Incompleteness and Incomparability in Preference Aggregation

M.S. Pini, F. Rossi, K. Venable[1] and **T. Walsh** [2]

Abstract. We consider how to combine the preferences of multiple agents despite the presence of incompleteness and incomparability in their preference orderings. An agent's preference ordering may be incomplete because, for example, there is an ongoing preference elicitation process. It may also contain incomparability, which can be useful, for example, in multi-criteria scenarios. We focus on the problem of computing the *possible* and *necessary* winners, that is, those outcomes which can be or always are the most preferred for the agents. Possible and necessary winners are useful in many scenarios. For example, preference elicitation need only focus on the unknown relations between possible winners and can ignore completely all other outcomes. Whilst computing the sets of possible and necessary winners is in general a difficult problem, we identify sufficient conditions where we can obtain the necessary winners and an upper approximation of the set of possible winners in polynomial time. Such conditions concern either the language for stating preferences, or general properties of the preference aggregation function.

1 Introduction

We consider a multi-agent setting where each agent specifies its preferences by means of an ordering over the possible outcomes. Such an ordering may include both incomparability and incompleteness. A pair of outcomes can be ordered, incomparable, in a tie, or the relationship between them may not yet be specified. Incomparability and incompleteness represent very different concepts. Outcomes may be incomparable because the agent does not wish very dissimilar outcomes to be compared. For example, we might not want to compare a biography with a novel as the criteria along which we judge them are just too different. Outcomes can also be incomparable because the agent has multiple criteria to optimize. For example, we might not wish to compare a faster but more expensive laptop with a slower and cheaper one. Incompleteness, on the other hand, represents simply an absence of knowledge about the relationship between certain pairs of outcomes. Incompleteness arises naturally when we have not fully elicited an agent's preferences or when agents have privacy concerns which prevent them revealing their complete preference ordering.

We wish to aggregate together the agents' preferences into a single preference ordering. How do we modify preference aggregation functions to deal with incompleteness? One possibility is to consider all possible ways in which the incomplete preference orders can be consistently completed. In each possible completion, preference aggregation may give different optimal elements (or *winners*). This leads to the idea of the *possible winners* (those outcomes which are winners in at least one possible completion) and the *necessary winners* (those outcomes which are winners in all possible completions) [7]. Possible and necessary winners are useful in many scenarios including preference elicitation [3]. In fact, elicitation is over when the set of possible winners coincides with that of the necessary winners [5]. In addition, as we argue later, preference elicitation can focus just on the incompleteness concerning those outcomes which are possible and necessary winners. We can ignore completely all other outcomes.

Computing the set of possible and necessary winners is in general a difficult problem. However, we identify sufficient conditions that assure tractability. Such conditions concern properties of the preference aggregation function, such as monotonicity and independence to irrelevant alternatives [1], which are desirable and natural properties to require. Restrictions on the possible results returned by the preference aggregation function can also ensure that an upper approximation on the set of possible winners can be computed tractably. One such restriction is when the preference aggregation functions takes in incomparability but never returns it.

Parts of this paper have appeared in [9].

2 Basic notions

Preferences. We assume that each agent's preferences are specified via a (possibly incomplete) partial order with ties (IPO) over the set of possible outcomes, that we will denote by Ω. An incomplete partial order is a partial order where some relation between pairs of outcomes is unknown. Given two outcomes A and B, an agent will specify exactly one of the following: $A < B$, $A > B$, $A = B$, $A \sim B$, or $A?B$, where $A \sim B$ means that A and B are incomparable, and $A?B$ that the relation between A and B is unknown, this means that it can be any element of $\{=, >, <, \sim\}$.

Example 1. Given outcomes A, B, and C, an agent may state preferences such as $A > B$, $B \sim C$, and $A > C$, or also just $A > B$ and $B \sim C$. However, an agent cannot state preferences such as $A > B$, $B > C$, $C > A$, or also $A > B$, $B > C$, $A \sim C$ since neither are POs. □

Profiles. A *profile* is a sequence of n partial orders $p_1, \ldots, p_n$ over outcomes, one for each agent $i \in \{1, \ldots, n\}$, describing the preferences of the agents. An *incomplete profile* is a sequence in which one or more of the partial orders is incomplete.

