Overlaps in Requirements Engineering

George Spanoudakis[1], Anthony Finkelstein[2] & David Till[1]

[1] Department of Computing, City University, Northampton Square, London EC1V 0HB, UK

email: {gespan| till} @cs.city.ac.uk

[2] Department of Computer Science, University College London, Gower Street, London WC1E 6BT, UK

email: acwf@cs.ucl.ac.uk

Abstract

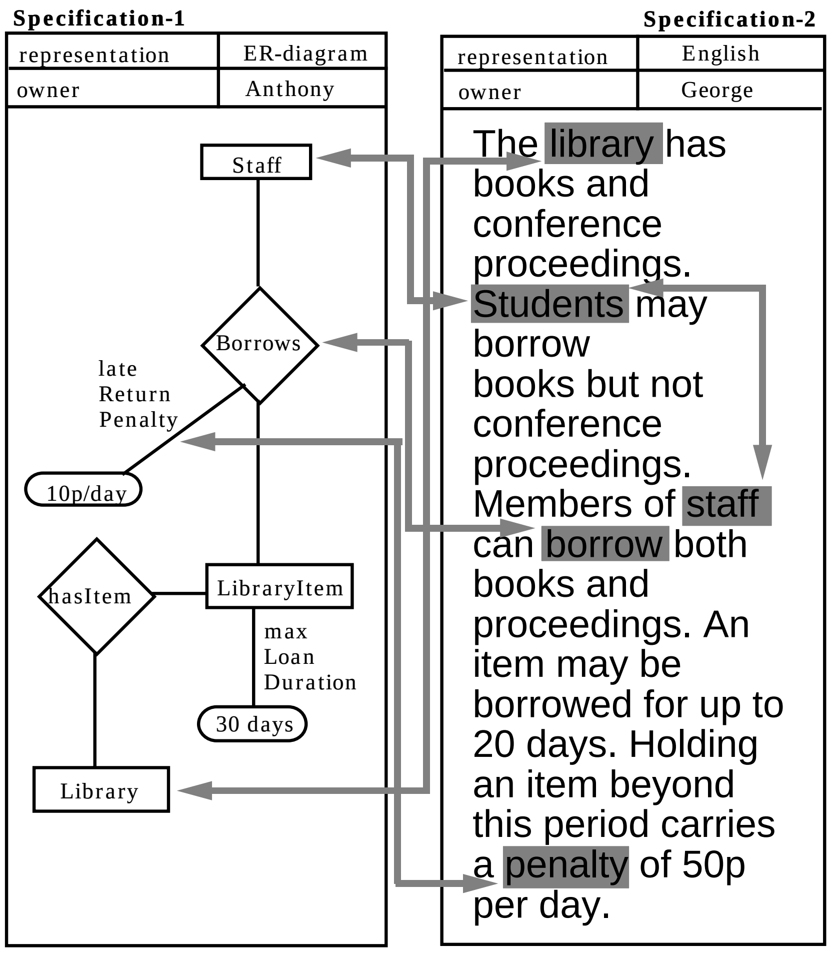
Although overlap between specifications – that is the incorporation of elements which designate common aspects of the system of concern – is a precondition for specification inconsistency, it has only been a side concern in requirements engineering research. This paper is concerned with overlaps. It defines overlap relations in terms of specification interpretations, identifies properties of these relations which are derived from the proposed definition, shows how overlaps may affect the detection of inconsistency; shows how specifications could be rewritten to reflect overlap relations and still be amenable to consistency checking using theorem proving; analyses various methods that have been proposed for identifying overlaps with respect to the proposed definition; and outlines directions for future research.

1. Motivation

In software engineering settings where different stakeholders may have conflicting requirements for the system to be built, there is no point in worrying about the consistency of the resulting specifications unless you are pretty confident that they refer to the same things within a shared domain of discourse. In other words, unless there is an *overlap* between these specifications.

Specification components overlap if they designate common aspects of the system of concern. A *consistency rule* is a condition that the two specifications must jointly satisfy. A breach of such a rule manifests itself as a logical inconsistency, that is the assertion of a fact and its negation. Overlap and inconsistency are two different levels of *interference* between specifications, the first of which is precondition for the second (Finkelstein et al [1996]).

Consider as a simple example the specifications in Figure 1, which describe loans from a university library. These specifications have been expressed in different languages (Specification-1 in an ER-modeling notation and Specification-2 in plain English) and constructed by different agents (Anthony and George). The component *Staff* of Specification-1 overlaps with the component *Student* of Specification-2 if there are members of the university staff who are also students. Given this overlap between *Student* and *Staff,* the specifications in Figure 1 would be inconsistent with respect to a consistency rule requiring that the amount of money that any library borrower should pay for late returns to the library must be the same according to both specifications. The inconsistency in this case occurs because Specification-1 and Specification-2 introduce different late return penalties for students and staff members (this inconsistency is formally derived in section 5). Note that, it would be meaningless to check consistency with respect to this rule if the specifications in Figure 1 were describing loans from libraries of different universities and therefore there was no overlap between the penalties indicated (these penalties are incurred due to late returns to different libraries). Also, checking the same consistency rule would reveal no inconsistency if the specifications were concerned with the same university library but no student could be a member of the staff at that university. In this case *Student* and *Staff* would not overlap.



Specification-1
representation
ER-diagram
owner
Anthony
Staff
Borrows
late Return Penalty
10p/day
hasItem
LibraryItem
max Loan Duration
30 days
Library
Specification-2
representation
English
owner
George
The library has books and conference proceedings. Students may borrow books but not conference proceedings. Members of staff can borrow both books and proceedings. An item may be borrowed for up to 20 days. Holding an item beyond this period carries a penalty of 50p per day.

Figure 1: Interfering specifications

Evidently from this example, overlaps have to be identified and made explicit before checking for inconsistencies. Also reporting on the consistency status of specifications must be made with reference to a specific set of overlap relations. These relations need to be represented in forms suitable for supporting consistency checking. They also need to be maintained after consistency checking since they indicate the potential for inconsistency when changes to specification elements violate previously satisfied rules or additional consistency rules are identified.

The objective of this paper is to distinguish clearly between the concepts of overlap and inconsistency, provide a formal semantics for overlap, clarify the relationship between the two concepts, and thus establish a single framework for understanding and dealing with both these phenomena. To do that it uses the formal framework of first-order logic. This is because the notion of inconsistency is well understood and defined within this formal framework. However, the use of first-order logic does not entail any claim about its practicality as a representation scheme for specifying requirements nor its necessity for deriving inconsistencies between requirements specifications.

The rest of the paper is structured as follows. Section 2 proposes a formal semantics for overlap relations, identifies properties of overlap relations which can be derived from the definition proposed and shows how these properties may be used to check the consistency of relations asserted by different agents. Section 3 shows how specifications could be rewritten to reflect overlap relations. Section 4 establishes the relationship between overlap and inconsistency. Section 5 gives examples of checking the consistency of specifications with respect to specific overlap relations and consistency rules using theorem proving. Section 6 analyses various methods that have been proposed for identifying overlaps with respect to the semantic framework proposed and section 7 outlines directions for future research. Conclusions are provided in section 8.

2. A semantics for overlap relations

We define overlaps as relations between interpretations ascribed to specifications by specific *agents*. An agent is an active entity – generally a person – but sometimes a computational mechanism. An interpretation reflects how an agent understands a specification. Agents may ascribe interpretations alone or jointly. In the latter case, the joint interpretation reflects the common understanding of the specification by the group of agents involved.

In the discussion which follows we assume specifications expressed in predicate logic. For specifications which are originally expressed in another language, we assume that they can be transformed into predicate logic by some rewriting system. Formally an interpretation is defined as follows:

Definition 1: *The interpretation of a first-order theory T, whose vocabulary W is partitioned into two sets of constant and predicate symbols C and P, is a pair (I,D) where:*

- *D is a non empty set of individuals, called the domain of the interpretation;*
- *I is a total morphism which maps:*
  - *each constant symbol in C onto an element of D*
  - *each predicate symbol S of degree n in P onto a relation $R \subseteq D^n$ (R is called the extension of S)*

Note that in this definition the vocabulary W of a first-order theory is assumed to include no function symbols since functions do not add to the expressive power of the framework.

In the field of requirements engineering the domain of an interpretation is often referred to as the universe of discourse (UoD). Given the previous definition of an interpretation, overlaps between specifications are defined as follows:

Definition 2: *Assume a pair of symbols x_i and x_j in the vocabularies W_i and W_j of two specifications S_i and S_j and two interpretations $T_{iA}=(I_{iA}, D_{iA})$ and $T_{jB}=(I_{jB}, D_{jB})$ of S_i and S_j ascribed to them by two sets of agents A and B, respectively.*

- *x_i and x_j will be said not to overlap at all with respect to T_{iA} and T_{jB} if $I_{iA}(x_i) \neq Ø$, $I_{jB}(x_j) \neq Ø$, and $I_{iA}(x_i) \cap I_{jB}(x_j) = Ø$*[1]
- *x_i and x_j will be said to overlap totally with respect to T_{ia} and T_{jB} if $I_{iA}(x_i) \neq Ø$, $I_{jB}(x_j) \neq Ø$, and $I_{iA}(x_i)=I_{jB}(x_j)$*
- *x_i will be said to overlap inclusively with x_j with respect to T_{iA} and T_{jB} if x_i and x_j are predicate symbols of degree k, $I_{iA}(x_i) \neq Ø$, $I_{jB}(x_j) \neq Ø$, and $I_{jB}(x_j) \subset I_{iA}(x_i)$*
- *x_i will be said to overlap partially with x_j with respect to T_{iA} and T_{jB} if x_i and x_j are predicate symbols of degree k, $I_{iA}(x_i) \neq Ø$, $I_{jB}(x_j) \neq Ø$, $I_{iA}(x_i) \cap I_{jB}(x_j) \neq Ø$, $I_{iA}(x_i) – I_{jB}(x_j) \neq Ø$, and $I_{jB}(x_j) – I_{iA}(x_i) \neq Ø$*

According to this definition:

- Overlap relations are defined in terms of interpretations rather than models. This is a deliberate choice made due to the fact that at the early stages of requirements engineering when specification overlaps need to be explored it is probable that specifications may have no models (i.e., they may consist of non satisfiable formulas).

- Overlap relations may only be asserted between specification symbols that have the same degree. This is not the finest level of granularity at which overlaps could be established. We could for instance have overlaps between projections of the extensions of the predicates. For instance, the extension of the predicate *person(x)* inclusively overlaps with the projection of the pairs (x,y) in the extension of the predicate *marriage(x,y)* over any of their elements. However, asserting overlaps at this level of granularity would dramatically increase the number of the relations that should be identified and therefore the scalability of the approach (given the restrictions of definition 2, N*M overlap relations need to be established between two specifications that have N and M symbols respectively).

