

# Interval Comparisons and Lattice Operations based on the Interval Overlapping Relation

Marco Nehmeier, Jürgen Wolff von Gudenberg

September 1, 2010

## 1 Introduction

In this paper we start with the definition of the interval overlapping relation showing the 13 different cases that may occur when to compare two non-empty intervals. This definition and especially the naming follow Allen who, in [1] defined this relation in a temporal logic setting. Other names may be chosen. This paper serves 2 purposes:

1. to define a solid foundation for interval comparisons and lattice operations
2. to provide an advanced interface for language embedding and application programs.

Based on the overlapping relation the paper then identifies 7 atomic independent interval operations, which form a base for the binary relation algebra (BRA) for intervals, [5].

The atomic operations can be used to compute the most commonly used interval comparisons, as those have been proposed in motion 13 [4]. Hence, this paper contributes to the rationale of motion 13.

The next section introduces the necessary lattice operations which are closely related with comparisons. These also can be derived from the overlapping relation.

In [6] we have shown that the general overlapping relation can be efficiently implemented in hardware. 4 bits are sufficient to encode the different states. Together with an object oriented interface allowing the direct access to the states (cases) this will presumably lead to a better performance of application programs.

### 1.1 Abstraction Levels

Concerning the levels of abstraction from level 1 that deals with real numbers and sets, via level 2 providing an abstract data type Interval with floating-point bounds, we come to the level 3 with executable representation, we try to stay as general or abstract as possible. Our main definition (Table 1) clearly is level 1. In our opinion that also holds for Table 2. Since P1788 defines the standard for floating-point intervals, we do need the level 2 tables 3 and 7.

## 2 Interval Overlapping Relation

Let  $Q$  be the finite set of states representing the 13 different possible situations of relative position of 2 non-empty intervals on the real line.

**Definition 1 (Interval Overlapping)** *The overlapping relation for two non-empty intervals is defined by the mapping*

$$\varpi : \overline{\mathbb{IR}} \times \overline{\mathbb{IR}} \rightarrow Q \quad (1)$$

$$A \varpi B \mapsto q_i \in Q, \quad i = 1 \dots 13 \quad (2)$$

where the predicates characterizing the  $q_i$  are given in Table 1

Table 2 illustrates the definition by deducing conditions for the endpoints and drawing sketches for each state. The states that are meaningful for empty intervals, see Remark 3, are also marked.

State	Definition
before	$\forall_a \forall_b a < b$
meets	$\forall_a \forall_b a \leq b \wedge \exists_{a'} \forall_b a' < b \wedge \exists_{a''} \exists_{b'} a'' = b'$
overlaps	$\exists_{a'} \forall_b a' < b \wedge \exists_{b'} \forall_a a < b' \wedge \exists_{a''} \exists_{b''} b'' < a''$
starts	$\exists_{a'} \forall_b a' \leq b \wedge \exists_{b'} \forall_a b' \leq a \wedge \exists_{b''} \forall_a a < b''$
containedBy	$\exists_{b'} \forall_a b' < a \wedge \exists_{b''} \forall_a a < b''$
finishes	$\exists_{b'} \forall_a b' < a \wedge \exists_{a'} \forall_b b \leq a' \wedge \exists_{b''} \forall_a a \leq b''$
equal	$\forall_a \exists_{b'} a = b' \wedge \forall_b \exists_{a'} b = a'$
finishedBy	$\exists_{a'} \forall_b a' < b \wedge \exists_{b'} \forall_a a \leq b' \wedge \exists_{a''} \forall_b a \leq a''$
contains	$\exists_{a'} \forall_b a' < b \wedge \exists_{a''} \forall_b b < a''$
startedBy	$\exists_{b'} \forall_a b' \leq a \wedge \exists_{a'} \forall_b a' \leq b \wedge \exists_{a''} \forall_b b < a''$
overlappedBy	$\exists_{b'} \forall_a b' < a \wedge \exists_{a'} \forall_b b < a' \wedge \exists_{b''} \exists_{a''} a'' < b''$
metBy	$\forall_b \forall_a b \leq a \wedge \exists_{b'} \exists_{a'} b' = a' \wedge \exists_{b''} \forall_a b'' < a$
after	$\forall_b \forall_a b < a$

Table 1: The 13 different states of interval overlapping situations

In order to keep the conditions readable we have omitted parentheses, the priority of the operators is assumed appropriately. We further qualify each “for all” quantifier with the plain letter and each “exists” quantifier with a primed letter, we further do not spell out the interval, i.e. we write  $\forall_b$  instead of  $\forall_{b \in B}$  etc.