Social welfare and preference aggregation. *Social welfare functions* [1] are functions from profiles to partial orders with ties. Given a social welfare function f, we define a corresponding *preference*

[1] University of Padova, Italy, email: {mpini,frossi,kvenable}@math.unipd.it
[2] NICTA and UNSW Sydney, Australia, email:Toby.Walsh@nicta.com.au

aggregation function, written pa_f, which is a function from incomplete profiles to sets of partial orders with ties (POs). Precisely, given an incomplete profile $ip = (ip_1, \ldots, ip_n)$, where the ip_i's are IPOs, consider all the profiles, say $p_1, \ldots, p_k$, obtained from ip by replacing any occurrence of ? in the ip_i's with either $<, >$, $=$, or $\sim$ which is consistent with a partial order. Let us then set $pa_f(ip) = \{f(p_1), \ldots, f(p_k)\}$. This set will be called the *set of results* of f on profile ip.

Example 2. Consider the Pareto social welfare function f defined as follows [1]: given a profile p, for any two outcomes A and B, if all agents say $A > B$ or $A = B$ and at least one says $A > B$ in p, then $A > B \in f(p)$; if all agents say $A = B$ in p, then $A = B \in f(p)$; otherwise, $A \sim B \in f(p)$. In Figure 1 we show an example with three agents and three outcomes A, B, and C. □

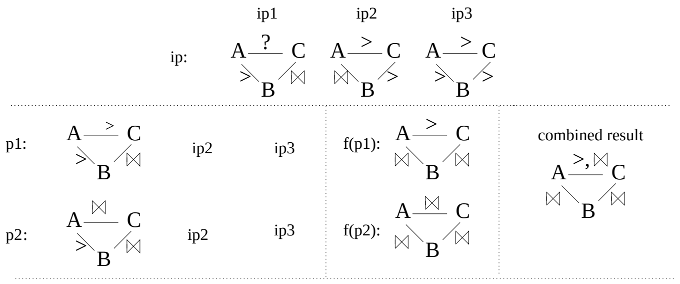


Figure 1. An incomplete profile ip, its completions p_1 and p_2, the results $f(p_1)$ and $f(p_2)$, and the combined result $cr(f, ip)$.

Necessary and possible winners. We extend to the case of partial orders the notions of possible and necessary winners presented in [7] in the case of total orders. Given a social welfare function f and an incomplete profile ip, we define *necessary winners* of f given ip as all those outcomes which are maximal elements in all POs in $pa_f(ip)$. A necessary winner must be a winner, no matter how incompleteness is resolved in the incomplete profile. Analogously, the *possible winners* are all those outcomes which are maximal elements in at least one of the POs in $pa_f(ip)$. A possible winner is a winner in at least one possible completion of the incomplete profile.

We will write $NW(f, ip)$ and $PW(f, ip)$ for the set of necessary and possible winners of f on profile ip. We will sometimes omit f and/or ip, and just write NW and PW when they will be obvious or irrelevant.

Example 3. In Example 2, A and B are the necessary winners, since they are top elements in all POs $f(p_i)$, for all $i = 1, 2$. C is a possible winner since it wins in $f(p_2)$.□

Combined result. Unfortunately, the set of results can be exponentially large. We will therefore also consider a compact representation that is polynomial in size. This necessarily throws away information by compacting together results into a single combined result. Given a social welfare function f and an incomplete profile ip, consider a graph, whose nodes are the outcomes, and whose arcs are labeled by non-empty subsets of $\{<, >, =, \sim\}$. Label l is on the arc between outcomes A and B if there exists a PO in $pa_f(ip)$ where A and B are related by l. This graph will be called the *combined result* of f on ip, and will be denoted by $cr(f, ip)$. If an arc is labeled by set $\{<, >, =, \sim\}$, we will say that it is *fully incomplete*. Otherwise, we say that it is *partially incomplete*. The set of labels on the arc between A and B will be called $rel(A, B)$.