- Overlap relations may only be asserted between specification symbols that have non-empty interpretations. An empty interpretation for a symbol could indicate either that there are no elements in the domain for which the property represented by the symbol becomes true or that the agent is not able to ascribe an interpretation to the symbol. In both these cases it would be unreasonable to try to establish an overlap relation.

The four types of overlap relations distinguished by definition 2, i.e. null, partial, inclusive and total overlaps, cover all the possible relations between non empty interpretations of specification elements exhaustively.

Our definition accommodates overlaps established with respect to:

- An interpretation assigned to a single specification by a single group of agents (Figure 2, Case 1) – for instance according to the interpretation assumed by George "student" overlaps partially with "staff" in Specification 2 (Figure 1).

[1] The symbols $\cap$, $\cup$ and - denote the set intersection, union and difference operations respectively throughout the paper.

- Interpretations assigned to a single specification by different groups of agents (Figure 2, Case 2) – for instance according to the interpretations assumed by George for "student" and Anthony for "staff" in Specification 2 (Figure 1), "student" overlaps partially with "staff".

- Interpretations assigned to different specifications by the same group of agents (Figure 2, Case 3) – for instance according to the interpretation assumed by George "Student" in Specification 2 (Figure 1) overlaps partially with "Staff" in Specification 1 (Figure 1).

- Interpretations assigned to different specifications by different groups of agents (Figure 2, Case 4) – for instance according to the interpretations assumed by George for "borrow" in Specification 2 (Figure 1) and Anthony for the relationship type "Borrows" in Specification 1 (Figure 1), "borrow" overlaps totally with "borrows".

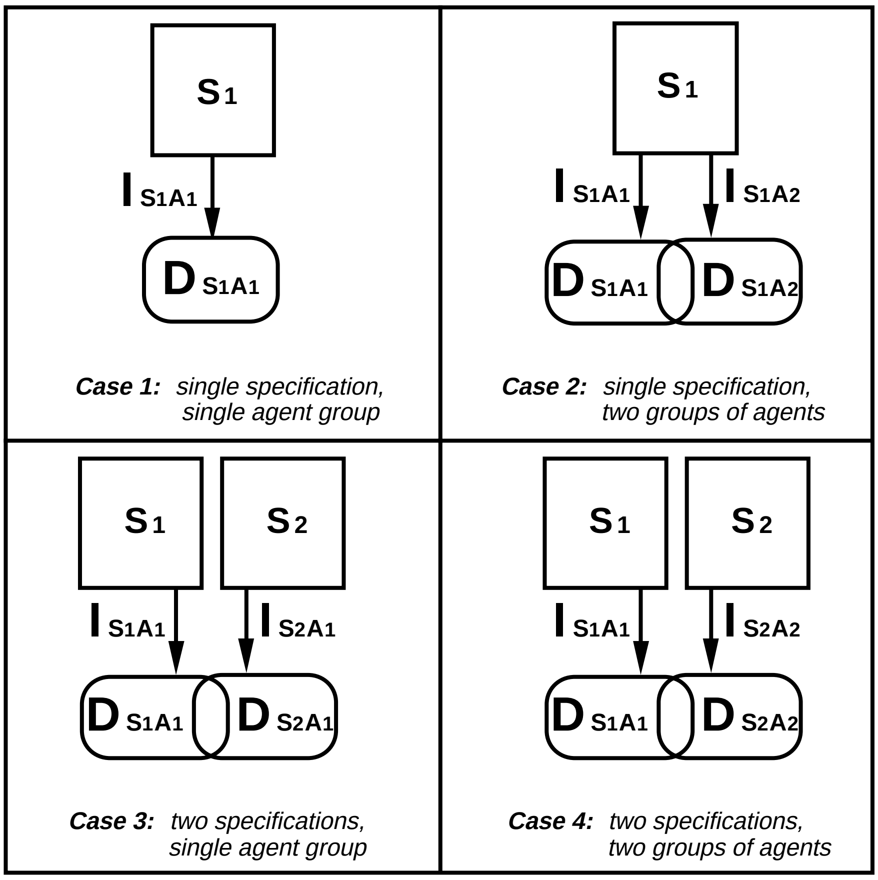

S1
I S1A1
D S1A1
Case 1: single specification, single agent group
S1
I S1A1
I S1A2
D S1A1
D S1A2
Case 2: single specification, two groups of agents
S1
S2
I S1A1
I S2A1
D S1A1
D S2A1
Case 3: two specifications, single agent group
S1
S2
I S1A1
I S2A2
D S1A1
D S2A2
Case 4: two specifications, two groups of agents

Figure 2: Different settings of overlap exploration

Note that in reality it is impossible to enumerate the interpretations of specifications with sufficient completeness for them to be used to identify overlaps. This is because it is impossible to enumerate the domains of interpretations, which in most cases are of unbounded size or change frequently. Consider for example the domains of the specifications in Figure 1. They change as new students and members of staff join the university; new items are purchased for the library; existing students and members of staff leave the university; and old library items are withdrawn or become obsolete. To overcome this problem interpretations have to be associated with the agents who ascribe them. Agents can assert overlap relations without having to define the interpretations underlying these relations. Thus, if an agent A asserts that predicate P overlaps inclusively with predicate Q, this agent essentially confirms that $I_A(Q) \subset I_A(P)$ without having to define $I_A(Q)$ or $I_A(P)$. In essence, the identity of the agent in our framework is a substitute for a precise account of the interpretation underlying the relation.

Our notion of agent, as the source of interpretations underlying overlap relations, includes people and computational mechanisms for identifying overlap relations. A human agent might be the "owner of a specification" or some third party. A computational agent might be an automatic scheme for identifying overlaps such as the unification algorithms used in theorem proving (see section 3 below). In section 6 we review how the identification of overlaps is treated in the literature.

Since agents substitute for interpretations, overlaps can be represented as relations between the symbols in the vocabularies W_i and W_j of two specifications S_i and S_j, the powerset of overlap identifying agents (2^A), and the set of overlap types (O_T):

$$OV \subset W_i \times W_j \times 2^A \times 2^A \times O_T$$ [2]

An association $ov(c_i, c_j, \{a1\}, \{a2\}, t)$ means that there is an overlap of type t between the symbols c_i and c_j in the vocabularies of S_i and S_j given the interpretation of S_i that is assumed by a1 and the interpretation of S_j that is assumed by a2. The set of overlap types O_T includes the following elements: no (null overlap); to (total overlap); po (partial overlap); $\subset$ (inclusive overlap – c_i overlaps inclusively with c_j); and $\supset$ (inclusive overlap – c_j overlaps inclusively with c_i).

Obviously different (sets of) agents may assert different overlap relations between the same specifications. A set of overlap associations between the symbols in the vocabularies of two specifications S_i and S_j, which have been asserted by the sets of agents a1 and a2, $O_{a1a2}(S_i, S_j)$, may be seen as the viewpoint of the agents in a1 and a2 about the overlap between S_i and S_j.

2.2 Properties of overlap

To ensure the coherence of overlap relations, which are asserted by a group of agents, we make the following axiomatic assumption about the interpretations that may be ascribed to any specification by a specific agent:

> **Axiom:** *An agent (set of agents) may only have one interpretation (joint interpretation) of the same specification.*

[2] The type of the overlap acts as a qualifier for each of the elements of OV and enables the distinction of subsets of OV by the type of the overlap associations they represent. This is necessary since the relation OV includes all the overlaps that have been asserted for the components of two specifications by two sets of agents regardless of their type.

In other words, we assume that an agent does not assume different interpretations for a specification when it explores the overlap relations between the symbols of this specification and symbols of any other specification. Below we present a set of properties, which hold for overlap relations of different types given the previous axiom and definition 2. Proofs of these properties are provided in the appendix. In the formulas which formally express these properties, x_i are variables whose values are items in specification vocabularies (W_i), t_i are variables whose values are overlap types (O_T), and a_i are variables whose values are sets of agents (i.e., elements of the powerset of agents 2^A).

Property 1: Overlap types are mutually exclusive. Formally, this property is expressed as follows:

$$(\forall x_1: W_1)\ (\forall x_2: W_2)\ (\forall t_1: O_T)\ (\forall t_2: O_T)\ (\forall a_1: 2^A)(\forall a_2: 2^A)\ (ov(x_1,x_2, a_1, a_2,t_1) \wedge ov(x_1,x_2, a_1, a_2,t_2) \rightarrow (t_1 = t_2))$$

Property 2: Total overlap is a reflexive relation between the symbols in the vocabulary of the same specification given the interpretation of this specification by the same group of agents. Formally, this property is expressed as follows:

$$(\forall x_1: W_1)\ (\forall a_1: 2^A)\ (ov(x_1, x_1, a_1, a_1,to))$$

Property 3: Total overlap is a symmetric relation. Formally, this property is expressed as follows:

$$(\forall x_1: W_1)\ (\forall x_2: W_2)\ (\forall a_1: 2^A)\ (\forall a_2: 2^A)\ (ov(x_1, x_2, a_1, a_2,to) \rightarrow ov(x_2, x_1, a_2, a_1,to))$$

Property 4: Total overlap is a transitive relation. Formally, this property is expressed as follows:

$$(\forall x_1: W_1)\ (\forall x_2: W_2)\ (\forall x_3:W_3)\ (\forall a_1: 2^A)\ (\forall a_2: 2^A)\ (\forall a_3: 2^A)\ (ov(x_1, x_2, a_1, a_2,to) \wedge ov(x_2, x_3, a_2, a_3,to) \rightarrow (ov(x_1, x_3, a_1, a_3,to))$$

Property 5: Null overlap is an irreflexive relation for symbols in the vocabulary of the same specification given the interpretation of this specification by the same group of agents. Formally, this property is expressed as follows:

$$(\forall x_1: W_1)\ (\forall a_1: 2^A)\ (\neg\ ov(x_1, x_1, a_1, a_1,no))$$

Property 6: Null overlap is a symmetric relation. Formally, this property is expressed as follows:

$$(\forall x_1: W_1)\ (\forall x_2: W_2)\ (\forall a_1: 2^A)\ (\forall a_2: 2^A)\ (ov(x_1, x_2, a_1, a_2,no) \rightarrow ov(x_2,x_1, a_2, a_1,no))$$