**Remark 1** *The active or passive verb-forms are kind of inverse or reverse:  $A$  activates  $B \equiv B$  activatedBy  $A$*

**Remark 2** *Since containment is important, we group the situation with the point interval  $A = [a, a], a = \underline{b}$  under “starts” and not under “meets”. For the same reason we prefer the situation  $A = [a, a], a = \overline{b}$  to belong to “finishes” instead of “metBy”.*

**Remark 3** *Column 5 of Table 2 defines the mapping, if at least one operand is the empty set. Empty sets are commonly used only together with containment. Hence, we chose*

$$A = \emptyset, B \neq \emptyset \Rightarrow A \text{ containedBy } B$$

$$A \neq \emptyset, B = \emptyset \Rightarrow A \text{ contains } B$$

$$A = \emptyset, B = \emptyset \Rightarrow A \text{ equal } B$$

That does not mean that these are the only states possible for empty sets, see Remark 7. These relations correspond with the common sense rules for empty sets.

State	Condition				
before	$\overline{a} < \underline{b}$				
meets	$\underline{a} < \overline{a}$ $\overline{a} = \underline{b}$ $\underline{b} < \overline{b}$				
overlaps	$\underline{a} < \underline{b}$ $\underline{b} < \overline{a}$ $\overline{a} < \overline{b}$				
starts	$\underline{a} = \underline{b}$ $\overline{a} < \overline{b}$				
containedBy	$\underline{b} < \underline{a}$ $\overline{a} < \overline{b}$			$A = \emptyset$ $B \neq \emptyset$	
finishes	$\underline{b} < \underline{a}$ $\overline{a} = \overline{b}$				
equal	$\underline{a} = \underline{b}$ $\overline{a} = \overline{b}$			$A = \emptyset$ $B = \emptyset$	
finishedBy	$\underline{a} < \underline{b}$ $\overline{b} = \overline{a}$				
contains	$\underline{a} < \underline{b}$ $\overline{b} < \overline{a}$			$B = \emptyset$ $A \neq \emptyset$	
startedBy	$\underline{b} = \underline{a}$ $\overline{b} < \overline{a}$				
overlappedBy	$\underline{b} < \underline{a}$ $\underline{a} < \overline{b}$ $\overline{b} < \overline{a}$				
metBy	$\underline{b} < \overline{b}$ $\overline{b} = \underline{a}$ $\underline{a} < \overline{a}$				
after	$\overline{b} < \underline{a}$				

Table 2: The 13 different states in endpoint representation

Column 2 of Table 1 contains the set theoretic (level 1) predicates. The single states are obtained by shifting  $A$  from left to right and noticing each change of the predicate, see Figure 1. The three columns correspond (from left to right) to  $width(A)(> | = | <)width(B)$ , respectively.

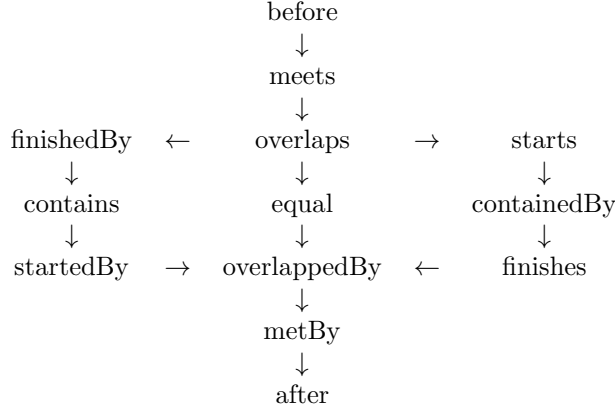


Figure 1: Shifting  $A$  from left to right

Since each bounded real interval can be characterized by 2 bounds, we can deduce equivalent formulas using the endpoints. Because our definition of unbounded intervals allows to use  $-\infty$ , or  $+\infty$  as endpoints, and because  $[-\infty, +\infty]$  equals  $[-\infty, +\infty]$  etc, we can apply these formulas for unbounded intervals as well.

**Remark 4** *For unbounded intervals we obtain*

$$[\underline{a}, \infty] \otimes [\underline{b}, \infty] = \begin{cases} \text{finishedBy} & : \underline{a} < \underline{b} \\ \text{equal} & : \underline{a} = \underline{b} \\ \text{finishes} & : \underline{a} > \underline{b} \end{cases}$$

*Analogous rules hold for left hand site unbounded intervals.*

**Remark 5** *Note that the rules for unbounded intervals (Remark 4) depend on the “bounded” tetrit, see [3].*

The illustrations in column 3 and following of Table 2 display the interval  $A$  on the bottom and the interval  $B$  on the top. This table provides the level 2 interface that exactly specifies the conditions on the endpoints. Note that we are using the endpoints for specification of the relation, and not necessarily for representation of the intervals.

### 3 Standard Interval Comparisons

**Remark 6 (Notation)** *From now on we carefully distinguish between*

- *(atomic) operations to check the state of overlapping,*
- *comparisons to be called for two intervals, or*
- *functions or relations that have no specified meaning.*

We have at least three alternatives to define interval comparisons based on the overlapping relation.

