Voting Theory in Answer Set Programming

Kathrin Konczak

Institut für Informatik, Universität Potsdam, Postfach 90 03 27, D–14439 Potsdam
konczak@cs.uni-potsdam.de

Abstract. In this paper, we combine answer set programming with voting theory for the first time. For this, we use the voting procedures defined in [5], which allow winner determination under incomplete preference relations. We present the problem of scheduling a meeting for a research group as an application. In those scheduling problems, often no meetings are schedulable because of conflicting unavailabilities of the attendees or multiple dates for meetings are possible, where one has to respect the preferences of the attendees. In the first case, we provide a diagnostic model analyzing why no meeting was schedulable. In the second case, we use voting procedures for incomplete preference relations to determine preferred meetings.

1 Introduction

During the last decade, Answer Set Programming (ASP) [4] has become an increasingly acknowledged tool for knowledge representation and reasoning. A main advantage of ASP is that it is based on solid theoretical foundations, while being able to model commonsense reasoning in an arguably satisfactory way. The availability of efficient solvers has furthermore stimulated its use in practical applications, e.g. configuration, planning, scheduling, in recent years.

Automated group decision making is an important issue in AI: autonomous agents often have to agree on a common decision, and may for this reason apply *voting procedures*, which is one of the most common ways of making a collective choice. Voting procedures are usually defined for total preference relations. But naturally, agents will only express partial preferences among a set of given candidates. For this, we have focused in [5] on the application of a voting procedure when the voters' preferences are incomplete. There, we have defined lower and upper bounds for winners of voting procedures, which are called *possible* and *necessary* winners.

In this paper, we connect for the first time voting procedures with ASP. More precisely, we implement the voting procedures defined in [5] within ASP by using aggregate functions provided by the `DLV` system [3, 2]. Furthermore, we show the usefulness of these voting procedures within an application of ASP, namely the problem of scheduling a meeting for a research group. There, group members can express their preferences among a set of candidates, i.e. dates. Since usually one wants to express partial orders instead of total orders, e.g. one prefers to meet on Monday over Tuesday without given any requirements regarding time, the voting procedures for incomplete preference relations [5] seem to be appropriate for such kinds of problems.

Furthermore, we present a formalism for analyzing the scheduling problem in the case where no solutions are found. More precisely, we give a diagnostic model, following [7, 8], which gives explanations why no meeting was schedulable.

In Section 2, we recall basic background on answer set programming and voting theory. In Section 3, we present an encoding of voting procedures within answer set programming. In Section 4, we describe the problem of scheduling a meeting for a research group, which consists of several subgroups. In Section 4.1, we describe the basic problem, where a meeting is schedulable if from every subgroup a certain number of persons is available. If no meeting is schedulable due to unavailabilities of group members, we provide diagnostic reasoning for analyzing why no meeting was schedulable in Section 4.2. Lastly, in Section 4.3 we combine the basic scheduling algorithm with the voting procedures from [5] to determine preferred meetings, whenever multiple meetings are schedulable. We conclude with Section 5.

2 Background

A *logic program* is a finite set of rules such as $p_0 \leftarrow p_1, \ldots, p_m, not\ p_{m+1}, \ldots, not\ p_n$, where $n \geq m \geq 0$, and each p_i $(0 \leq i \leq n)$ is an *atom*. For such a rule r, we let $head(r)$ denote the *head*, p_0, of r and $body(r)$ the *body*, $\{p_1, \ldots, p_m,\ not\ p_{m+1}, \ldots, not\ p_n\}$, of r. Let $body^+(r) = \{p_1, \ldots, p_m\}$ and $body^-(r) = \{p_{m+1}, \ldots, p_n\}$. For a set of rules Π, we write $head(\Pi) = \{head(r) \mid r \in \Pi\}$. A program is *basic* if $body^-(r) = \emptyset$ for all its rules. The *reduct*, Π^X, of a program Π *relative to* a set X of atoms is defined by $\Pi^X = \{head(r) \leftarrow body^+(r) \mid r \in \Pi, body^-(r) \cap X = \emptyset\}$. A set of atoms X is *closed under* a basic program Π if for any $r \in \Pi$, $head(r) \in X$ if $body^+(r) \subseteq X$. The smallest set of atoms being closed under a basic program Π is denoted by $Cn(\Pi)$. Then, a set X of atoms is an *answer set* of a program Π if $Cn(\Pi^X) = X$. We use $AS(\Pi)$ for denoting the set of all answer sets of Π.