Property 7: Null overlap is a pseudo-transitive relation with respect to total and inclusive overlaps. Formally, this property is expressed as follows:

$$(\forall x_1: W_1)\ (\forall x_2: W_2)\ (\forall x_3: W_3)\ (\forall a_1: 2^A)\ (\forall a_2: 2^A)\ (\forall a_3: 2^A)\ (ov(x_1, x_2, a_1, a_2,no) \wedge ov(x_2,x_3, a_2, a_3,to) \rightarrow ov(x_1,x_3, a_1, a_3,no))$$

$$(\forall x_1: W_1)\ (\forall x_2: W_2)\ (\forall x_3: W_3)\ (\forall a_1: 2^A)\ (\forall a_2: 2^A)\ (\forall a_3: 2^A)\ (ov(x_1, x_2, a_1, a_2,no) \wedge ov(x_2,x_3, a_2, a_3,\subset) \rightarrow ov(x_1,x_3, a_1, a_3,no))$$

Property 8: Inclusive overlap is an irreflexive relation for symbols in the vocabulary of the same specification given the interpretation of this specification by the same group of agents. Formally, this property is expressed as follows:

$$(\forall x_1: W_1)\ (\forall a_1: 2^A)\ (\neg\ ov(x_1, x_1, a_1, a_1,\subset))$$

Property 9: Inclusive overlap is an anti-symmetric relation. Formally, this property is expressed as follows:

$$(\forall x_1: W_1)\ (\forall x_2: W_2)\ (\forall a_1: 2^A)\ (\forall a_2: 2^A)\ (ov(x_1, x_2, a_1, a_2,\subset) \rightarrow \neg\ ov(x_2, x_1, a_2, a_1,\subset))$$

Property 10: Inclusive overlap is a transitive relation. Formally, this property is expressed as follows:

$$(\forall x_1: W_1)\ (\forall x_2: W_2)\ (\forall x_3: W_3)\ (\forall a_1: 2^A)\ (\forall a_2: 2^A)\ (\forall a_3: 2^A)\ (ov(x_2, x_1, a_2, a_1,\subset) \wedge ov(x_3, x_2, a_3, a_2,\subset) \rightarrow ov(x_3, x_1, a_3, a_1,\subset))$$

Property 11: Inclusive overlap is a pseudo-transitive relation with respect to total overlaps. Formally, this property is expressed as follows:

$$(\forall x_1: W_1)\ (\forall x_2: W_2)\ (\forall x_3: W_3)\ (\forall a_1: 2^A)\ (\forall a_2: 2^A)\ (\forall a_3: 2^A)\ (ov(x_2, x_1, a_2, a_1,\subset) \wedge ov(x_3, x_2, a_3, a_2,to) \rightarrow ov(x_3,x_1, a_3, a_1,\subset))$$

Property 12: Inclusive overlap is a pseudo-symmetric relation. Formally, this property is expressed as follows:

$$(\forall x_1: W_1)\ (\forall x_2: W_2)\ (\forall a_1: 2^A)\ (\forall a_2: 2^A)\ (ov(x_1, x_2, a_1, a_2,\subset) \leftrightarrow ov(x_2,x_1, a_2, a_1,\supset))$$

Property 13: Partial overlap is an irreflexive relation for symbols in the vocabulary of the same specification given the interpretation of this specification by the same group of agents. Formally, this property is expressed as follows:

$$(\forall x_1: W_1)\ (\forall a_1: 2^A)\ (\neg\ ov(x_1, x_1, a_1, a_1,po))$$

Property 14: Partial overlap is a symmetric relation. Formally, this property is expressed as follows:

$$(\forall x_1: W_1)\ (\forall x_2: W_2)\ (\forall a_1: 2^A)\ (\forall a_2: 2^A)\ (ov(x_1, x_2, a_1, a_2,po) \rightarrow ov(x_2, x_1, a_2, a_1,po))$$

Property 15: Partial overlap is a pseudo-transitive relation with respect to total overlaps. Formally, this property is expressed as follows:

$$(\forall x_1: W_1)\ (\forall x_2:W_2)\ (\forall x_3: W_3)\ (\forall a_1: 2^A)\ (\forall a_2: 2^A)\ (\forall a_3: 2^A)\ (ov(x_1, x_2, a_1, a_2,po) \wedge ov(x_2, x_3, a_2, a_3,to) \rightarrow ov(x_1, x_3, a_1, a_3,po))$$

2.3 Consistency of overlap relations

Often, our assumption that an agent (or a set of agents) does not assume different interpretations for a specification when it explores the overlap relations between the symbols of this specification and symbols of any other specification turns out to be false. This happens more often if the agent involved is a human being. It is possible to detect such cases by checking the consistency of a set of asserted overlap relations with respect to the properties set out above.

For instance, we can establish that the following overlap relations asserted between the symbols of the specifications in Figure 1,

$$ov(LibraryItem,book,\{a1\},\{a2\},\subset),$$
$$ov(LibraryItem,item,\{a1\},\{a3\},\supset),$$
$$ov(book,item,\{a2\},\{a3\},no)$$

are contradictory if,

a) the interpretations assumed for LibraryItem by the agent a1 and which gave rise to ov(LibraryItem,book,{a1},{a2},$\subset$) and ov(LibraryItem,item,{a1},{a3},$\supset$) are the same;

b) the interpretations assumed for book by the agent a2 and which gave rise to ov(LibraryItem,book,{a1},{a2},$\subset$) and ov(book,item,{a2},{a3},no) are the same; and

c) the interpretations assumed for item by the agent a3 that gave rise to ov(LibraryItem,item,{a1},{a3},$\supset$) and ov(book,item,{a2},{a3},no) are the same.

This is because the first two of these relations imply that ov(item,book,{a3},{a2},$\subset$) (by transitivity of inclusive overlap – see property 10 in section 2.2), which contradicts the third (since overlap types are mutually exclusive – see property 1 in section 2.2). This contradiction indicates that the agents did not assume the same interpretations of symbols across all the comparisons.

Furthermore, the properties set out in section 2.2 may be used to infer additional overlap relations from given ones. For instance, from ov(item,book,{a1},{a2},$\subset$) and ov(item,LibraryItem,{a1},{a1},to) it may be deduced (by pseudo-transitivity of inclusive overlaps) that, ov(LibraryItem,book,{a1},{a2},$\subset$).

3. Representation of overlap relations in first-order logic

As we argued in section 1, the consistency status of specifications needs to be checked and established in reference to specific overlap relations and this status may change given different assumptions about overlaps (made by different agents).

In the case where specifications are expressed in first-order logic and their consistency is checked using theorem proving, the ability to derive an inconsistency with respect to specific overlap relations depends on an explicit representation of these relations. This is because theorem provers assume that only identical constants and predicates with the same name are totally overlapping and therefore can be mated and resolved in constructing proofs (Ramsay [1988]). Thus, the specifications S1 = {p(a)} and S2 = {$\neg$q(a)}, which are inconsistent if the predicates p and q are known to overlap totally, would not be found to be so by theorem proving since p(a) cannot be mated and resolved with $\neg$q(a). However, if the total overlap relation between p and q, ov(p,q,{a},{a},to), is explicitly represented by the formula $(\forall x)\ (p(x) \leftrightarrow q(x))$ (since $I_{S1a}(p) = I_{S2a}(q)$) then it is possible to construct a refutation from the set of formulas $S1 \cup S2 \cup \{(\forall x)\ (p(x) \leftrightarrow q(x))\}$ and thus show that they are not satisfiable.

Figures 3-5 show algorithms which define a transformation F that transforms two specifications (expressed in first-order logic) in a way which explicitly represents the overlap relations between their predicate and constant symbols. Thus, F makes it possible to take into account these relations when checking their consistency by theorem proving.

```
Algorithm:  F
Input:  $S_i$,$S_j$,$O_{a1a2}$ ($S_i$,$S_i$), $O_{a1a2}$ ($S_j$,$S_j$), $O_{a1a2}$ ($S_i$,$S_j$)
Output: S
begin
        let $C_i$ and $C_j$ be two sets consisting of the constant symbols of $S_i$ and $S_j$ respectively;
        let $P_i$ and $P_j$ be two sets consisting of the predicate symbols of $S_i$ and $S_j$ respectively;
        intra-overlap-f($S_i$, $C_i$, $P_j$, $O_{a1a2}$ ($S_i$,$S_i$), $NS_i$);
        intra-overlap-f($S_j$, $C_j$, $P_j$, $O_{a1a2}$ ($S_j$,$S_j$), $NS_j$);
        inter-overlap-f($NS_i$, $C_i$, $P_i$, $NS_j$, $C_j$, $P_j$, $O_{a1a2}$ ($S_i$,$S_j$), S);
        return (S);
end
```

Figure 3: The algorithm for transformation F

```
Algorithm:      intra-overlap-f
Input:          S, C, P, $O_{a1a2}$ (S,S)
Output:         NS
comment:
• S: a set consisting of the formulas which constitute
  the specification to be transformed
• C: a set consisting of the constant symbols  of S
• P: a set consisting of the  predicate symbols of S
• $O_{a1a2}$ (S,S): a set consisting of the  overlap relations
  between the symbols of S
• NS: a set consisting of the formulas resulting from
  the transformation of S

begin
NS = {};
for all ov(p,q,a1,a2,to) $\in$ $O_{a1a2}$ (S,S) do
  if ((p, q $\in$ C) and (p $\neq$ q)) then
    NS = NS $\cup$ {(p = q)}
  else if ((p, q $\in$ P) and (p $\neq$ q)) then
    NS = NS $\cup$  {($\forall x_1$)($\forall x_2$)...($\forall x_n$) (p($x_1$,$x_2$,...,$x_n$)
        $\leftrightarrow$ q($x_1$,$x_2$,...,$x_n$))};
  end if;
for all ov(p,q,a1,a2,no) $\in$ $O_{a1a2}$ (S,S) do
  if ((p, q $\in$ C) and (p $\neq$ q)) then
    NS = NS $\cup$ { $\neg$ (p = q)};
  else if ((p, q $\in$ P) and (p $\neq$ q)) then
    NS = NS $\cup$ {($\forall x_1$) ($\forall x_2$)...($\forall x_n$)( $\neg$
        (p($x_1$,$x_2$,...,$x_n$) $\wedge$ q($x_1$,$x_2$,...,$x_n$)))};
  end if;
end for;
for all  ov(p,q,a1,a2,$\subset$) $\in$ $O_{a1a2}$ (S,S) do
  if ((p, q $\in$ P) and (p $\neq$ q)) then
    NS = NS $\cup$ {($\forall x_1$) ($\forall x_2$)...($\forall x_n$) (q($x_1$,$x_2$,...,$x_n$)
        $\rightarrow$ p($x_1$,$x_2$,...,$x_n$))}
  end if;
end for;
for all  ov(p,q,a1,a2,po) $\in$ $O_{a1a2}$ (S,S) do
  if ((p, q $\in$ P) and (p $\neq$ q)) then
    NS = NS $\cup$ {($\exists x_1$) ($\exists x_2$)...($\exists x_n$) (p($x_1$,$x_2$,...,$x_n$) $\wedge$
        q($x_1$,$x_2$,...,$x_n$))};
    NS = NS $\cup$ {($\exists x_1$) ($\exists x_2$)...($\exists x_n$) (p($x_1$,$x_2$,...,$x_n$) $\wedge$
        $\neg$ q($x_1$,$x_2$,...,$x_n$))};
    NS = NS $\cup$ {($\exists x_1$) ($\exists x_2$)...($\exists x_n$) ($\neg$p($x_1$,$x_2$,...,$x_n$) $\wedge$
        q($x_1$,$x_2$,...,$x_n$))};
  end if ;
end for;
NS = NS $\cup$ S;
end
```

Figure 4: The algorithm for intra-overlap transformation

According to F:

- All the symbols in the vocabularies of the specifications are prefixed by the identifier of the specification they come from. For instance, every occurrence of a predicate symbol p(x) of specification S_1 is rewritten as $S_1p(x)$. Thus, the identical symbols of S_1 and S_2 are not necessarily considered as totally overlapping.