1. Define a data type `IntervalOverlapping` providing access to the 13 states.
2. Define a comparison for each of the 13 atomic operations.
3. Define a basic set of commonly used comparisons.

The first alternative has the advantage that the precise information about the relative positions of the two interval operands may be exploited to speed up evaluations of related comparisons.

The disadvantage is the new kind of interface defining an object instead of a set of boolean functions or operators.

We will discuss that approach in section 5. A hardware implementation is introduced in [6].

### 3.1 Atomic Operations

The second alternative provides 13 operations, denoted by the name of the state. By Remark 1 it is sufficient to provide 7 operations corresponding to the first 7 rows in the tables.

If these atomic operations are considered as independent, their treatment of empty sets has to be specified.

**Remark 7** *Atomic operations for empty arguments*

*According to Table 1*

- “before” and “after” return true, if at least one operand is empty.
- “meets” and “overlaps” return false, if at least one operand is empty.
- “starts” and “finishes” return false, if one operand is empty, but return true, if both operands are empty.
- The behavior of “containedBy” and “equal” is specified in Table 2.

**Remark 8** *Properties of atomic operations*

- “equal” is an equivalence relation.
- All others are asymmetric ( $a \sqsubset b \implies b \not\sqsubset a$ ).
- “before”, “starts”, “containedBy”, and “finishes” are transitive.

These 7 atomic operations can be used as a base for generating a binary relation algebra (BRA) of  $2^7 = 128$  interval comparisons, as proposed in [5]. We think that this opportunity is a challenge for the sophisticated programmer, but the every day user shall be given a simpler interface consisting of functions or operators.

However, the 7 atomic operations are obviously not the best candidates for comparisons of the standard, because they are not partial orders. (A partial order is reflexive, antisymmetric and transitive.) If we consider the isolated operations, we lose the context of the overlapping relation, that tells us to reuse the result for several related comparisons.

On one hand the overlapping relation is too detailed, but on the other hand it keeps the context in order to make related comparisons simpler.

### 3.2 Standard Comparisons

Hence, starting with the overlapping relation, we will now derive the interval comparisons. We perform the overlapping relation and then put together several related states.

Interval arithmetic is containment arithmetic. Hence, a comparison based on the operation

- $A$  containedBy  $B$   
checking whether  $A$  is contained in the interior of  $B$  is mandatory.
- Another important comparison is the test whether two intervals are disjoint. It is delivered by  
 $A$  before  $B$   
together with its reverse “after”.
- The third basic comparison is the “overlaps” test.  
It is used for interval ranking in minimization problems.

These 3 operations together with their reverses are characterized in Table 1 as those states which can be described only with the “strictly less” operator  $<$ . All the others compare at least one bound for equality. Hence special cases for singleton or point intervals have to be considered. In Table 1 we have carefully paid attention that the cases describe a complete partition of the argument space.

We add the related reflexive relations where  $\leq$  replaces  $<$ .

Table 3 displays our suggestion for a basic set of interval comparisons now using operator symbols instead of names. The 3rd column lists the set of overlapping states whose union describes the part of the domain that delivers “true” for the comparison.

The suggested comparisons are the most commonly used interval comparisons, any way. Another justification is given by Kulisch in [4].

Note that that motion uses an alternative specification for empty intervals delivering false, if either of the operands is empty in the comparisons  $<$ ,  $\prec$ ,  $\leq$ ,  $\preceq$ .

<i>comparison</i>	<i>implementation</i>	<i>definition</i>
$A = B$	$\underline{a} = \underline{b} \wedge \bar{a} = \bar{b}$	equal
$A \subset B$	$\underline{b} < \underline{a} \wedge \bar{a} < \bar{b}$	containedBy
$A < B$	$\underline{a} < \underline{b} \wedge \bar{a} < \bar{b}$	before $\vee$ meets $\vee$ overlaps
$A \prec B$	$\bar{a} < \underline{b}$	before
$A \subseteq B$	$\underline{b} \leq \underline{a} \wedge \bar{a} \leq \bar{b}$	equal $\vee$ starts $\vee$ containedBy $\vee$ finishes
$A \leq B$	$\underline{a} \leq \underline{b} \wedge \bar{a} \leq \bar{b}$	before $\vee$ meets $\vee$ overlaps $\vee$ starts $\vee$ equal $\vee$ finishedBy
$A \preceq B$	$\bar{a} \leq \underline{b}$	before $\vee$ meets $\vee$ equal $\wedge$ isSingleton( $A$ )

Table 3: standard comparisons

In Table 3 the states are considered as characteristic predicates describing the subset of the set of interval pairs that evaluates to “true”. Hence the  $\vee$  for states (predicates) corresponds to a  $\cup$  for subsets.