Let $X = \{x_1, \ldots, x_m\}$ be a finite set of *candidates* and $I = \{1, \ldots, n\}$ a finite set of *voters*. A *complete preference profile* is a collection $\mathcal{T} = \langle T_1, \ldots, T_n \rangle$ of total orders on X, where T_i represents the preferences of voter i. A *voting procedure* F maps every complete preference profile $\mathcal{T}$ to a nonempty subset of X, where $F(\mathcal{T})$ denotes the set of winners of $\mathcal{T}$ w.r.t. F. A *positional scoring procedure* is defined from a *scoring vector* $\boldsymbol{s} = (s_1, \ldots, s_m)$ of integers such that $s_1 \geq s_2 \geq \ldots \geq s_m$ and $s_1 > s_m$, e.g. the *Borda* procedure has the scoring vector $s_k = m - k$ for all $k = 1, \ldots, m$; and the *plurality* procedure the scoring vector $s_1 = 1$, and $s_k = 0$ for all $k > 1$. For every $x \in X$ and every $i \in I$, let $r(T_i, x) = \#\{y \mid y >_{T_i} x\} + 1$ be the rank of x in the complete order T_i; [1] then, we define the *scoring function* as $S(x, \mathcal{T}) = \sum_{i=1}^{n} s_{r(T_i,x)}$. The positional scoring rule $F_{\boldsymbol{s}}$ associated with a scoring vector $\boldsymbol{s}$ is defined by its set $F_{\boldsymbol{s}}(\mathcal{T}) = \{x \mid S(x, \mathcal{T}) \text{ is maximal }\}$ that is, the set of winning candidates for $\mathcal{T}$ with respect to $F_{\boldsymbol{s}}$ is the set of candidates in X maximizing the scoring function $S(., \mathcal{T})$. For an overview of voting theory see [6, 1].

An incomplete preference profile $\mathcal{R} = \langle R_1, \ldots, R_n \rangle$ is a collection of partial orders R_i on X, where R_i represents the preferences of voter i. In the following, R_i is often denoted as $\succeq_{R_i}$ or as $\succeq_i$. In [5], winners wrt incomplete preference profiles were defined by lower and upper bounds: $x \in X$ is a *necessary winner* for $\mathcal{R}$ (w.r.t. F) iff for all complete extensions $\mathcal{T}$ of $\mathcal{R}$ we have $x \in F(\mathcal{T})$, and $x \in X$ is a *possible winner* for $\mathcal{R}$ (w.r.t. F) iff there exists a complete extension $\mathcal{T}$ of $\mathcal{R}$ such that $x \in F(\mathcal{T})$. The *minimal* (resp. *maximal*) *rank* of a candidate x for a partial order R is the lowest (resp. highest) possible rank of x obtained when considering all complete extensions of R. These bounds can be computed in polynomial time, where the minimal rank of x is determined by the number of candidates which are higher ranked in the order, i.e. $rank_R^{min}(x) = \#\{y \mid y >_R x\} + 1$ and the maximal rank of x is determined by the number of lower ranked candidates, i.e. $rank_R^{max}(x) = m - \#\{y \mid x >_R y\}$.

For positional scoring procedures F_s like Borda and plurality, necessary and possible winners can be computed by considering the best and the worst case for values of scoring functions [5]: $S_{\mathcal{R}}^{min}(x) = \sum_{i=1}^{n} s_{rank_{R_i}^{max}(x)}$ and $S_{\mathcal{R}}^{max}(x) = \sum_{i=1}^{n} s_{rank_{R_i}^{min}(x)}$ Then, x is a necessary winner for $\mathcal{R}$ w.r.t. F_s iff $S_{\mathcal{R}}^{min}(x) \geq S_{\mathcal{R}}^{max}(y)$ holds for all $y \neq x$; and x is a possible winner for $\mathcal{R}$ w.r.t. F_s iff $S_{\mathcal{R}}^{max}(x) \geq S_{\mathcal{R}}^{min}(y)$ for all $y \neq x$.