- Totally overlapping constants are represented by asserting an equality relation between them (unless they are identical and belong to the same specification in which case no action is necessary). For instance a total overlap relation *ov(c1,c2,{a1},{a2},to)* between two constant symbols c1 and c2 is asserted by:
  $S_1c1 = S_2c2$ (since I(c1) = I(c2))

- Totally overlapping predicates are represented by a universally quantified equivalence between them. For instance a total overlap relation *ov(p,q,{a1},{a2},to)* between two predicates p(x) and q(x) is asserted by:
  $(\forall x)\ (S_1p(x) \leftrightarrow S_2q(x))$ (since I(p) = I(q))

- Non overlapping constants are represented by asserting that they are not equal. For instance a null overlap relation *ov(c1,c2,{a1},{a2},no)* between two constant symbols c1 and c2 is asserted by: $\neg(S_1c1 = S_2c2)$ (since $I(c1) \cap I(c2) = \emptyset$)
  In this way, F distinguishes between non identical constants for which a null overlap relation has been asserted and non identical constants for which no null overlap relation has been asserted. The latter case is treated as one in which the overlap between the constants involved is uncertain.

- Non overlapping predicates are represented by an assertion that that there is no tuple of individuals for which both of them become true. For instance a null overlap relation *ov(p,q,{a1},{a2},no)* between two predicates p(x) and q(x) is asserted by:
  $(\forall x)\ (\neg(S_1p(x) \wedge S_2q(x)))$ (since $I(p) \cap I(q) = \emptyset$)

- Predicates which overlap inclusively are represented by a universally quantified implication between them. For instance an inclusive overlap relation *ov(p,q,{a1},{a2},$\subset$)*between two predicates p(x) and q(x) is asserted by:
  $(\forall x)\ (S1q(x) \rightarrow S_2p(x))$ (since $I(q) \subset I(p)$)

- Partially overlapping predicates are represented by assertions expressing that there will be tuples of individuals for which both predicates are true and tuples for which only one of them is true and the other is false. For instance a partial overlap relation *ov(p,q,{a1},{a2},po)* between two predicates p(x) and q(x) is asserted by:
  $(\exists x)\ (S_1p(x) \wedge S_2q(x))$ (since $I(p) \cap I(q) \neq \emptyset$)
  $(\exists x)\ (S_1p(x) \wedge \neg\ S_2q(x))$ (since $I(p) - I(q) \neq \emptyset$)
  $(\exists x)\ (\neg S_1p(x) \wedge S_2q(x))$ (since $I(q) - I(p) \neq \emptyset$)

The general approach of F is to assert formulas which express the semantics of overlap relations as defined by definition 2 in first-order logic rather than re-writing symbols in a way that would have the same effect during unification (for instance totally overlapping non identical predicate symbols could have been re-written as the same symbol). This approach makes it possible to identify the overlap relations which have been involved in deriving an inconsistency between specifications as one of the reasons for this inconsistency (see section 5 below). The assertion of equalities between the totally overlapping constant symbols make necessary the extension of the axioms of first-order logic with axioms to handle equalities (see definition 3 below).

As an example of how F works assume the following partial representations of the specifications of Figure 1 in predicate logic: $S_1 = \{(\forall x_1)\ (staff(x_1) \rightarrow penalty(x_1,10p)\ \}$ and $S_2 = \{(\forall x_1)\ (student(x_1) \rightarrow penalty(x_1,50p)\ \}$.

$O_{AA}(S_1,S_1)$	**staff**	**penalty**	**10p**
staff	to		
penalty		to	
10p			to

Table 1: Intra–S_1 overlaps given by an agent A

$O_{AA}(S_2,S_2)$	**student**	**penalty**	**50p**
student	to		
penalty		to	
50p			to

Table 2: Intra–S_2 overlaps given by an agent A

$O_{AA}(S_1,S_2)$	**student**	**penalty**	**50p**
staff	*po*		
penalty		to	
10p			no

Table 3: Overlaps between S_1 and S_2 given by an agent A

Given the overlap relations between the symbols of S_1 and S_2 shown in Tables 1-3 (the overlap types shown in italics denote those overlap relations that cannot be detected by standard unification), F produces the following formulas:

$(\forall x)\ (S_1staff(x) \rightarrow S_1penalty(x, S_110p))$

$(\forall x)\ (S_2student(x) \rightarrow S_2penalty(x, S_250p))$

$(\exists x)\ (S_1staff(x) \wedge S_2student(x))$

$(\exists x)\ (S_1staff(x) \wedge \neg\ S_2student(x))$

$(\exists x)\ (\neg S_1staff(x) \wedge S_2student(x))$

$(\forall x_1)\ (\forall x_2)\ (S_1penalty(x_1,x_2) \leftrightarrow S_2penalty(x_1,x_2))$

$\neg\ (S_110p = S_250p)$

```
Algorithm: inter-overlap-f
Input:  $NS_i, C_i, P_i, NS_j, C_j, P_j, O_{a1a2}(S_i,S_j)$
Output:  S
comment:
• $NS_i$ ($NS_j$): the set of the formulas resulting from the
  transformation of $S_i$ ($S_j$) due to its internal overlaps
• $C_i$ ($C_j$): the set of the constant symbols of $S_i$ ($S_j$)
• $P_i$ ($P_j$): the set of the predicate symbols in $S_i$ ($S_j$)
• $O_{a1a2}(S_i,S_j)$ : a set of overlap relations between the
  constant and predicate symbols of $S_i$ and $S_j$
• S: set of the formulas resulting from the
  transformation of $S_i$ and $S_j$

begin
$S = \{\}$; $T_i = NS_i$; $T_j = NS_j$;

comment: prefix the names of the symbols in the
vocabularies of Si and Sj by the identifier of the
relevant specification

for all p in $P_i$ do $NP_i[p]$ = concatenate ("$S_i$",p);
end for;
for all p in $P_j$ do $NP_j[p]$ = concatenate ("$S_j$",p);
end for;
for all c in $C_i$ do $NC_i[c]$ = concatenate ("$S_i$",c);
end for;
for all c in $C_j$ do $NC_j[c]$ = concatenate ("$S_j$",c);
end for;

for all $ov(p,q,a1,a2,to) \in O_{a1a2}(S_i,S_j)$ do
  if (($p \in P_i$) and ($q \in P_j$)) then
    $np = NP_i[p]$; $nq = NP_j[q]$;
    $S = S \cup \{(\forall x_1)(\forall x_2)...(\forall x_n)(np(x_1,x_2,...,x_n) \leftrightarrow$
            $nq(x_1,x_2,...,x_n))\}$;
  else if (($p \in C_i$) and ($q \in C_j$)) then
    $np = NC_i[p]$; $nq = NC_j[q]$;
    $S = S \cup \{(np = nq)\}$;
  end if;
end for;

for all $ov(p,q,a1,a2,no) \in O_{a1a2}(S_i,S_j)$ do
  if (($p \in P_i$) and ($q \in P_j$)) then
    $np = NP_i[p]$; $nq = NP_j[q]$;
    $S = S \cup \{(\forall x_1)(\forall x_2)...(\forall x_n)(\neg(np(x_1,x_2,...,x_n) \wedge$
          $nq(x_1,x_2,...,x_n)))\}$;
  else if (($p \in C_i$) and ($q \in C_j$)) then
    $np = NC_i[p]$; $nq = NC_j[q]$;
    $S = S \cup \{\neg(np = nq)\}$;
  end if;
end for;

for all $ov(p,q,a1,a2,\subset) \in O_{a1a2}(S_i,S_j)$ do
  if (($p \in P_i$) and ($q \in P_j$)) then
    $np = NP_i[p]$; $nq = NP_j[q]$;
    $S = S \cup \{(\forall x_1)(\forall x_2)...(\forall x_n)(nq(x_1,x_2,...,x_n) \rightarrow$
            $np(x_1,x_2,...,x_n))\}$;
  end if;
end for;

for all $ov(p,q,a1,a2,po) \in O_{a1a2}(S_i,S_j)$ do
  if (($p \in P_i$) and ($q \in P_j$)) then
    $np = NP_i[p]$; $nq = NP_j[q]$;
    $S = S \cup \{(\exists x_1)(\exists x_2)...(\exists x_n)(np(x_1,x_2,...,x_n) \wedge$
          $nq(x_1,x_2,...,x_n))\}$;
    $S = S \cup \{(\exists x_1)(\exists x_2)...(\exists x_n)(np(x_1,x_2,...,x_n) \wedge$
          $\neg nq(x_1,x_2,...,x_n))\}$;
    $S = S \cup \{(\exists x_1)(\exists x_2)...(\exists x_n)(\neg np(x_1,x_2,...,x_n) \wedge$
          $nq(x_1,x_2,...,x_n))\}$;
  end if;
end for;

comment: rewrite the predicate symbols of $S_i$ and $S_j$
for all p in $P_i$ do
    rewrite every occurrence of p in $T_i$ as $NP_i[p]$;
end for;
for all p in $P_j$ do
    rewrite every occurrence of p in $T_j$ as $NP_j[p]$;
end for;

comment: rewrite the constant symbols of $S_i$ and $S_j$
for all c in $C_i$ do
  rewrite every occurrence of c in $T_i$ as $NC_i[c]$;
end for;
for all c in $C_j$ do
  rewrite every occurrence of c in $T_j$ as $NC_j[c]$;
end for;
$S = S \cup T_i \cup T_j$;
return(S);
end
```

Figure 5: The algorithm for inter-overlap transformation

The transformation F adds up to 3*N formulas to the original specifications where N is the number of the overlap relations identified between their symbols. The formulas asserted by F may make some of the formulas in the original specifications (or literals in them) redundant. Consider for instance a specification $S_1 = \{(\forall x)(p(x) \wedge q(x) \rightarrow u(x))\}$ for which the overlap relation ov(p,q,{a},{a},$\subset$) is known to be true. F will transform S_1 into the formulas:

$\{(\forall x)(S_1p(x) \wedge S_1q(x) \rightarrow S_1u(x)), (\forall x)(S_1q(x) \rightarrow S_1p(x))\}$

Thus, the literal $S_1p(x)$ in the formula $(\forall x)\ (S_1p(x) \wedge S_1q(x) \rightarrow S_1u(x))$ becomes redundant. This suggests that F could perhaps incorporate some formula simplification mechanism or the use of a *simplification resolution strategy* (see Nilsson [1971] pp. 217-18) to reduce the computational cost of checking the consistency of specifications by theorem proving. However, it has to be realised that the net benefit from using simplification mechanisms depends on the computational cost of identifying the formulas and literals which can be eliminated and the reduction of the rate of growth of new clauses in resolution which results from this elimination. The extent of such a benefit (if it exists at all) given the particular specifications has to be identified by careful analysis before adopting a simplification approach since in any case resolution is going to generate clauses at a combinatorial growth rate.