**Remark 9** *Note that we need an additional test for a point interval.*

We have a problem with the empty set, however. In  $A < B$ , e.g., we combine states with contradictory behavior, if one of the operands is the empty set. According to remark 7 we have to return “false” in the case of state meets but “true” in case of before. We resolve this ambiguity by the following remark.

**Remark 10** *The comparisons  $<$ ,  $\prec$ ,  $\leq$ ,  $\preceq$  deliver “false”, if at least one operand is the empty interval.*

**Proposition 1** *In Table 3 columns 2 and 3 are equivalent.*

**Proof:** We have to prove that Table 3 is correct. That means that for columns 2 and 3 the subsets  $S$  and  $T$  of  $\overline{\mathbb{IR}} \times \overline{\mathbb{IR}}$  that are declared by the conditions are identical. We outline the proof of row 3 ( $A < B$ ).

$$\begin{aligned}
S = S_{<} &:= \{(A, B) | \underline{a} < \underline{b} \wedge \bar{a} < \bar{b}\} \\
&= \{(A, B) | \underline{a} < \underline{b} \wedge \bar{a} < \bar{b} \wedge \text{true}\} \\
&= \{(A, B) | \underline{a} < \underline{b} \wedge \bar{a} < \bar{b} \wedge (\bar{a} < \underline{b} \vee \bar{a} = \underline{b} \vee \underline{b} < \bar{a})\} \\
&= \{(A, B) | \underline{a} < \underline{b} \wedge \bar{a} < \bar{b} \wedge (\bar{a} < \underline{b})\} \\
&\cup \{(A, B) | \underline{a} < \underline{b} \wedge \bar{a} < \bar{b} \wedge (\bar{a} = \underline{b})\} \\
&\cup \{(A, B) | \underline{a} < \underline{b} \wedge \bar{a} < \bar{b} \wedge (\underline{b} < \bar{a})\} \\
&= T_{\text{before}} \cup T_{\text{meets}} \cup T_{\text{overlaps}}
\end{aligned}$$

□

**Remark 11** *Let  $\sqsubset$  be an asymmetric relation. Then its related reflexive partial order relation  $\sqsubseteq$  is obtained by replacing the  $<$  operator with the  $\leq$  operator. The well known equation rule  $x \sqsubseteq y \wedge x \neq y \iff x \sqsubset y$  does not hold, because we replace twice.*

**Remark 12**  $\subseteq$  and  $\leq$  are partial orders, whereas  $\preceq$  is reflexive and antisymmetric only for point intervals.

$$A \subseteq B \wedge B \subseteq A \implies A = B \quad (3)$$

$$A \leq B \wedge B \leq A \implies A = B \quad (4)$$

$$A \preceq B \wedge B \preceq A \implies A = B = [a, a] \quad (5)$$

$$A \preceq A \implies A = [a, a] \quad (6)$$

In Table 3 we composed the comparisons as a disjunction of the state predicates. Or, in other words sets fulfilling the state predicates were united to the set fulfilling the comparisons.

In practice, the overlapping state of two intervals provides enough information in order to determine the result of a comparison as well as a lattice operation. Table 4 displays the situation that the overlapping state is known and we have to check whether the comparisons are true. Here “t” stands for true, no entry for false, “r” for reverse, and “s” means true in case of a singleton interval. Due to remark 1 Table 4 is symmetric with respect to the center row, t’s and r’s exchanged.

$A \infty B$	$A \subset B$	$A \prec B$	$A < B$	$A \subseteq B$	$A \preceq B$	$A \leq B$	$A = B$
before		$t$	$t$		$t$	$t$	
meets			$t$		$t$	$t$	
overlaps			$t$			$t$	
starts				$t$	$s(A)$	$t$	
containedBy	$t$			$t$			
finishes				$t$		$r$	
equal				$t$	$s$	$t$	$t$
finishedBy				$r$		$t$	
contains	$r$			$r$			
startedBy				$r$	$s(B)$	$r$	
overlappedBy			$r$			$r$	
metBy			$r$		$r$	$r$	
after		$r$	$r$		$r$	$r$	

Table 4: standard comparisons computed with overlapping relation

We have to show that Table 4 is correct.

Let us look into row 3:

The characteristic predicate for the state “starts” is

$$\underline{a} = \underline{b} \wedge \bar{a} < \bar{b} \quad (7)$$

The first condition does not hold for the first 3 irreflexive comparisons. But it holds for the reflexive cases. So we have to check the second condition. It obviously holds for containment and the  $\leq$  comparison, The  $\preceq$  operator is more subtle:

We have (7) and we have to show

$$\bar{a} \leq \underline{b} \quad (8)$$

We first treat the general case where neither  $A$  nor  $B$  is a point interval.

$$\begin{aligned} \underline{a} = \underline{b} \wedge \bar{a} < \bar{b} \wedge \underline{a} < \bar{a} &\Rightarrow \\ \underline{b} = \underline{a} < \bar{a} &\Rightarrow \neg(8) \end{aligned}$$

That proof also holds if  $B$  is a point interval. But the situation changes, if  $A$  is a singleton.

$$\underline{a} = \bar{a} = \underline{b} \quad (9)$$

and (8) is valid.

