ends with a positive inheritance link whereas the other ends with a negative inheritance link.

Two cases are then possible:

- the two links have different types (one is a strict link, the other a defeasible link); priority is given to the strict link. Horty [11] justifies this choice by specifying that a strict link which cannot be excepted,
has priority over a defeasible link which can be excepted.
- the two links have the same type (they are all strict links or defeasible links); priority is given to the positive link, i.e. to the affirmation in order to favour inheritance.

If an object-property contradiction must be solved, the value of the parameter \( \gamma \) translates, in this case, if the last link is strict or defeasible and if the path allows inheritance of the property to take place or not. Two new cases are possible:
- the two links have different types: priority is given to the strict link for the same reasons as before.
- the two links have the same type: priority is given to the path which allows inheritance to take place.

4: Extension of the method to multiple inheritance conflicts

4.1: Multiple inheritance conflicts

Inheritance systems which we consider are multiple inheritance systems with exceptions. We have already seen how to manage exceptions and contradictions which can appear, however we are not yet interested in problems of inheritance: conflicts.

Inheritance conflicts may appear if several homonymous properties appear in the ascendants of the same object. We distinguish single inheritance conflicts in which the objects concerned may be compared by the inheritance relation and multiple inheritance conflicts in which the objects concerned cannot be compared. Single inheritance conflicts are usually solved by masking which consists in searching for the first ancestor in the object hierarchy which possesses the property [10]. Resolution of multiple inheritance conflicts is the subject of several works and remains a crucial problem for multiple inheritance systems [12] [13]. Two resolution strategies emerge from the different methods used to solve multiple inheritance conflicts [14]: linearization and resolution by the designer. The former consists in linearizing the inheritance graph in order to apply the rules of single inheritance resolution. The latter consists in leaving conflict management to the designer.

We will see that the method for resolving contradictions proposed in the previous chapters can be extended to the resolution of multiple inheritance conflicts.

4.2: Resolution of multiple inheritance conflicts

The method for the management of exceptions can be extended to the resolution of multiple inheritance conflicts because it also concerns the comparison of inheritability paths. Multiple inheritance conflicts are indeed due to the simultaneous existence of positive inheritability paths, each allowing different values to be inherited for the same property.

The resolution of multiple inheritance conflicts is achieved in two stages:
- computation of the complexity of the paths involved
- comparison of the paths, i.e. comparison of the complexities of the paths.

4.2.1: Complexity of the paths: A degree and a complexity are associated with each inheritability path. All inheritability paths which must be compared are positive.

The computation of the degree of a path is the same as that defined in §3.2.2. Only the computation of the second parameter of the complexity is modified.

\[ \gamma = \begin{cases} 1 & \text{if the last link of } \sigma \text{ is a strict link} \\ 0 & \text{if the last link of } \sigma \text{ is a defeasible link} \end{cases} \]

The value of \( \gamma \) translates as the type of the last link in the inheritability path.

4.2.2: Comparison of paths: The second stage of the resolution consists in comparing the complexity of the paths. In order to compare the complexities, we first compare the degrees. If the degrees are different, the path with the greatest complexity is the one having the greatest degree. If the degrees are equal, we compare the values of \( \gamma \); the path with the greatest complexity is then the one having the greatest value for \( \gamma \) (cf. §3.2.2).

The relation \( < \) defines here only a partial order on the paths. Indeed, the value of \( \gamma \) translates only the type of the last link in the path. If the types are the same, the paths cannot be ordered.

To solve multiple inheritance conflicts, two cases must be considered:
- the paths can be ordered; the multiple inheritance conflict is solved by choosing the path with the greatest complexity (case 1),
- the paths cannot be ordered; the management of the conflict is left to the designer (case 2).
Let us consider the inheritance graph in Figure 12. A multiple inheritance conflict exists in O6 on the property P. O6 may inherit either the value of P as defined in O2, in O4 or in O5. The value of P defined in O1 is overridden. To solve this conflict, we compare the complexity of the inheritability paths: \( \sigma_1 = O6 \rightarrow O3 \rightarrow O2 \), \( \sigma_2 = O6 \rightarrow O4 \) and \( \sigma_3 = O6 \rightarrow O5 \). The complexity of \( \sigma_1 \) is \((-2, 0)\), that of \( \sigma_2 \) is \((1, 1)\) and that of \( \sigma_3 \) \((-1, 0)\). If we compare the complexities, we obtain \( \sigma_1 <_c \sigma_3 <_c \sigma_2 \). The path \( \sigma_2 \) has the greatest complexity; the inherited value of P by O6 is v4.

5: Conclusion

Object-object and object-property contradictions constitute the crucial problem of the management of object-object and object-property exceptions. The resolution method that we propose allows all contradictions to be solved.

We have implemented this method in the object-oriented language ORL (Object Representation Language) developed in our laboratory [15].

References