Note that F is not meant to have a permanent effect on the contents of specifications. It is only used to rewrite specifications in a way which represents specific sets of intra- and inter-specification overlap relations that have been asserted by specific agents so as to enable their consistency checking with respect to these relations. After the consistency check, the specifications are maintained in their original form. Their consistency status with respect to different sets of overlap relations may be checked after transforming them again using F.

4. Overlap and inconsistency

As we argued in section 1, the presence of overlaps is directly related to the need to determine whether specifications satisfy certain consistency rules. The consistency status of specifications needs to be checked and established in reference to specific overlap relations. This status may change given different assumptions about overlaps made by different agents.

To account for this dependency, we propose a definition of specification inconsistency which takes into account specific consistency rules (as in Finkelstein *et. al.* [1994]) and overlap relations. Formally,

Definition 3: *Assume two specifications S_i and S_j, three sets of intra- and inter-specification overlap relations $O_{a1a2}(S_i,S_i)$, $O_{a1a2}(S_j,S_j)$ and $O_{a1a2}(S_i,S_j)$ and a consistency rule CR. S_i and S_j will be said to be inconsistent with respect to CR when overlapping as indicated by $O_{a1a2}(S_i,S_i)$, $O_{a1a2}(S_j,S_j)$ and $O_{a1a2}(S_i,S_j)$ if:*

$$\{F(S_i;S_j;\ O_{a1a2}(S_i,S_i);\ O_{a1a2}(S_j,S_j);\ O_{a1a2}(S_i,S_j)) \cup\ EA\} \models \neg CR$$

where

- F is the transformation defined in section 3
- EA is the set of the equality axioms (Shepherdson [1988]) which are needed to handle the equalities that F asserts in order to express the total overlaps between the constants of S_i and S_j:
  $(\forall x)\ x = x$
  $(\forall x_1)\ (\forall x_2)\ x_1 = x_2 \rightarrow x_2 = x_1$
  $(\forall x_1)\ (\forall x_2)\ (\forall x_3)\ x_1 = x_2 \wedge x_2 = x_3 \rightarrow x_1 = x_3$
  for each predicate symbol p: $(\forall x_1)\ (\forall y_1)\ ...\ (\forall x_n)\ (\forall y_n)\ x_1 = y_1 \wedge x_2 = y_2 \wedge ... \wedge x_n = y_n \rightarrow (p(x_1, ..., x_n) \leftrightarrow p(y_1, ..., y_n))$

Our definition is a special case of the Craig & Robinson's theorem of joint consistency. According to this theorem (see Shoenfield [1967], pg. 79) two theories T_i and T_j are inconsistent if and only if there is a formula A such that $T_i \models A$ and $T_j \models \neg A$. This is the general case where the objective is to find a formula A, which is derived from one of the specifications and satisfies the conditions of the theorem, and thus show that T_i and T_j are not satisfiable. In our case, the objective is to check whether the specifications violate a specific consistency rule by trying to prove that the negation of a rule can be derived from the specifications. In terms of the theorem above, $F(S_i;S_j;\ O_{a1a2}(S_i,S_i);\ O_{a1a2}(S_j,S_j);\ O_{a1a2}(S_i,S_j))$ plays the role of the theory T_j from which we try to prove $\neg$CR and CR plays the role of the theory T_i. If $\neg$CR can be proved from $F(S_i;S_j;\ O_{a1a2}(S_i,S_i);\ O_{a1a2}(S_j,S_j);\ O_{a1a2}(S_i,S_j))$ (i.e. $\{F(S_i;S_j;\ O_{a1a2}(S_i,S_i);\ O_{a1a2}(S_j,S_j);\ O_{a1a2}(S_i,S_j))\} \models \neg CR$) then since $CR \models CR$ by virtue of the theorem $F(S_i;S_j;\ O_{a1a2}(S_i,S_i);\ O_{a1a2}(S_j,S_j);\ O_{a1a2}(S_i,S_j))$ and CR will be inconsistent. Note that checking the satisfiability of the specifications as such with respect to specific overlap relations can also be expressed in terms of our definition. In this case, $\neg$CR would be the empty clause ($\neg CR \equiv \perp$).

We assume that the consistency rule CR is expressed in terms of the re-written (by F) predicates of S_i and S_j. This rule may represent:

- constraints which ensure that specific system features have been consistently specified in partial overlapping specifications of the system as for example the case of the penalty for late returns in section 1
- semantic properties of constructs of the specification language(s) used, which must be obeyed by S_i and S_j for them to be legitimate specifications in this language(s) – for instance the graph formulated by the generalisation relations of a class diagram expressed in the Unified Modeling Language must by acyclic (Rational [1997], pg. 31))
- constraints that must hold for specifications if they are to be compliant with software engineering practices or standards – for instance every data flow in a data-flow diagram must have an entry in the corresponding requirement dictionary (Emmerich et al [1997]).

Examples of expressing consistency rules and checking the consistency of specifications with respect to specific overlap relations between their symbols are given in the following section.

5. Examples

5.1 Example 1: Inter–specification inconsistency

As shown in section 3, the following formulas capture part of the information in the specifications in Figure 1 in a way which represents the overlaps shown in Tables 1-3:

(f1) $(\forall x)\ (S_1staff(x) \rightarrow S_1penalty(x,\ S_110p))$

(f2) $(\forall x)\ (S_2student(x) \rightarrow S_2penalty(x,\ S_250p))$

(f3) $(\exists x)\ (S_1staff(x) \wedge S_2student(x))$

(f4) $(\exists x)\ (S_1 staff(x) \wedge \neg\ S_2 student(x))$

(f5) $(\exists x)\ (\neg S_1 staff(x) \wedge S_2 student(x))$

(f6) $(\forall x_1)\ (\forall x_2)\ (S_1 penalty(x_1,x_2) \leftrightarrow S_2 penalty(x_1,x_2))$

(f7) $\neg\ (S_1 10p = S_2 50p)$

These formulas are inconsistent with respect to a rule requiring that the amount of money that any library borrower should pay for late returns to the library according to the specifications involved must be the same. This consistency rule is expressed by the formula:

$CR = \{\ (\forall x_1)\ (\forall x_2)\ (\forall x_3)\ (S_1 penalty(x_1, x_2) \wedge S_2 penalty(x_1, x_3) \rightarrow (x_2 = x_3)\}$

The inconsistency may be shown by constructing a refutation from the set of formulas $\{f1,...,f7\} \cup \{CR\} \cup EA$ using resolution as follows:

- From f3 we have that there will be an a such that $S_1 staff(a) \wedge S_2 student(a)$ or equivalently:
  (f8) $S_1 staff(a)$
  (f9) $S_2 student(a)$
- From f1 and f8 we have: (f10) $S_1 penalty(a, S_1 10p)$
- From f2 and f9 we have: (f11) $S_2 penalty(a, S_2 50p)$
- From f10 and CR we have: (f12) $\neg\ S_2 penalty(a, x_3) \vee (S_1 10p = x_3)$
- From f11 and f12 we have: (f13) $(S_1 10p = S_2 50p)$
- From f7 and f13 we have: $\perp$

Note that the partial overlap between the predicate *staff* (S1) and the predicate *student* (S2) had a special role in revealing this inconsistency. The assertion of this relation indicated that, given the interpretation of the predicates *staff* and *student* by the agent A, there will be students who are also members of the university staff as well as students who are not members of the university staff and members of staff who are not students. This was not apparent in the original specifications. Due to this overlap S1 and S2 are inconsistent with respect to the late return penalties they specify for different categories of library borrowers.

5.2 Example 2: Intra–specification inconsistency

The case of intra-specification inconsistency is covered by definition 3 assuming that $S_j = \emptyset$, $O_{a1a2}(S_j,S_j) = \emptyset$, and $O_{a1a2}(S_i,S_j) = \emptyset$. Then, the condition of the definition becomes:

$$\{F(S_i;\ \emptyset;\ O_{a1a2}(S_i,S_i);\ \emptyset;\ \emptyset) \cup EA\} \models \neg\ CR$$

Consider as an example the following part of Specification 2 in Figure 1 represented in first-order logic:

$S_2 = \{\ (\forall x_1)\ (\forall x_2)\ (student(x_1) \wedge book(x_2) \rightarrow canborrow(x_1,x_2)),$

$(\forall x_1)\ (\forall x_2)\ (student(x_1) \wedge proceedings(x_2) \rightarrow \neg\ canborrow(x_1,x_2)),$

$(\forall x_1)\ (\forall x_2)\ (staff(x_1) \wedge book(x_2) \rightarrow canborrow(x_1,x_2)),$

$(\forall x_1)\ (\forall x_2)\ (staff(x_1) \wedge proceedings(x_2) \rightarrow canborrow(x_1,x_2))$,

$(\exists x)\ (book(x))$,

$(\exists x)\ (proceedings(x))\}$

These formulas are inconsistent given the overlaps between the predicate symbols of S_2 shown in Table 4 (the overlap relations whose types appear in italics are those that cannot be detected by standard unification algorithms). The inconsistency arises due to the partial overlap between the predicates *student* and *staff*, which allows for the possibility to have students who are also members of staff. Given this overlap and the subsequent transformation of S_2 by F, it can be derived that such students can and cannot borrow proceedings, which is a contradiction.