In the same manner we can prove the correctness of the other rows.

**Proposition 2** *Table 4 displays a correct way to implement the interval comparisons.*

### 3.3 Another set of comparisons

Another set of comparisons is used in the Fortran 95 language or in Sun’s interval extensions. Those are divided into 3 groups, the “set”, “possibly” and



“certainly” comparisons. They can easily be obtained from the standard comparisons and thus from the overlapping relation. We show the relationship in 2 tables.

**Proposition 3** *Table 5 shows how the set of Fortran 95 relations can be realized with the standard comparisons [2] and by direct use of the overlapping relation. Table 6 displays their realization.*

## 4 Lattice Operations

A lattice is a partially ordered set where each pair of elements has a *meet*<sup>1</sup> and a *join*. If this property holds for any (bounded) subset of elements, the lattice is called (conditionally) complete. Hence, a proper choice of *meet* and *join* as commutative, associative and absorbing binary operations provides a partial order relation. The other way round is also valid : If we have an appropriate partial order (a comparison), we can construct the corresponding complete lattice.

A direct introduction of the 7 basic comparisons that we deduced from the overlapping relation is given in [4]. The paper summarizes that interval comparisons are implemented by conditions on the endpoints.

Our two reflexive partial order relations induce the following lattices

1. The containment relation  $\subseteq$  defines the complete lattice  $(\overline{\mathbb{IR}}, \subseteq)$  with least element  $\emptyset$  and greatest element  $[-\infty, +\infty]$  Here the meet of two intervals, their intersection  $\cap$  and the join, their interval hull  $\cup$  are important and frequently used operations.
2. The “lessThanOrEqual” operation  $\leq$  does only form a conditionally complete lattice on  $\overline{\mathbb{IR}} \setminus \emptyset$ , because the meet  $[-\infty, -\infty]$  of the sequences like  $(x_n = [-infy, -n])$  is in  $(\mathbb{R}^* \times \mathbb{R}^*, \leq)$ , but not in  $\overline{\mathbb{IR}}$ . Similar with the join of growing sequences.
3. On level 2, however, since all sequences are finite,  $\overline{\mathbb{IF}} \setminus \emptyset$  is a complete lattice with smallest element  $[-\infty, minreal]$ , and greatest element  $[maxreal, \infty]$ . The meet or join of 2 intervals are  $glb(A, B) = [min(\underline{a}, \underline{b}), min(\overline{a}, \overline{b})]$  or  $lub(A, B) = [max(\underline{a}, \underline{b}), max(\overline{a}, \overline{b})]$ , respectively.
4. The third relation, the “meets” or “precedes” comparison is not a partial order, therefore we do not construct a corresponding complete lattice.

Note that the switch to level 2 is necessary, because in our definition  $\overline{\mathbb{IR}}$  does not contain unbounded singleton intervals.

The lattice operations can be determined with the information obtained from the interval overlapping relation in the same way as the comparisons, see Table 7.

**Proposition 4** *Table 7 is correct. Note that the intersection is empty, if one of the operands is empty, whereas the intervall hull is the other operand, cf. Table 2, column 5.*

---

<sup>1</sup>not to disturb with the overlapping state “meets”