Example 4. The combined result for Example 2 is shown in Figure 1. □

3 From the combined result to winners

We would like to compute efficiently the set of possible and necessary winners, as well as to determine whether a given outcome is a possible or a necessary winner. In general, even if the social welfare function is polynomial, incompleteness in the profile may require us to consider an exponential number of completions. As observed in [7], determining the possible winners is in NP, and the necessary winners is in coNP.

We first consider how to compute the possible and necessary winners given the combined result. We will then consider how to compute the combined result.

Consider the arc between an outcome A and an outcome B in the combined result. Then, if this arc has the label $A < B$, then A is not a necessary winner, since there is an outcome B which is better than A in some result. If this arc *only* has the label $A < B$, then A is not a possible winner since we must have $A < B$ in all results. Moreover, consider all the arcs between A and every other outcome C. Then, if no such arc has label $A < C$, then A is a necessary winner. Notice, however, that, even if none of the arcs connecting A have just a single label $A < C$, A could not be possible winner. A could be better than some outcomes in every completion, but there might be no completion where it is better than all of them.

We can thus define the following Algorithm 1 to compute NW and a superset of PW, that we will call PW^*.

Algorithm 1: Computing NW and PW^*

```
Input: f: preference aggregation function; ip: incomplete profile;
Output: P, N: sets of outcomes;
$P \leftarrow \Omega$;
$N \leftarrow \Omega$;
foreach $O \in \Omega$ do
    if $\exists\, O' \in \Omega$ such that $(O < O') \in cr(f,ip)$ then
        $N \leftarrow N - O$;
    if $\exists\, O' \in \Omega$ such that $(O < O') \in cr(f,ip)$ and
    $(OrO') \notin cr(f, ip)$ for $r \in \{=, >, \sim\}$ then $P \leftarrow P - O$;
return $P$, $N$;
```

Theorem 1 *Algorithm 1 terminates in $O(m^2)$ time, where $m = |\Omega|$, returning $N = NW$ and $P = PW^* \supseteq PW$.*

Example 5. Consider the set of results $pa_f(ip) = (f(p_1), f(p_2))$, where $f(p_1) = \{A > B, B > C, A > C\}$ and $f(p_2) = \{C > B, B > A, C > A\}$. Then the combined result is the graph with nodes A, B, and C, in which all labels are $\{>, <\}$. Since there are no arcs with only the label $<$, Algorithm 1 returns $P = PW^* = \{A, B, C\}$. However, B is not a possible winner, since it does not win in $f(p_1)$ or $f(p_2)$.□

Example 6. Consider the set of results $(f(p_1), f(p_2))$, where $f(p_1) = \{A = B, A < C, B < C\}$ and $f(p_2) = \{A = C, A < B, C < B\}$. Then the combined result is the graph with nodes A, B, and C, in which the arcs between A and B, and between A and C are labeled $\{=, <\}$, and the arc between B and C has the label $\{<, >\}$. Here Algorithm 1 would compute $N = NW = \emptyset$ and $P = PW^* = \{A, B, C\}$. However, A is not a possible winner. □

To summarize, we have shown how to compute the set of necessary winners, as well as a superset of the set of possible winners from the combined result in time quadratic in the number of outcomes. Unfortunately, the computation of the combined result requires applying the preference aggregation function f on $O(4^{n\times m^2})$ possible completions. In each of the n IPOs of an incomplete profile there could be up to m^2 relations which are not revealed. Even if f can be computed in polynomial time, this is exponential in the number of both agents and outcomes. In later sections, we will discuss circumstances under which we can compute an approximation to the combined result efficiently.