$O_{BB}(S_2,S_2)$	**staff**	**student**	**book**	**proceedings**	**canborrow**
staff	to	*po*			
student	*po*	to			
book			to		
proceedings				to	
canborrow					to

Table 4: Intra-Specification overlaps for S2 asserted by an agent B

More specifically, the algorithm F transforms S_2 into the following formulas (note that only *the intra-overlap-f* algorithm asserts new formulas in this case since that $S_j = \emptyset$, $O_{a1a2}(S_j,S_j) = \emptyset$ and $O_{a1a2}(S_i,S_j) = \emptyset$):

(f1) $(\forall x_1)\ (\forall x_2)\ (S_2student(x_1) \wedge S_2book(x_2) \rightarrow S_2canborrow(x_1,x_2))$

(f2) $(\forall x_1)\ (\forall x_2)\ (S_2student(x_1) \wedge S_2proceedings(x_2) \rightarrow \neg\ S_2canborrow(x_1,x_2))$

(f3) $(\forall x_1)\ (\forall x_2)\ (S_2staff(x_1) \wedge S_2book(x_2) \rightarrow S_2canborrow(x_1,x_2))$

(f4) $(\forall x_1)\ (\forall x_2)\ (S_2staff(x_1) \wedge S_2proceedings(x_2) \rightarrow S_2canborrow(x_1,x_2))$

(f5) $(\exists x)\ (S_2book(x))$

(f6) $(\exists x)\ (S_2proceedings(x))$

(f7) $(\exists x)\ (S_2staff(x) \wedge S_2student(x))$

(f8) $(\exists x)\ (S_2staff(x) \wedge \neg\ S_2student(x))$

(f9) $(\exists x)\ (\neg S_2staff(x) \wedge S_2student(x))$

Then, given a consistency rule which has the truth value "true", a refutation may be constructed from $\{f1,...,f9\} \cup EA$ using resolution as follows (as it turns out in this case the equality axioms are not required):

- From f7 we have that there will be an *a* such that $S_2staff(a) \wedge S_2student(a)$ or equivalently:

  (f10) $S_2staff(a)$

  (f11) $S_2student(a)$
- From f6 we have that there will be a *b* such that: (f12) $S_2proceedings(b)$
- From f2 and f11 we have: (f13) $(\forall x_2)\ (S_2proceedings(x_2) \rightarrow \neg\ S_2canborrow(a,x_2))$

- From f12 and f13 we have: (f14) $\neg$ S_2canborrow(a,b)
- From f4 and f10 we have: (f15) $(\forall x_2)$ (S_2proceedings(x_2) $\rightarrow$ S_2canborrow(a,x_2))
- From f12 and f15 we have: (f16) S_2canborrow(a,b))
- From f14 and f16 we have: $\perp$

Note that since S_2 says nothing about the relationship between the predicates *student* and *staff* the previous inconsistency could not have been detected from the formulas in S_2. It has to be realised that this ambiguity was present in Specification-2 of Figure 1 and did not arise due to the representation of this specification in first-order logic. Such ambiguities are likely to characterise requirements (Mullery, [1996]) and cannot be totally removed by specifying them in any formal language like first-order logic. The only way to remove them is to try to identify overlap relations.

6. Related work

Despite the importance of overlap for inconsistency, its identification has only been a side concern in requirements engineering research. Overlaps are usually identified by representation conventions, shared ontologies, designations and specification owners.

The simplest and most common representation convention is to assume total overlap between elements with identical names and no overlap between any other pair of elements (Easterbrook et. al. [1994], Finkelstein et. al. [1994], Easterbrook & Nuseibeh [1996], Heitmeyer et. al. [1995], van Lamsweerde [1996], Nissen et. al. [1996]). Undoubtedly this convention is unacceptably weak for even simple forms of specification heterogeneity such as the presence of synonyms and homonyms. Overlaps may also be identified by relations embedded in specification languages. For instance, the "Is-a" relation in various object-oriented specification languages, which is usually given a set-inclusion semantics, is a statement of inclusive overlap: the subtype designates a subset of the instances of the supertype. Similarly, the implication ($\rightarrow$) between two predicates of the same arity constitutes a statement of inclusive overlap in a first order language. Note that such embedded relations may only interrelate specifications represented in the same language, and need to be identified explicitly for independently constructed specifications.

In order to extend this approach for specifications expressed in different languages it is necessary to interrelate the constructs of these languages. Fiadeiro & Maibaum [1995] interrelate such constructs using the formal framework of category theory. They formalise specifications as categories (directed graphs with a composition and identity structure) and use functors between these categories (i.e. functional mappings between the nodes and edges of the categories) to interconnect them in a consistent way (i.e. in a way which preserves their structures and therefore the properties that these structures represent). Note however that functors cannot be used to represent overlap relations since the latter are N:M mappings in the general case. Also the existence of a functor between two categories does not mean that they also overlap. Consider for instance the following two Is-a graphs:

Graph 1: object-class(Student), object-class(LibraryBorrower), Is-a(Student,LibraryBorrower)
Graph 2: object-class(Book), object-class(LibraryItem), Is-a(Book,LibraryItem)

If we assume a subset semantics for the Is-a relations then these relations will constitute partial orders and therefore the previous graphs may be considered as categories (Rydeheard & Burstall [1988]). Between these graphs there is a functor which maps:
object-class(Student) onto *object-class(Book),*
object-class(LibraryBorrower) onto *object-class(LibraryItem),* and
Is-a(Student,LibraryBorrower) onto *Is-a(Book,LibraryItem)*

This functor preserves the Is-a relations between the classes in the two graphs. Nevertheless these classes do not overlap given any plausible interpretation that may be derived from their names.

An alternative approach is to use shared ontologies for assigning interpretations to specification elements and to identify overlaps between elements that have been "tagged" with the same item in the ontology (Leite & Freeman [1991], Robinson & Fickas [1994], Boehm & In [1996]). In order to be useful – that is to facilitate communication between agents about a specific domain of discourse without necessarily assuming a global shared theory – ontologies have to include definitions of their items and be general. Furthermore, the agents need to "commit" themselves to the ontology in the sense that their observable actions are consistent with the definitions of the items (Gruber [1993], Guarino [1994]). As a consequence of ontology generality, specifications have to add a lot of details to an ontology in order to describe a system with reasonable degree of completeness. Inevitably this leads to associations of many specification elements with the same item in the ontology and as a result only coarse-grain overlaps can be identified by using these items. Furthermore since ontologies incorporate item definitions, they are models themselves and as such they might be interpreted in different ways by different specification owners! Thus the same ontology item may be used for ascribing different meanings to different specification elements. As a consequence the existence of overlaps between elements associated with the same item in an ontology cannot be assumed with safety unless there is evidence that the specification owners understood the ontology in the same way.

A third approach is to involve people in overlap identification (Easterbrook [1991], Zave & Jackson [1993], Spanoudakis & Finkelstein [1997]). In this approach, specification owners (or a third party) explore their specifications and cooperatively identify overlaps possibly with the support of some tool or method. The support available might range from visual aid for browsing, graphical selection and recording of overlaps (Easterbrook [1991]) to specification matching methods (Spanoudakis & Constantopoulos [1995]). However inspection is not sufficient when it comes to specifications of substantial complexity. Matching methods tend to be sensitive to heterogeneity in the specification representation, granularity and level of abstraction.

Finally, there is a possibility to identify overlaps by virtue of "designations" (Jackson [1997]). Designations constitute a way of associating non ground terms in formal specifications with ground terms which are known to have reliable and unambiguous interpretations (called "phenomena"). A designation associates a non ground term with a recognition rule which identifies the phenomena designated by it. The rule is specified in natural language. Designations can certainly help human agents in identifying overlap relations between specification components. However, they shouldn’t be used as definitive indications of overlaps. The reason is that the recognition rules of designations might themselves admit different interpretations. Even Jackson (1997) admits that:

" ... the recognition rule in a designation is inevitably imperfect. However carefully it may be formulated, there may always be hard cases in which we are uncertain whether or not the rule applies." (Jackson [1997], page 14).

7. Future Work

The distinction between overlap and inconsistency is crucial and, we believe, necessitates the investigation of overlap as a key component of any research whose objective is the effective management of inconsistency in requirements engineering. The particular issues which deserve special attention in our view include the identification, synthesis, representation and management of overlap relations.

7.1 Overlap identification

It should be clearly appreciated that the identification of overlap is complicated by specifications which might have been constructed independently by stakeholders with varying concerns, backgrounds and knowledge. As a consequence specifications might have been expressed in different languages; might be at different levels of abstraction, granularity and formality, and deploy different terminologies. The complexities arising from these forms of heterogeneity – set alongside the normal software engineering problems of scale – make the identification of overlap a very complex activity.

We have been working on the assumption that effective overlap identification is possible by developing specialised methods or algorithms for specifications expressed in particular specification languages. This approach has been successful in developing algorithms for specialised consistency checks (e.g. consistency checking for SCR specifications [Heitmeyer et al 1995]). In [Spanoudakis & Finkelstein 1997, Spanoudakis & Finkelstein 1998], we presented a method for identifying overlaps and inconsistencies between specifications expressed in an object-oriented framework. This method is based on an automatic similarity analysis of specifications which identifies possible overlaps between them which are then assessed and confirmed or rejected by humans. The similarity analysis performed by the method has polynomial complexity, something that makes it scaleable.

Our experience with this method indicates that the accuracy of the similarity analysis is considerably reduced when it comes to specifications which express requirements at very different levels of granularity and abstraction. This is exactly where human intervention is required. The articulation of heuristic criteria for deciding which parts of the specifications are problematic in this respect and as such should be inspected carefully by humans is the next step we wish to explore in this direction of research.

7.2 Synthesis of overlap relations

We envisage that in many settings a combination of different overlap identification techniques will be required, and overlap relations identified by different agents will need to be synthesised in an attempt to reduce the cost and complexity of overlap identification. For instance, in cases where a shared ontology has been evidently used in a coherent way for some time, it makes sense to use it for identifying coarse-grain overlaps and then refine them using some other method (e.g. specification matching or inspection). Or, in checking the consistency of more than two specifications it might be necessary to rely on pairwise overlaps between them asserted by different agents.

In cases where overlap relations have been identified by different agents but not jointly, it is necessary to be able to check if these relations are consistent, and to synthesise them in a coherent and systematic way prior to informing consistency checking. It is also necessary to be able to assess our confidence in the subsequent consistency checking. Dealing systematically with these issues requires an extension of the formal framework presented in this paper.