Fortran 95 relation	level2 definition	standard comparisons	$A \otimes B$
isDisjoint(A,B)	$\bar{a} < \underline{b}$ or $\bar{b} < \underline{a}$	$A \prec B \vee B \prec A$	before $\vee$ after
isInterior(A,B)	$\underline{b} < \underline{a}$ and $\bar{a} < \bar{b}$	$A \subset B$	containedBy
isProperSubset(A,B)	$(\underline{b} \leq \underline{a} \text{ and } \bar{a} < \bar{b})$ or $(\underline{b} < \underline{a} \text{ and } \bar{a} \leq \bar{b})$	$A \subsetneq B \wedge \neg(A = B)$	starts $\vee$ containedBy $\vee$ finishes
isSubset(A,B)	$\underline{b} \leq \underline{a}$ and $\bar{a} \leq \bar{b}$	$A \subseteq B$	equal $\vee$ starts $\vee$ containedBy $\vee$ finishes
setIsLess(A,B)	$\underline{a} < \underline{b}$ and $\bar{a} < \bar{b}$	$A < B$	before $\vee$ meets $\vee$ overlaps
setIsLessOrEqual(A,B)	$\underline{a} \leq \underline{b}$ and $\bar{a} \leq \bar{b}$	$A \leq B$	before $\vee$ meets $\vee$ overlaps $\vee$ starts $\vee$ equal $\vee$ finishedBy
setIsEqual(A,B)	$\underline{a} = \underline{b}$ and $\bar{a} = \bar{b}$	$A = B$	equal
certainlyIsLess(A,B)	$\bar{a} < \underline{b}$	$A \prec B$	before
certainlyIsLessOrEqual(A,B)	$\bar{a} \leq \underline{b}$	$A \preceq B$	before $\vee$ meets $\vee$ (equal $\wedge$ isSingleton(A))
certainlyIsEqual(A,B)	$\bar{b} \leq \underline{a}$ and $\bar{a} \leq \underline{b}$	$A \preceq B \wedge B \preceq A$	equal $\wedge$ isSingleton(A)
possiblyIsLess(A,B)	$\underline{a} < \bar{b}$	$\neg(B \preceq A)$	before $\vee$ meets $\vee$ overlaps $\vee$ starts $\vee$ containedBy $\vee$ finishes $\wedge \neg$ isSingleton(A) $\vee$ equal $\vee$ finishedBy $\vee$ contains $\vee$ startedBy $\wedge \neg$ isSingleton(B))
possiblyIsLessOrEqual(A,B)	$\underline{a} \leq \bar{b}$	$\neg(B \prec A)$	$\neg$ (overlappedBy $\vee$ metBy $\vee$ after)
possiblyIsEqual(A,B)	$\underline{a} \leq \bar{b}$ or $\underline{b} \leq \bar{a}$	$\neg(A \prec B \vee B \prec A)$	$\neg$ (before $\vee$ after)

Table 5: Fortran 95 relations via overlapping relation

	<i>containment</i>				<i>set</i>			<i>certainly</i>			<i>possibly</i>		
	<i>dis</i>	<i>int</i>	<i>prop</i>	<i>sub</i>	<i>&lt;</i>	<i>≤</i>	<i>=</i>	<i>&lt;</i>	<i>≤</i>	<i>=</i>	<i>&lt;</i>	<i>≤</i>	<i>=</i>
<i>before</i>	<i>t</i>				<i>t</i>	<i>t</i>		<i>t</i>	<i>t</i>		<i>t</i>	<i>t</i>	
<i>meets</i>					<i>t</i>	<i>t</i>		<i>t</i>			<i>t</i>	<i>t</i>	<i>t</i>
<i>overlaps</i>					<i>t</i>	<i>t</i>					<i>t</i>	<i>t</i>	<i>t</i>
<i>starts</i>			<i>t</i>	<i>t</i>		<i>t</i>		<i>s(A)</i>			<i>t</i>	<i>t</i>	<i>t</i>
<i>containedBy</i>	<i>t</i>	<i>t</i>	<i>t</i>								<i>t</i>	<i>t</i>	<i>t</i>
<i>finishes</i>			<i>t</i>	<i>t</i>							<i>t</i>	<i>t</i>	<i>t</i>
<i>equal</i>				<i>t</i>	<i>s</i>	<i>t</i>		<i>s</i>	<i>s</i>		<i>ns</i>	<i>t</i>	<i>t</i>
<i>finishedBy</i>			<i>t</i>	<i>t</i>	<i>t</i>			<i>s(B)</i>			<i>t</i>	<i>t</i>	<i>t</i>
<i>contains</i>											<i>t</i>	<i>t</i>	<i>t</i>
<i>startedBy</i>											<i>t</i>	<i>t</i>	<i>t</i>
<i>overlappedBy</i>											<i>t</i>	<i>t</i>	<i>t</i>
<i>metBy</i>												<i>t</i>	<i>t</i>
<i>after</i>	<i>t</i>												

Table 6: Fortran comparisons computed with overlapping relation

$A \otimes B$	$A \cap B$	$A \cup B$	$glb(A, B)$	$lub(A, B)$
	$[\max(\underline{a}, \underline{b}), \min(\overline{a}, \overline{b})]$	$[\min(\underline{a}, \underline{b}), \max(\overline{a}, \overline{b})]$	$[\min(\underline{a}, \underline{b}), \min(\overline{a}, \overline{b})]$	$[\max(\underline{a}, \underline{b}), \max(\overline{a}, \overline{b})]$
<i>before</i>	$\emptyset$	$[\underline{a}, \overline{b}]$	$A$	$B$
<i>meets</i>	$[\overline{a}, \overline{a}]$	$[\underline{a}, \overline{b}]$	$A$	$B$
<i>overlaps</i>	$[\underline{b}, \overline{a}]$	$[\underline{a}, \overline{b}]$	$A$	$B$
<i>starts</i>	$A$	$B$	$A$	$B$
<i>containedBy</i>	$A$	$B$	$[\underline{b}, \overline{a}]$	$[\underline{a}, \overline{b}]$
<i>finishes</i>	$A$	$B$	$B$	$A$
<i>equal</i>	$A$	$A$	$A$	$A$
<i>finishedBy</i>	$B$	$A$	$A$	$B$
<i>contains</i>	$B$	$A$	$[\underline{a}, \overline{b}]$	$[\underline{b}, \overline{a}]$
<i>startedBy</i>	$B$	$A$	$B$	$A$
<i>overlappedBy</i>	$[\underline{a}, \overline{b}]$	$[\underline{b}, \overline{a}]$	$B$	$A$
<i>metBy</i>	$[\overline{b}, \overline{b}]$	$[\underline{b}, \overline{a}]$	$B$	$A$
<i>after</i>	$\emptyset$	$[\underline{b}, \overline{a}]$	$B$	$A$