4 A consistency test for the possible winners

The set $P = PW^*$ computed by Algorithm 1 can be different from the set of possible winners for two reasons. First, as the algorithm considers one arc at a time, it is not able to recognize global inconsistencies due to violation of the transitivity property. This can be seen in Example 6, where A is in set P but it is not a possible winner. In fact, there is no way to choose a label for each arc such that A is a winner and we have a PO. Second, the algorithm starts from the combined result where we have already thrown away some information. Even if we consider only the POs that are consistent with the combined result, we may still have more POs than returned by the preference aggregation function. For instance, in Example 5, B is included in the set P but is not a possible winner. Thus, if we just use the combined result, there is no way to compute the set PW exactly. We can, however, eliminate the first problem by deleting outcomes which cannot be possible winners because of intransitivity.

PO-Consistency test. To check whether O is a possible winner, we eliminate $O < O'$ from the label of each arc connecting O in the combined result, and test whether the new structure, which we call the *possibility structure* of outcome O (or $poss(O)$) is consistent with transitivity. This test can be reduced to testing the consistency of a set of branching temporal constraints [2]. In branching temporal reasoning, the possible relations between two events are exactly the possible labels of arcs in the combined result: $<, >, =, \sim$. Thus a branching temporal problem is a set of of constraints of the form xRy, where $R \subset \{<, >, =, \sim\}$. It is shown in [2] that checking the consistency of a branching temporal constraint problem is NP-hard. Thus, it is in general a difficult problem to check the consistency of a possibility structure.

Theorem 2 *Given the combined result $cr(f, ip)$ and an outcome O, checking the consistency of the possibility structure $poss(O)$ is NP-hard.*

Fortunately, however, there are many classes of branching temporal constraint problems which are tractable, that are likely to occur in our setting.

Sufficient conditions for a tractable PO-consistency test. One of the tractable classes is defined by restricting the labels to the set $\{<, >, =\}$. That is, we do not permit incomparability ($\sim$) in the result. If we do this, then a possibility structure is a set of (non-branching) temporal constraints. This coincides with the temporal constraint language Γ_b, which is defined and shown to be tractable in [2].

Theorem 3 *Given the combined result $cr(f, ip)$, if none of its labels include incomparability, then checking the consistency of a possibility structure is polynomial.*

This situation occurs when the social welfare function only ever returns a total order with ties. Examples are any of the social welfare functions considered in classical voting theory (which take in total orders and return a total order).

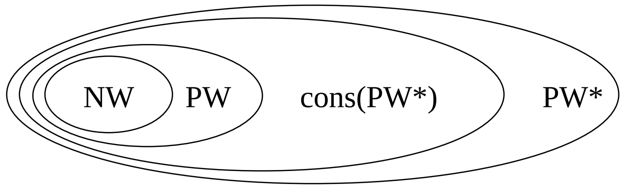
Another case in which it is easy to remove inconsistencies due to non-transitivity is when the social welfare function is Pareto (see Example 2 for its definition), provided that, for each pair of outcomes, at least one agent declares a preference (that is, $<, >$, or $\sim$) over them. In this case, checking if an outcome is a possible winner is equivalent to checking the consistency of a set of branching temporal constraints built on the language called Γ_A in [2], which is shown to be tractable.

Theorem 4 *Given the combined result, consider the Pareto social welfare function and an incomplete profile where each pair of outcomes is strictly ordered by at least one agent. Then, for every outcome O, the consistency test of its possibility structure $poss(O)$ is tractable.*

This language allows any subset of $\{<, >, =, \sim\}$ among two events, except those that contain both $<$ and $>$. By using the Pareto rule, we are guaranteed that no arc is labeled both $<$ and $>$ in the combined result. If $A < B$ holds in one result, then it cannot be that $A > B$ holds in another result, unless no agent expresses a preference among A and B, which is false by assumption.

Similar tractability results hold for other classes of branching temporal constraints [2]. If we have a preference aggregation function such that the labels of the arcs in the combined result belong to one such class, these tractability results allow us to deduce that checking consistency of a possibility structure is tractable.