A candidate x is a *Condorcet winner* for a complete profile $\mathcal{T} = \langle \succeq_1, \ldots, \succeq_n \rangle$ iff for all $y \neq x$, $\#\{i \mid x \succ_i y\} > \frac{n}{2}$. Analogously to positional scoring procedures, we can define possible and necessary Condorcet winners [5]. Let

$$N_{R_i}^{max}(x, y) = \begin{cases} +1 & \text{if not } (y \geq_i x); \\ -1 & \text{if } y >_i x \end{cases}, \quad N_{R_i}^{min}(x, y) = \begin{cases} +1 & \text{if } x >_i y; \\ -1 & \text{if not } (x \geq_i y) \end{cases},$$

$N_{\mathcal{R}}^{min}(x, y) = \sum_{i=1}^{n} N_{R_i}^{min}(x, y)$, and $N_{\mathcal{R}}^{max}(x, y) = \sum_{i=1}^{n} N_{R_i}^{max}(x, y)$. Then, x is a *necessary Condorcet winner* for $\mathcal{R}$ iff $\forall y \neq x,\ N_{\mathcal{R}}^{min}(x, y) > 0$, and x is a *possible Condorcet winner* for $\mathcal{R}$ iff $\forall y \neq x,\ N_{\mathcal{R}}^{max}(x, y) > 0$.

[1] The lower the rank, the more preferred is the candidate.

3 Encoding within Answer Set Programming

Next, we want to give an encoding of the voting procedures for incomplete preference profiles within ASP. For this, we use aggregate functions as provided by the DLV system [2, 3].

Let $X = \{x_1, \ldots, x_m\}$ be a set of candidates and $V = \{v_1, \ldots, v_n\}$ be a set of voters, where each voter has a partial preference profile over X. For representing the number of candidates, voters and preference relations, we use the following rules:

$$c(x_i) \leftarrow \text{ for all candidates } i = 1, \ldots, m \tag{1}$$
$$v(v_i) \leftarrow \text{ for all voters } i = 1, \ldots, n \tag{2}$$
$$pref(V, X, Y) \leftarrow \text{ for all voters } V \text{ preferring } X \text{ over } Y \tag{3}$$

Additionally, we need the following rules defining some basic concepts.

$$\#maxint = 50 \leftarrow \tag{4}$$
$$nbc(Nb) \leftarrow Nb = \#count\{M : c(M)\} \tag{5}$$
$$nbv(Nb) \leftarrow Nb = \#count\{V : v(V)\} \tag{6}$$
$$vtp(V) \leftarrow \text{ for } V \in \{borda, plurality, condorcet\} \tag{7}$$
$$sp(S) \leftarrow \text{ for } S \in \{borda, plurality\} \tag{8}$$

Rule (4) is DLV specific. The number of all candidates and voters is computed by rules (5)-(6). Rules (7)-(8) initialize the voting, respectively scoring, procedures. Since the computation of Borda, plurality, and Condorcet winners will be done independently from each other, one can initialize $vtp(.)$ and $sp(.)$ for all voting procedures.

Candidate x is a possible, respectively necessary, winner if there exists no candidate who "beats" x. The algorithm given in Section 2 are directly carried over.

$$possible(VP, X) \leftarrow c(X), vtp(VP), not\ no\_possible(VP, X) \tag{9}$$
$$no\_possible(VP, X) \leftarrow c(X), c(Y), X \neq Y, vtp(VP), sp(VP), \tag{10}$$
$$s\_max(VP, X, XS), s\_min(VP, Y, YS), XS < YS$$
$$no\_possible(condorcet, X) \leftarrow c(X), c(Y), Y \neq X, nbv(Nbv), \tag{11}$$
$$Z = \#count\{V : pref(V, Y, X), v(V)\}, Z1 = 2 * Z, Nbv \leq Z1$$
$$necessary(VP, X) \leftarrow c(X), vtp(VP), not\ no\_necessary(VP, X) \tag{12}$$
$$no\_necessary(VP, X) \leftarrow c(X), c(Y), X \neq Y, vtp(VP), sp(VP), \tag{13}$$
$$s\_min(VP, X, XS), s\_max(VP, Y, YS), XS < YS$$
$$no\_necessary(condorcet, X) \leftarrow c(X), c(Y), Y \neq X, nbv(Nbv), \tag{14}$$
$$Z = \#count\{V : pref(V, X, Y), v(V)\}, Z1 = 2 * Z, Z1 \leq Nbv$$