Table 7: lattice operations computed with overlapping relation

result	operation	parameters	explanation
IOV	construct	(interval A, interval B)	computes state $A \otimes B$
bool	empty	()	test if $A$ or $B$ is empty
bool	disjoint	()	test for empty intersection
bool	equals	()	$A = B$
bool	subset	()	$A \subset B$ interior
bool	precedes	()	$A \prec B$ , before
bool	less	()	$A < B$
bool	subsetq	()	$A \subseteq B$
bool	preceq	()	$A \preceq B$
bool	leq	()	$A \leq B$
interval	intersect	()	intersection
interval	hull	()	interval hull
interval	glb	()	greatest lower bound
interval	lub	()	lest upper bound
float	first	()	first operand $A$
float	second	()	second operand $B$
$Q$	state	()	set of states $Q$ , see Def 1

Table 8: The abstract data type IOV

## 5 The Object-Oriented API

In section 3 we introduced 3 alternatives for the definition of interval comparisons. The first alternative always computes the interval overlapping relation for the two operands and then looks-up the result of the comparison in Table 4 or Table 7 for lattice operations. It has the advantage that the precise information about the relative positions of the two interval operands may be exploited to speed up evaluations of related comparisons.

The disadvantage is the new kind of interface defining an object instead of a set of boolean functions or operators. Furthermore one may argue that specific comparisons like  $A \subseteq B$  or  $A = B$  can already be checked by 2 (parallel) floating-point comparisons whereas the overlapping relation needs up to 8 floating-point comparisons. That has to be checked for the used algorithms, the specific hardware design, or the applicaton itself.

The class IOV manages the result of the overlapping test between 2 intervals like an abstract data type. Its range is a pair of intervals (the operands) together with the state information. Its operations are given in Table 8.

The state and the operands are initialized by the constructor and then never changed anymore. For each interval comparison and lattice operation there is a method without parameters, the operands are taken from the calling object. Two utility functions for frequently occuring tests are supplied. The operands and the current state can be accessed.

In such a setting the predefined comparisons can be evaluated, nothing more. When we, however, open the access to the overlapping states, we can compose new efficient comparisons. Hence, additionally to the abstract data type IOV the atomic operations shall be available as comparisons, see Table 9. For convenience all 13 states should be provided.

The two additionally defined functions `empty` and `disjoint` can be implemented

result	operation	parameters	explanation
bool	before	(interval A, interval B)	atomic op for state “before”
bool	meets	(interval A, interval B)	atomic op for state “meets”
bool	overlaps	(interval A, interval B)	atomic op for state “overlaps”
bool	starts	(interval A, interval B)	atomic op for state “starts”
bool	containedBy	(interval A, interval B)	atomic op for state “containedBy”
bool	finishes	(interval A, interval B)	atomic op for state “finishes”
bool	equal	(interval A, interval B)	atomic op for state “equal”
bool	finishedBy	(interval A, interval B)	atomic op for state “finishedBy”
bool	contains	(interval A, interval B)	atomic op for state “contains”
bool	startedBy	(interval A, interval B)	atomic op for state “startedBy”
bool	overlappedBy	(interval A, interval B)	atomic op for state “overlappedBy”
bool	metBy	(interval A, interval B)	atomic op for state “metBy”
bool	after	(interval A, interval B)	atomic op for state “after”

Table 9: Atomic comparisons

as follows:

```
bool disjoint(){
    return (state() == before or state() == after);
}
```

This is the comparison “isDisjoint” from the Fortran95 set. Tests for empty intervals are very common. We may distinguish 4 different methods indicating that either operand, or both or one of them, is empty. Here is the latter.

```
bool empty(){
    return first() == emptyset or second() == emptyset
}
```

More efficient implementations can be provided when the representation is fixed. As an example for the use of the interval overlapping relation we discuss the extended interval Newton method.

The classical algorithm 1 uses 6 interval comparisons and 2 intersections.

If we transform it into a relational algorithm 2, the information with 2 calls of the interval overlapping relation is sufficient to control the flow of the method. Note that the resulting state of overlapping is stored and re-used by methods or properties.