7.3 Overlap representation and management

Overlap relations need to be stored separately from the specifications they relate, represented in forms suitable for consistency checking, and maintained after consistency checking. This would make checking consistency with respect to different sets of overlap relations easier. It would also facilitate the role of overlap as an indicator of the potential for inconsistency when changes to specification elements violate previously satisfied rules or additional consistency rules are identified. However, representing overlaps separately from the specifications they relate introduces interesting questions about what happens to them when specifications change, especially in cases where the agents that identified these relations were not involved in making the changes.

8. Conclusions

In this paper, we have argued that there are two distinct levels of interference between requirement specifications, namely specification overlap and inconsistency, provided formal definitions for both these levels, and discussed their relationship. The paper suggests that overlap is a relation between interpretations ascribed to specifications by specific agents, inconsistency is unsatisfiability of specifications with respect to specific consistency rules and overlap relations, and the consistency status of specifications has to be checked with respect to specific overlap relations and might change if these relations change.

The consequence of this stance is that the identification of overlaps deserves special attention in research which is concerned with the detection of inconsistencies between specifications. This is because however carefully two specifications may be constructed or even formalised there will always be vagueness regarding the overlap relations between their components which must be reduced (and ideally totally removed) before being able to say anything about their consistency with certainty.

This paper makes a step towards this direction by providing a formal semantics for overlap relations and establishing a set of properties that may be used to check their consistency. Note that according to this semantics overlap relations are restricted to symbols of the same degree in the vocabularies of specifications (see Definition 2). Clearly, there might be cases where specifications would exhibit inconsistencies if overlap relations between symbols of different degrees had been taken into account. Nevertheless, we have deliberately made this restriction to avoid an increase in the computational complexity of the entire approach if such relations are taken into account.

In this paper we have also reviewed existing approaches to the identification of overlap and assessed their merit with respect to the proposed semantics. Finally, we have suggested the research directions that should be pursued in order to deal effectively with the problem of identifying overlaps between requirement specifications.

References

Boehm B., In H., 1996. "Identifying Quality Requirements Conflicts", IEEE Software, March 1996, pp. 25-35.

Easterbrook S., 1991. "Handling Conflict between Domain Descriptions with Computer-Supported Negotiation", Knowledge Acquisition, 3, pp. 255-289.

Easterbrook S., Finkelstein A., Kramer J. & Nuseibeh B., 1994. "Co-Ordinating Distributed ViewPoints: the anatomy of a consistency check", International Journal on Concurrent Engineering: Research & Applications, 2,3, CERA Institute, USA, pp. 209-222.

Easterbrook S., Nuseibeh B., 1996. "Using ViewPoints for Inconsistency Management", Software Engineering Journal, 11, 1, BCS/IEE Press, pp. 31-43.

Emmerich W., Finkelstein A., Montangero C. and Stevens R., 1997. "Standards Compliant Software Development", ICSE '97 Workshop on Living with Inconsistency (electronic publication - available by ftp from: http://www.cs.ucl.ac.uk/staff/w.emmerich/publications/)

Fiadeiro J., Maibaum T., 1995. "Interconnecting Formalisms: Supporting Modularity, Reuse and Incrementality", Proceedings of the Symposium on the Foundations of Software Engineering (FSE 95), ACM Press, pp. 1-8.

Finkelstein, A., Spanoudakis, G. & Till D., 1996. "Managing Interference", Joint Proceedings of the Sigsoft '96 Workshops – Viewpoints '96, ACM Press, pp. 172-174.

Finkelstein, A., Gabbay, D., Hunter, A., Kramer, J., & Nuseibeh, B., 1994. "Inconsistency Handling In Multi-Perspective Specifications", IEEE Transactions on Software Engineering, 20, 8, pp. 569-578.

Gruber T., 1993. "Towards Principles for the Design of Shared Ontologies Used for Knowledge Sharing", International Workshop on Formal Ontology, Padova, Italy (also available as Technical Report KSL 93-04, Knowledge Systems Laboratory, Stanford University).

Guarino N., 1994. "The Ontological Level", In Philosophy and the Cognitive Sciences, (eds) R. Casati, B. Smith and G. White, Holder-Pichler-Tempsky, Vienna.

Heitmeyer C., Labaw B., Kiskis D., 1995. "Consistency Checking of SCR-Style Requirements Specifications", Proceedings of the 2nd International Symposium on Requirements Engineering (RE '95), IEEE CS Press, pp. 56-63.

Jackson M., 1997. "The meaning of requirements", Annals of Software Engineering, Vol. 3, pp. 5-21.

Leite J.C.S.P., Freeman P.A., 1991. "Requirements Validation through Viewpoint Resolution", IEEE Transactions on Software Engineering, Vol. 12, No. 12, pp. 1253-1269.

Mullery G., 1996. "Tool Support For Multiple Viewpoints", Joint Proceedings of the Sigsoft '96 Workshops – Viewpoints '96, ACM Press, pp. 227-231.

Nilsson N., 1971. "Problem Solving Methods in Artificial Intelligence", McGraw Hill.

Nissen H., Jeusfeld M., Jarke M., Zemanek G., Huber H., 1996. "Managing Multiple Requirements Perspectives with Metamodels", IEEE Software, March 1996, pp. 37-47.

Ramsay A., 1988. "Formal Methods in Artificial Intelligence", Cambridge Tracts in Theoretical Computer Science, Cambridge University Press, ISBN 0 521 35236 3.

Rational, 1997. "UML Semantics", Rational Software Corp., 2800 San Thomas Expressway, Santa Clara, CA, USA, version 1.1, (available by ftp from: http://www.rational.com/uml/adobe.html)

Robinson W., Fickas S., 1994. "Supporting Multiple Perspective Requirements Engineering", Proceedings of the 1st International Conference on Requirements Engineering (ICRE 94), IEEE Computer Society Press, pp.206-215

Rydeheard D.E., Burstall R.M., 1988. "Computational Category Theory", Prentice Hall, International Series in Computer Science, ISBN 0-13-162736-8

Shepherdson J.C., 1988. "Negation in Logic Programming", In Foundations of Deductive Databases and Logic Programming, (ed) Minker J., Morgan Kaufmann, pp.19-89

Shoenfield J., 1967. "Mathematical Logic", Addison Wesley.

Spanoudakis G., Constantopoulos P., 1995. "Integrating Specifications: A Similarity Reasoning Approach", Automated Software Engineering Journal, 2, 4, pp. 311-342.

Spanoudakis, G., Finkelstein, A., 1997. "Reconciling Requirements: a method for managing interference, inconsistency and conflict", Annals of Software Engineering, Vol. 3, pp. 433-457

Spanoudakis, G., Finkelstein, A., 1998. "A Semi-automatic Process of Identifying Overlaps and Inconsistencies between Requirements Specifications", In Proceedings of the 5th International Conference on Object Oriented Information Systems (OOIS ‘98), Springer, ISBN 1-85233-046-5, pp. 405-426

van Lamsweerde A., 1996. "Divergent Views in Goal-Driven Requirements Engineering", Joint Proceedings of the Sigsoft ‘96 Workshops – Viewpoints ‘96, ACM Press, pp. 252-256.

Zave P., Jackson M., 1993. "Conjunction as Composition", ACM Transactions on Software Engineering and Methodology, Vol. 2, No. 4, pp. 379-411.

Appendix: Properties of overlap relations

In the formulas which formally express the properties of overlap below, x_i are variables whose values are items in specification vocabularies (W_i), t_i are variables whose values are overlap types (O_T), and a_i are variables whose values are sets of agents (elements of the powerset 2^A).

Property 1: Overlap types are mutually exclusive. Formally, this property is expressed as follows:

$$(\forall x_1: W_1)\ (\forall x_2: W_2)\ (\forall t_1: O_T)\ (\forall t_2: O_T)\ (\forall a_1: 2^A)(\forall a_2: 2^A)\ (ov(x_1, x_2, a_1, a_2,t_1) \wedge ov(x_1, x_2, a_1, a_2,t_2) \rightarrow (t_1 = t_2))$$

Proof. Assume two predicate or constant symbols s1 and s2 and two sets of agents a1 and a2. If s1 and s2 overlap totally with respect to the interpretations ascribed to them by a1 and a2(ov(s1,s2,a1,a2,to)) then by definition 2 $I_{a1}(s1) = I_{a2}(s2)$. Therefore, $I_{a1}(s1) \cap I_{a2}(s2) \neq \emptyset$, $I_{a1}(s1) \not\subset I_{a2}(s2)$, $I_{a2}(s2) \not\subset I_{a1}(s1)$. However,

- From $I_{a1}(s1) \cap I_{a2}(s2) \neq \emptyset$ we have that $\neg$ ov(s1,s2,a1,a2,no),
- From $I_{a1}(s1) \not\subset I_{a2}(s2)$ we have that $\neg$ ov(s1,s2,a1,a2,$\supset$), and
- From $I_{a2}(s2) \not\subset I_{a1}(s1)$ we have that $\neg$ ov(s1,s2,a1,a2,$\subset$).

Similar proofs may be constructed in the cases where s1 and s2 overlap partially, inclusively or do not overlap at all.

Property 2: Total overlap is a reflexive relation between the symbols in the vocabulary of the same specification given the interpretation of this specification by the same group of agents. Formally, this property is expressed as follows:

$$(\forall x_1: W_1)\ (\forall a_1: 2^A)\ (ov(x_1, x_1, a_1, a_1,to))$$

Proof. Since an agent (set of agents) *a* has a single interpretation (I_a,D_a) of a specification S and the mapping I_a is a total morphism each symbol in S will be mapped onto a single relation R ($R \subseteq D^n$) given (I_a,D_a). Therefore for each symbol x in S it will be that $I_a(x) = I_a(x)$ or equivalently, ov(x,x,a,a,to).

Property 3: Total overlap is a symmetric relation. Formally, this property is expressed as follows:

$$(\forall x_1: W_1)\ (\forall x_2: W_2)\ (\forall a_1: 2^A)\ (\forall a_2: 2^A)\ (ov(x_1, x_2, a_1, a_2,to) \rightarrow ov(x_2, x_1, a_2, a_1,to))$$

Proof. Assume two predicate or constant symbols s1 and s2 and two sets of agents a1 and a2. If it is known that ov(s1,s2,a1,a2,to) then $I_{a1}(s1) = I_{a2}(s2)$ or equivalently $I_{a2}(s2) = I_{a1}(s1)$ and by implication ov(s2,s1,a2,a1,to).