For computing the Borda score (cf. Section 2), we use rules (15)- (18).

$$borda\_smax(V, X, S) \leftarrow v(V), c(X), S{=}\#count\{Y : pref(V, X, Y), c(Y)\} \tag{15}$$
$$borda\_smin(V, X, S) \leftarrow v(V), c(X), nbc(M), \tag{16}$$
$$Rk = \#count\{Y : pref(V, Y, X), c(Y)\}, M = Rk + Rk1, Rk1 = S + 1$$
$$s\_min(borda, X, S) \leftarrow c(X), S = \#sum\{Sc, V : borda\_smax(V, X, Sc)\} \tag{17}$$
$$s\_max(borda, X, S) \leftarrow c(X), S = \#sum\{Sc, V : borda\_smin(V, X, Sc)\} \tag{18}$$

For the plurality score (cf. Section 2), we use rules (19)- (26).

$$rmin(V,X,R) \leftarrow v(V), c(X), R'=\#count\{Y : pref(V,Y,X), c(Y)\}, R=R'+1 \quad (19)$$
$$rmax(V,X,R) \leftarrow v(V), c(X), nbc(M), R1=\#count\{Y : pref(V,X,Y)\}, M = R1 + R \quad (20)$$
$$plural\_smin(V,X,1) \leftarrow v(V), c(X), rmin(V,X,1) \quad (21)$$
$$plural\_smin(V,X,0) \leftarrow v(V), c(X), rmin(V,X,R), R \neq 1 \quad (22)$$
$$plural\_smax(V,X,1) \leftarrow v(V), c(X), rmax(V,X,1) \quad (23)$$
$$plural\_smax(V,X,0) \leftarrow v(V), c(X), rmax(V,X,R), R \neq 1 \quad (24)$$
$$s\_min(plurality,X,S) \leftarrow c(X), S=\#count\{V : plural\_smax(V,X,1)\} \quad (25)$$
$$s\_max(plurality,X,S) \leftarrow c(X), S=\#count\{V : plural\_smin(V,X,1)\} \quad (26)$$

Thus, rules (1)- (26) represent the encoding of our voting problem within answer set programming. The logic program Π consisting of the rules (1)- (26) has exactly one answer set, which gives us the set of all possible and necessary winners.

Theorem 1. *Let be given a set of candidates X, a set of voters V, partial preference profiles for each voter over the set of candidates, and a voting procedure $VP \in \{Borda, plurality, Condorcet\}$. Then, the logic program Π, consisting of the rules (1)- (26), has exactly one answer set Y, where*

1. *the set $\{X : possible(VP,X) \in Y\}$ is the set of all possible winners wrt VP, and*
2. *the set $\{X : necessary(VP,X) \in Y\}$ is the set of all necessary winners wrt VP.*

Proof. (Sketch) Π has exactly one answer set, which can be seen by considering the well-founded semantics. Condition 1 and 2 follow by the semantics of aggregates functions.

4 Scheduling a meeting

In this section, we present an application of voting theory in scheduling problems. First, we describe the basic problem. Then, we include diagnostic reasoning and, afterwards, we present the integration of voting procedures into the scheduling algorithm.

4.1 Schedule a meeting for a research group

Next, we want to describe a solution for the problem of scheduling a meeting for a research group. We have given m possible times $d_1, \ldots, d_m$ for scheduling a meeting. The group consists of n subgroups $X_1, \ldots, X_n$. Each subgroup X_i has k_i members, where each member may have unavailabilities for certain dates. We want to schedule a meeting such that from every group at least k persons are available for that meeting.

Definition 1. *Let D be a set of dates, $X = X_1, \ldots, X_n$ be a set of research groups, where X_i has k_i members, and let NA be a set of unavailabilities $na(p,d)$ expressing that person $p \in X_i$ is unavailable at time $d \in D$. Furthermore, let k be the number of required persons from every sub-group, which should at least attend to a meeting.*

Then, we call $\mathcal{M} = \langle D, X, NA, k\rangle$ a meeting scheduling problem.