## 6 Conclusion

In this position paper we have outlined how the different sets of interval comparisons, like the one proposed in motion 13 [4] or the Fortran95 set can be defined and evaluated with the help of the interval overlapping relation. Newly combined interval comparisons can easily be defined, if the states of the interval overlapping relation are known.

---

**Algorithm 1:** INewton (classical)

---

**Input:** $f$  : function $Y$  : interval $\epsilon$  : epsilon $yUnique$  : flag $Zero$  : list of enclosing intervals $Info$  : flag vector $N$  : number**Output:** $Zero$ **begin**    **if**  $0 \notin f(Y)$  **then**        **return**  $(Zero, Info, N)$      $c \leftarrow \mu(Y)$ ;     $[Z_1, Z_2] \leftarrow f(c)/f'(Y)$  // extended division;     $[Z_1, Z_2] \leftarrow c - [Z_1, Z_2]$ ;     $V_1 \leftarrow Y \cap Z_1$ ;     $V_2 \leftarrow Y \cap Z_2$ ;    **if**  $V_1 = Y$  **then**         $V_1 \leftarrow [\underline{y}, c]$ ;         $V_2 \leftarrow [c, \overline{y}]$ ;    **if**  $V_1 \neq \emptyset$  **and**  $V_2 = \emptyset$  **then**         $yUnique \leftarrow yUnique$  **or**  $V_1 \subset Y$ ;    **foreach**  $i = 1, 2$  **do**        **if**  $V_i = \emptyset$  **then**            **continue**;        **if**  $drel(V_i) < \epsilon$  **then**             $N = N + 1$ ;             $Zero[N] = V_i$ ;             $Info[N] = yUnique$ ;        **else**             $INewton(f, V_i, \epsilon, yUnique, Zero, Info, N)$ ;    **end**    **return**  $(Zero, Info, N)$ ;**end**

---

---

**Algorithm 2:** INewtonRel (relational)

---

**Input:** $f$  : function $Y$  : interval $\epsilon$  : epsilon $yUnique$  : flag $Zero$  : list of enclosing intervals $Info$  : flag vector $N$  : number**Output:** [ $Zero$ ,  $Info$ ,  $N$ ]**begin**    **if**  $0 \notin f(Y)$  **then**        **return** ( $Zero, Info, N$ )     $c \leftarrow \mu(Y)$ ;     $[Z_1, Z_2] \leftarrow f(c)/f'(Y)$  // extended division;     $[Z_1, Z_2] \leftarrow c - [Z_1, Z_2]$ ;     $R_1 \leftarrow Y \otimes Z_1$ ;     $R_2 \leftarrow Y \otimes Z_2$ ;    **if**  $R_1.subseteq()$  **then**         $V_1 \leftarrow [\underline{y}, c]$ ;         $V_2 \leftarrow [c, \overline{y}]$ ;         $bisected \leftarrow true$ ;    **else if not**  $R_1.disjoint()$  **and**  $R_2.disjoint()$  **then**         $yUnique \leftarrow yUnique$  **or**  $R_1.state() == containedBy$ ;    **foreach**  $i = 1, 2$  **do**        **if not**  $bisected$  **then**            **if**  $R_i.disjoint()$  **then**                **continue**;             $R_i \leftarrow Y \otimes Z_i$ ;             $V_i \leftarrow R_i.intersect()$ ;        **if**  $drel(V_i) < \epsilon$  **then**             $N = N + 1$ ;             $Zero[N] = V_i$ ;             $Info[N] = yUnique$ ;        **else**             $INewtonRel(f, V_i, \epsilon, yUnique, Zero, Info, N)$ ;    **end**    **return** ( $Zero, Info, N$ );**end**

---

## References

- [1] J. F. Allen. Maintaining knowledge about temporal intervals. *Communications of the ACM*, 26/11/1983:832–843, 1983.
- [2] N. Hayes. IEEE Interval Standard Working Group - P1788 e-mail of April 11, 2010. <http://grouper.ieee.org/groups/1788/>.
- [3] N. Hayes. IEEE Interval Standard Working Group - P1788 Motion 8 Exception Handling, Sept 2009. <http://grouper.ieee.org/groups/1788/>.
- [4] U. Kulisch. IEEE Interval Standard Working Group - P1788 Motion 13 Version 4. <http://grouper.ieee.org/groups/1788/>.
- [5] D. Lohez. Comparison Relations for Intervals : A Framework. <http://grouper.ieee.org/groups/1788/>.
- [6] M. Nehmeier, S. Siegel, and J. Wolff von Gudenberg. Parallel Detection of Interval Overlapping. In *Proceedings of Para2010, to appear*, Apr. 2010. submitted to para2010 <http://www.yourhost.is/para2010/home.html>.