Property 4: Total overlap is a transitive relation. Formally, this property is expressed as follows:

$$(\forall x_1: W_1)\ (\forall x_2: W_2)\ (\forall x_3:W_3)\ (\forall a_1: 2^A)\ (\forall a_2: 2^A)\ (\forall a_3: 2^A)\ (ov(x_1, x_2, a_1, a_2,to) \wedge ov(x_2, x_3, a_2, a_3,to) \rightarrow (ov(x_1, x_3, a_1, a_3,to))$$

Proof. Assume three predicate symbols s1, s2 and s3 and three sets of agents a1, a2 and a3. If it is known that ov(s1,s2,a1,a2,to) and ov(s2,s3,a2,a3,to) then $I_{a1}(s1) = I_{a2}(s2)$ and $I_{a2}(s2) = I_{a3}(s3)$. Therefore $I_{a1}(s1) = I_{a3}(s3)$ and by implication ov(s1,s3,a1,a3,to).

Property 5: Null overlap is an irreflexive relation for symbols in the vocabulary of the same specification given the interpretation of this specification by the same group of agents. Formally, this property is expressed as follows:

$$(\forall x_1\text{: } W_1)\ (\forall a_1\text{: } 2^A)\ (\neg\ ov(x_1,\ x_1,a_1,\ a_1,no))$$

Proof. Since by property 2 a symbol in the vocabulary of a specification totally overlaps with itself and overlap types are mutually exclusive, a symbol cannot have a null overlap relation with itself.

Property 6: Null overlap is a symmetric relation. Formally, this property is expressed as follows:

$$(\forall x_1\text{: } W_1)\ (\forall x_2\text{: } W_2)\ (\forall a_1\text{: } 2^A)\ (\forall a_2\text{: } 2^A)\ (ov(x_1,\ x_2,\ a_1,\ a_2,no) \rightarrow ov(x_2,x_1,\ a_2,\ a_1,no))$$

Proof. Assume two predicate or constant symbols s1 and s2 and two sets of agents a1 and a2. If it is known that ov(s1,s2,a1,a2,no) then $I_{a1}(s1) \cap I_{a2}(s2) = Ø$ and by implication ov(s2,s1,a2,a1,no).

Property 7: Null overlap is a pseudo-transitive relation with respect to total and inclusive overlaps. Formally, this property is expressed as follows:

$$(\forall x_1\text{: } W_1)\ (\forall x_2\text{: } W_2)\ (\forall x_3\text{: } W_3)\ (\forall a_1\text{: } 2^A)\ (\forall a_2\text{: } 2^A)\ (\forall a_3\text{: } 2^A)\ (ov(x_1,\ x_2,\ a_1,\ a_2,no) \wedge ov(x_2,x_3,\ a_2,\ a_3,to) \rightarrow ov(x_1,x_3,\ a_1,\ a_3,no))$$

$$(\forall x_1\text{: } W_1)\ (\forall x_2\text{: } W_2)\ (\forall x_3\text{: } W_3)\ (\forall a_1\text{: } 2^A)\ (\forall a_2\text{: } 2^A)\ (\forall a3\text{: } 2^A)\ (ov(x_1,\ x_2,\ a_1,\ a_2,no) \wedge ov(x_2,x_3,\ a_2,\ a_3,\ \subset) \rightarrow ov(x_1,x_3,\ a_1,\ a_3,no))$$

Proof. Assume three predicate symbols s1, s2 and s3 and three sets of agents a1, a2 and a3.

Case 1 (total overlaps): If it is known that ov(s1,s2,a1,a2,no) and ov(s2,s3,a2,a3,to) then $I_{a1}(s1) \cap I_{a2}(s2) = Ø$ and $I_{a2}(s2) = I_{a3}(s3)$. Therefore $I_{a1}(s1) \cap I_{a3}(s3) = Ø$ and by implication ov(s1,s3,a1,a3,no).

Case 2 (inclusive overlaps): If it is known that ov(s1,s2,a1,a2,no) and ov(s2,s3,a2,a3,⊂) then $I_{a1}(s1) \cap I_{a2}(s2) = Ø$ and $I_{a2}(s2) \supset I_{a3}(s3)$. Therefore $I_{a1}(s1) \cap I_{a3}(s3) = Ø$ and by implication ov(s1,s3,a1,a3,no).

Property 8: Inclusive overlap is an irreflexive relation for symbols in the vocabulary of the same specification given the interpretation of this specification by the same group of agents. Formally, this property is expressed as follows:

$$(\forall x_1\text{: } W_1)\ (\forall a_1\text{: } 2^A)\ (\neg\ ov(x_1,\ x_1,\ a_1,\ a_1,\subset))$$

Proof. Since by property 2 a symbol in the vocabulary of a specification totally overlaps with itself and overlap types are mutually exclusive, a symbol cannot have an inclusive overlap relation with itself.

Property 9: Inclusive overlap is an anti-symmetric relation. Formally, this property is expressed as follows:

$$(\forall x_1: W_1)\ (\forall x_2: W_2)\ (\forall a_1: 2^A)\ (\forall a_2: 2^A)\ (ov(x_1, x_2, a_1, a_2,\subset) \rightarrow \neg\ ov(x_2, x_1, a_2, a_1,\subset))$$

Proof. Assume two predicate or constant symbols s1 and s2 and two sets of agents a1 and a2. If it is known that ov(s1,s2,a1,a2,$\subset$) then $I_{a1}(s1) \supset I_{a2}(s2)$. Therefore $I_{a1}(s2) \not\subset I_{a2}(s1)$ and by implication $\neg$ ov(s2,s1,a2,a1,$\subset$).

Property 10: Inclusive overlap is a transitive relation. Formally, this property is expressed as follows:

$$(\forall x_1: W_1)\ (\forall x_2: W_2)\ (\forall x_3: W_3)\ (\forall a_1: 2^A)\ (\forall a_2: 2^A)\ (\forall a_3: 2^A)\ (ov(x_2, x_1, a_2, a_1,\subset) \wedge ov(x_3, x_2, a_3, a_2,\subset) \rightarrow ov(x_3, x_1, a_3, a_1,\subset))$$

Proof. Assume three predicate symbols s1, s2 and s3 and three sets of agents a1, a2 and a3. If it is known that ov(s1,s2,a1,a2,$\supset$) and ov(s2,s3,a2,a3,$\supset$) then $I_{a1}(s1) \subset I_{a2}(s2)$ and $I_{a2}(s2) \subset I_{a3}(s3)$. Therefore $I_{a1}(s1) \subset I_{a3}(s3)$ and by implication ov(s1,s3,a1,a3,$\supset$).

Property 11: Inclusive overlap is a pseudo-transitive relation with respect to total overlaps. Formally, this property is expressed as follows:

$$(\forall x_1: W_1)\ (\forall x_2: W_2)\ (\forall x_3: W_3)\ (\forall a_1: 2^A)\ (\forall a_2: 2^A)\ (\forall a_3: 2^A)\ (ov(x_2, x_1, a_2, a_1,\subset) \wedge ov(x_3, x_2, a_3, a_2,to) \rightarrow ov(x_3, x_1, a_3, a_1,\subset))$$

Proof. Assume three predicate symbols s1, s2 and s3 and three sets of agents a1, a2 and a3. If it is known that ov(s1,s2,a1,a2,$\supset$) and ov(s2,s2,a2,a3,to) then $I_{a1}(s1) \subset I_{a2}(s2)$ and $I_{a2}(s2) = I_{a3}(s3)$. Therefore $I_{a1}(s1) \subset I_{a3}(s3)$ and by implication ov(s1,s3,a1,a3,$\supset$).

Property 12: Inclusive overlap is a pseudo-symmetric relation. Formally, this property is expressed as follows:

$$(\forall x_1: W_1)\ (\forall x_2: W_2)\ (\forall a_1: 2^A)\ (\forall a_2: 2^A)\ (ov(x_1, x_2, a_1, a_2,\subset) \leftrightarrow ov(x_2,x_1, a_2, a_1,\supset))$$

Proof. Assume two predicate symbols s1, s2 and two sets of agents a1 and a2. If it is known that ov(s1,s2,a1,a2,$\subset$) then $I_{a2}(s2) \subset I_{a1}(s1)$ and by implication ov(s2,s1,a2,a1,$\supset$). Similarly, if ov(s2,s1,a2,a1,$\supset$) then $I_{a2}(s2) \subset I_{a1}(s1)$ and by implication ov(s1,s2,a1,a2,$\subset$).

Property 13: Partial overlap is an irreflexive relation for symbols in the vocabulary of the same specification given the interpretation of this specification by the same group of agents. Formally, this property is expressed as follows:

$$(\forall x_1: W_1)\ (\forall a_1: 2^A)\ (\neg\ ov(x_1, x_1, a_1, a_1,po))$$

Proof. Since by property 2 a symbol in the vocabulary of a specification totally overlaps with itself and overlap types are mutually exclusive, a symbol cannot have a partial overlap relation with itself.

Property 14: Partial overlap is a symmetric relation. Formally, this property is expressed as follows:

$$(\forall x_1: W_1)\ (\forall x_2: W_2)\ (\forall a_1: 2^A)\ (\forall a_2: 2^A)\ (ov(x_1, x_2, a_1, a_2, po) \rightarrow ov(x_2, x_1, a_2, a_1, po))$$

Proof. Assume two predicate or constant symbols s1 and s2 and two sets of agents a1 and a2. If it is known that ov(s1,s2,a1,a2,po) then $I_{a1}(s1) \cap I_{a2}(s2) \neq \emptyset$, $I_{a1}(s1) - I_{a2} \neq \emptyset$, and $I_{a2}(s2) - I_{a1}(s2) \neq \emptyset$ and by implication ov(s2,s1,a2,a1,po).

Property 15: Partial overlap is a pseudo-transitive relation with respect to total overlaps. Formally, this property is expressed as follows:

$$(\forall x_1: W_1)\ (\forall x_2:W_2)\ (\forall x_3: W_3)\ (\forall a_1: 2^A)\ (\forall a_2: 2^A)\ (\forall a_3: 2^A)\ (ov(x_1, x_2, a_1, a_2, po) \wedge ov(x_2, x_3, a_2, a_3, to) \rightarrow ov(x_1, x_3, a_1, a_3, po))$$

Proof. Assume three predicate symbols s1, s2 and s3 and three sets of agents a1, a2 and a3. If it is known that ov(s1,s2,a1,a2,po) and ov(s2,s3,a2,a3,to) then $I_{a1}(s1) \cap I_{a2}(s2) \neq \emptyset$, $I_{a1}(s1) – I_{a2}(s2) \neq \emptyset$, $I_{a2}(s2) – I_{a1}(s1) \neq \emptyset$ and $I_{a2}(s2) = I_{a3}(s3)$. Therefore $I_{a1}(s1) \cap I_{a3}(s3) \neq \emptyset$, $I_{a1}(s1) – I_{a3}(s3) \neq \emptyset$ and $I_{a3}(s3) – I_{a1}(s1) \neq \emptyset$, and by implication ov(s1,s3,a1,a3,po).