Necessary and possible winners: exact sets and approximations. Given any set of outcomes S and a combined result C, we can check each outcome in S for consistency by evaluating the structures $poss(O)$ for every $O \in S$, and eliminating those outcomes that are not consistent. We will denote the remaining set of outcomes by $cons(S, C)$. C will be omitted when obvious. The relationship between sets NW, PW, PW^*, and $cons(PW^*)$, with reference to $cr(f, ip)$, can be seen in the following figure:



For the specific case of the Pareto function, it is possible to prove that $cons(PW^*, cr(f, ip)) = PW$. Thus the consistency test is enough to determine the possible winners.

Theorem 5 *Given the Pareto social welfare function and an incomplete profile ip where each pair of outcomes is strictly ordered by at least one agent, we have $cons(PW^*, cr(Pareto, ip)) = PW$.*

5 Tractable computation of possible and necessary winners

We have shown how to compute the set of necessary winners (that is, NW) and an upper approximation of the set of possible winners (that is, PW^* or $cons(PW^*)$), given the combined result. Unfortunately, we noticed also that computing the combined result is itself a difficult problem in general. In this section we identify some properties of preference aggregation functions which allow us to compute an upper approximation to the combined result in polynomial time, assuming that the social welfare function is polynomially computable. This can then be used to compute possible and necessary winners again in polynomial time. We recall that the set of labels of an arc between A and B in the combined result is called $rel(A, B)$.

Computing $rel(A, B)$ when f is IIA. The first property we consider is independence to irrelevant alternatives (IIA). A social welfare function is said to be IIA when, for any pair of outcomes A and B, the ordering between A and B in the result depends only on the relation between A and B given by the agents [1]. Many preference aggregation functions are IIA, and this is a desirable property which is related to the notion of fairness in voting theory [1]. Given a function which is IIA, to compute the set $rel(A, B)$, we just need to ask each agent their preference over the pair A and B, and then use f to compute all possible results between A and B. However, if agents have incompleteness between A and B, f has to consider all the possible completions, which is exponential in the number of such agents.

Computing $rel(A, B)$ when f is IIA and monotonic. Assume now that f is also monotonic. We say that an outcome B *improves with respect to* another outcome A if the relationship between A and B does not move left along the following sequence: $>, \geq$, ($\sim$ or $=$), $\leq, <$. For example, B improves with respect to A if we pass from $A \geq B$ to $A \sim B$. A social welfare function f is *monotonic* if for any two profiles p and p' and any two outcomes A and B passing from p to p' B improves with respect to A in one agent i and $p_j = p'_j$ for all $j \neq i$, then passing from $f(p)$ to $f(p')$ B improves with respect to A.

Consider now any two outcomes A and B. To compute $rel(A, B)$ under IIA and monotonicity, again, since f is IIA, we just need to consider the agents' preferences over the pair A and B. However, now we don't need to consider all possible completions for all agents with incompleteness between A and B, but just two completions: $A < B$ and $B > A$. Function f will return a result for each of these two completions, say AxB and AyB, where $x, y \in \{<, >, =, \sim\}$. Since f is monotonic, the results of all the other completions will necessarily be between x and y in the ordering $>, \geq$, ($\sim$ or $=$), $\leq$, $<$. By taking all such relations, we obtain a superset of $rel(A, B)$, that we call $rel^*(A, B)$. In fact, monotonicity of f assures that, if we consider profile $A < B$ and we get a certain result, then considering profiles where A is in a better position w.r.t. B (that is, $A > B$, $A = B$, or $A \sim B$), will give an equal or better situation for A in the result. Notice that we have obtained set $rel^*(A, B)$ in time polynomial in the number of agents as we only needed to consider two completions.

Under the IIA and monotonicity assumptions, we can thus obtain in polynomial time a structure similar to the combined result, but with possibly more labels on the arcs. We call $cr^*(f, ip)$ such a structure. However, notice that the additional labels in $cr^*(f, ip)$, if any, have a very specific structure. Only arcs with all four labels $<, >, \sim$, and $=$ can have additional labels, and such labels can only be $\sim$ and $=$.

Given the structure $cr^*(f, ip)$, we can now use the same techniques that we have described for the combined result to determine the possible and necessary winners. Thus, we can apply Algorithm 1 to $cr^*(f, ip)$. If N' and P' are the sets returned by this algorithm, it is possible to show that $N' = NW$ and $P' = PW^*$.

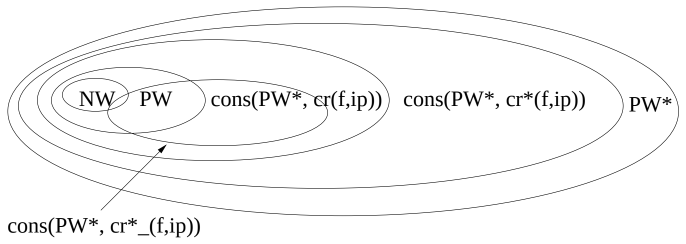
Theorem 6 *Given a IIA and monotonic social welfare function f, and an IPO ip, the sets NW and PW^* can be computed in polynomial time.*

In fact, the possible addition of labels $\sim$ or $=$ to some arcs does not change the necessary winners computed by the algorithm, as checking for necessary winners only looks for arcs with a label $<$. The same holds for determining the possible winners, which depends on arcs with only label $<$, which never have additional labels in $cr^*(f, ip)$. Thus we have a polynomial way to compute both NW and PW^*.

Let us now consider the application of the consistency check to the outcomes in PW^* based on the structure $cr^*(f, ip)$. It is easy to see that $cons(PW^*, cr(f, ip)) \subseteq cons(PW^*, cr^*(f, ip))$. In fact, the new structure may contain more labels than the old one, and thus it could be possible to select a PO which is not included in the old structure. Thus we may obtain an upper approximation of PW, but at the gain of being able to compute it in polynomial time.

A lower approximation of $cons(PW^*, cr(f, ip))$ can be obtained as well. It is enough to consider the structure obtained from $cr^*(f, ip)$ by eliminating labels $\sim$ and $=$ in all arcs where all four labels (that is, $<, >, \sim, =$) appear. Let us call $cr^*_-(f, ip)$ such a structure. If we run the consistency check on this structure, we get the set $cons(PW^*, cr^*_-(f, ip))$, which is included in $cons(PW^*, cr^*(f, ip))$, but may not contain all possible winners. However, it certainly contains at least one possible winner, that wins in one of the two completions we consider.

Thus, if the preference aggregation function is IIA and monotonic, and the conditions for a tractable consistency check are met, we can compute in polynomial time all the following sets: NW, PW^*, $cons(PW^*, cr^*(f, ip))$, $cons(PW^*, cr(f, ip))$, and $cons(PW^*, cr^*_-(f, ip))$. The relationship among these sets can be seen in the following figure:



Consider again the Pareto function, which is both monotone and IIA. In this case, as noticed above, for any pair of outcome A and B, $rel(A, B)$ cannot contain both $A < B$ and $B < A$. Thus the structure cr^* coincides with cr, since the only difference between the two structures is the possible addition of labels in arcs where both $<$

and $>$ are present. Moreover, we also noticed that the consistency test can be achieved in polynomial time, and $cons(PW^*, cr) = PW$. We, thus, have the following result.

Theorem 7 *Given the Pareto social welfare function and an IPO where each pair of outcomes is strictly ordered by at least one agent, the sets of NW and PW can be determined in polynomial time.*

6 Preference elicitation

One use of necessary and possible winners is in eliciting preferences [3]. Preference elicitation is the process of asking queries to agents in order to determine their preferences over outcomes.

At each stage in eliciting agents' preferences, there is a set of possible and necessary winners. When $NW = PW$, preference elicitation can be stopped since we have enough information to declare the winners, no matter how the remaining incompleteness is resolved [5]. At the beginning, NW is empty and PW contains all outcomes. As preferences are declared, NW grows and PW shrinks. At each step, an outcome in PW can either pass to NW or become a loser.

Determining the winners. In those steps where PW is still larger than NW, we can use these two sets to guide preference elicitation and avoid useless work. In fact, to determine if an outcome $A \in PW - NW$ is a loser or a necessary winner, it is enough to ask agents to declare their preferences over all pairs involving A and another outcome, say B, in PW. In fact, any outcome outside PW is a loser, and thus is dominated by at least one possible winner.

If the preference aggregation function is IIA, then all those pairs (A, B) with a defined preference for all agents can be avoided, since they will not help in determining the status of outcome A. Moreover, IIA allows us to consider just one profile when computing the relations between A and B in the result, and assures that the result is a precise relation, that is, either $<$, or $>$, or $=$, or $\sim$. In the worst case, we need to consider all such pairs. To determine all the winners, we thus need to know the relations between A and B for all $A \in PW - NW$ and $B \in PW$. Again, there are examples where all such pairs must be considered.

We can thus use the following Algorithm 2, which in $O(|PW|^2)$ steps eliminates enough incompleteness to determine the winners. At each step, the algorithm asks each agent to express its preferences on a pair of outcomes (via procedure $ask(A, B)$) and aggregates such preferences via function f. If function f is polynomially computable, the whole computation is polynomial in the number of agents and outcomes.

Algorithm 2: Winner determination

```
Input: PW, NW: sets of outcomes; $f$: preference aggregation function;
Output: W: set of outcomes;
$wins$: bool;
$P \leftarrow PW$; $N \leftarrow NW$;
while $P \neq N$ do
    choose A $\in P - N$;
    $wins \leftarrow true$; $P_A \leftarrow P - \{A\}$;
    repeat
        choose B $\in P_A$;
        if $\exists$ an agent such that A?B then
            ask(A,B);
            compute $f$(A,B);
            if $f(A, B) = (A > B)$ then
                $P \leftarrow P - \{B\}$;
            if $f(A, B) = (A < B)$ then
                $P \leftarrow P - \{A\}$; $wins \leftarrow false$;
        $P_A \leftarrow P_A - \{B\}$;
    until $f(A, B) \neq (A < B)$ or $P_A \neq \emptyset$ ;
    if $wins$ = $true$ then
        $N \leftarrow N \cup \{A\}$;
$W \leftarrow N$;
return $W$;
```

Theorem 8 *If f is IIA and polynomially computable, then determining the set of winners via preference elicitation is polynomial in the number of agents and outcomes.*

Using the results of the previous sections, under certain conditions we know how to compute efficiently the necessary winners and an upper approximation of the set of possible winners. Thus Algorithm 2 can be used with such an upper approximation. This means that we will possibly consider more pairs than needed. For example, if we use set PW^* rather than PW, we could examine also those pairs between elements in $PW^* - PW$ and elements in PW^*.

Discovering an agent's inconsistencies. The consistency test defined in the previous sections applies to the combined result structure (or its approximations). However, when we have just one agent, and the preference aggregation function is the identity, the combined result coincides with the agent's preferences. If we relax the assumption that agents provide preferences in the form of a PO or an IPO, the consistency check in this case determines whether the possibly incomplete preferences given by the agent are consistent.

Notice that agents only express non-disjunctive information about their preferences. That is, exactly one of $A > B$, $A < B$, $A = B$, $A \sim B$, or $A?B$. It is easy to see that a set of such preferences is consistent iff it is consistent also when each $A?B$ is replaced with $A \sim B$. Since labels $A > B$, $A < B$, $A = B$, and $A \sim B$ constitute a subset of language Γ_A [2], testing consistency of an agent's preferences is always tractable. Notice that the same would hold if we allow agents to express their preferences with partial incompleteness. For example, an agent may specify that A and B are either ordered or incomparable. This would still be within language Γ_A and thus consistency would still be tractable.

This can be useful also in a multi-agent setting, to determine the consistency of the preferences given so far by each of the agents. If at some step we realize that some of the agents have provided inconsistent preferences, we can communicate this to the agents.

Theorem 9 *If the agents express their preferences over a pair of outcomes, say A and B, using one of $A > B$, $A < B$, $A = B$, $A \sim B$ or $A?B$, then testing the consistency of the agents' preferences is polynomial.*

If the consistency test is successful, we can exploit the information deduced by the consistency enforcement to avoid asking for preferences which are implied by previously elicited ones. If instead we detect inconsistency, then we can help the agent to make their preferences consistent by providing one or more triangles where consistency fails.

7 Related work

In [7] preference aggregation functions for combining incomplete total orders are considered. Compared to our work, we permit both incompleteness and incomparability, while they allow only for incompleteness. Second, they consider social choice functions which return the (non-empty) set of winners. Instead, we consider social welfare functions which return a complete partial order. Social welfare functions give a finer grained view of the result. Third, they consider specific voting rules like the Borda procedure whilst we have focused on general properties that ensure tractability.

While voting theory has been mainly interested in possibility or impossibility results about social choice or social welfare functions, recently there has been some interest also in computational properties of preference aggregation [10, 8, 7, 5]. It is clear in fact that voting theory can be useful in multi-agent preference aggregation systems. However, such systems, to be usable in practice, need to know both what they can do and also how difficult it is to do it.

The results presented in this paper can be useful not just for combining preferences from multiple agents, but also for combining multiple conflicting preferences from a sigle agent. A recent work addressing the combination of multiple complex preferences is presented in [4]. It proposes various semantic optimization techniques applicable to preference queries. These techniques are based on the winnow operator, an algebraic operator that picks from a given relation the set of the most preferred outcomes, according to a given preference formula. In [6] it is proposed another methodology for combining complex preferences that is based on the SV-semantics, that is, a semantics characterizing equally good values among the indifferent ones.

8 Future work

A direction for future work involves adding constraints to agents' preferences. This means that preference aggregation must take into account the feasibility of the outcomes. Thus possible and necessary winners must now be feasible.

It is also important to consider compact knowledge representation formalisms to express agents' preferences, such as CP-nets and soft constraints. Possible and necessary winners should then be defined directly from such compact representations, and preference elicitation should concern statements allowed in the representation language.

Finally, a possibility distribution over the completions of an incomplete preference relation between two outcomes can be used to have additional information to exploit when computing possible and necessary winners.

Acknowledgements

This work has been supported by Italian MIUR PRIN project “Constraints and Preferences” (n. 2005015491).

REFERENCES

[1] K. J. Arrow, A. K. Sen, and K. Suzumara. *Handbook of Social Choice and Welfare.* North-Holland, Elsevier, 2002.

[2] M. Broxvall and P. Jonsson. Point algebras for temporal reasoning: Algorithms and complexity. *Artifcial Intelligence*, 149(2):179–220, 2003.

[3] L. Chen and P. Pu. Survey of preference elicitation methods. Technical Report IC/200467, Swiss Federal Institute of Technology in Lausanne (EPFL), 2004.

[4] Jan Chomicki. Semantic optimization of preference queries. In *CDB*, pages 133–148, 2004.

[5] V. Conitzer and T. Sandholm. Vote elicitation: Complexity and strategy-proofness. In *Proc. AAAI/IAAI 2002*, pages 392–397, 2002.

[6] W. Kießling. Preference queries with sv-semantics. In *11th International Conference on Management of Data (COMMAD 2005)*, pages 15–26. Computer Society of India, 2005.

[7] K. Konczak and J. Lang. Voting procedures with incomplete preferences. In *Proc. IJCAI-05 Multidisciplinary Workshop on Advances in Preference Handling*, 2005.

[8] J. Lang. Logical preference representation and combinatorial vote. *Annals of Mathematics and Artifi cial Intelligence*, 42(1):37–71, 2004.

[9] M. S. Pini, F. Rossi, K. Brent Venable, and T. Walsh. Computing possible and necessary winners from incomplete partially-ordered preferences. In *Poster paper in ECAI-06*, 2006.

[10] F. Rossi, K. B. Venable, and T. Walsh. mCP nets: representing and reasoning with preferences of multiple agents. In *AAAI-2004*, pages 322–327, 2004.