Furthermore, $m \in D$ is called meeting *whenever for all X_i, $1 \leq i \leq n$, we have $|\{p \in X_i : na(p,m) \notin NA\}| \geq k$.*

Let $\mathcal{M} = \langle T, X, NA, k\rangle$ be a meeting scheduling problem. Then, this problem is encoded within answer set programming by the following set of rules, where k in rule (31) presents the number of required persons from each research group.

$$g(X_i) \leftarrow \text{ for all subgroups } X_i \text{ of research group } X \quad (27)$$
$$p(P,X_i) \leftarrow \text{ for all members } P \text{ of subgroup } X_i \quad (28)$$
$$d(T) \leftarrow \text{ for all possible dates } T \quad (29)$$
$$na(P,T) \leftarrow \text{ for all } P \text{ and } T, \text{ where } P \text{ is unavailable for time } T \quad (30)$$
$$rnb(k) \leftarrow \quad (31)$$

$$a(P, G, D) \leftarrow not\ na(P, D), p(P, G), g(G), d(D) \tag{32}$$
$$present\_group(G, D) \leftarrow rnb(R), N = \#count\{P : a(P, G, D)\}, N \geq R, g(G), d(D) \tag{33}$$
$$unpresentgroup(D) \leftarrow d(D), g(G), not\ present\_group(G, D) \tag{34}$$
$$meeting(M) \leftarrow d(M), not\ unpresentgroup(M) \tag{35}$$
$$meet \leftarrow meeting(M), d(M) \tag{36}$$
$$\leftarrow not\ meet \tag{37}$$

Fig. 1. Meeting Scheduler

Furthermore, we include the logic program in Figure 1. Rule (32) expresses the status of availability of a person P from subgroup G for date D. Rules (33) and (34) express the status of subgroups. A subgroup G is *present* for date D whenever the number of available persons is higher than the number of required persons. Rule (35) generates possible meetings, where rules (36)–(37) ensure that at least one meeting is generated. Let $\Pi_{\mathcal{M}}$ be the logic program consisting of rules (27)–(37). $\Pi_{\mathcal{M}}$ has exactly one answer set if some meeting is schedulable, and $\Pi_{\mathcal{M}}$ has no answer set if no meeting is schedulable. If there exists an answer set X of $\Pi_{\mathcal{M}}$, then every date in the set $\{M : meeting(M) \in X\}$ represents a solution of our meeting scheduling problem.

Theorem 2. *Let $\mathcal{M} = \langle D, X, NA, k\rangle$ be a meeting scheduling problem and $\Pi_{\mathcal{M}}$ as described in rules (27)-(37). Then, one of the following holds:*

1. *$\Pi_{\mathcal{M}}$ has no answer set iff there exists no $m \in D$ such that for all X_i we have $|\{P \in X_i : P$ is available for $m\}| \geq k$; or*
2. *$\Pi_{\mathcal{M}}$ has exactly one answer set X, where $\{m : meeting(m) \in X\}$ contains all schedulable meetings for the scheduling problem $\mathcal{M}$.*

Example 1. At a university, we have a research group consisting of three subgroups: g_1, g_2 and g_3. Group g_1 has three members (p_1^1, p_2^1, p_3^1), g_2 has three members (p_1^2, p_2^2, p_3^2), and g_3 has only two members (p_1^3, p_2^3). They want to meet either on Monday or on Tuesday, where a meeting in the morning, in the afternoon, or in the evening is possible. Hence, we have 6 times: d_1 (Monday morning), d_2 (Monday afternoon), d_3 (Monday evening), d_4 (Tuesday morning), d_5 (Tuesday afternoon), and d_6 (Tuesday evening). Furthermore, we have the following unavailabilities: Persons p_1^1 is unavailable on Monday (d_1, d_2, d_3), p_2^3 is unavailable on Tuesday (d_4, d_4, d_6), p_3^2 is in the mornings unavailable (d_1, d_4), and p_2^2 is in the evenings unavailable (d_3, d_6). We want to schedule a meeting, where at least two persons from every subgroup can attend to it. With the unavailabilities at hand, we can schedule a meeting at times d_1, d_2, d_3 that is on Monday. If additionally person p_1^3 becomes unavailable on Monday, no meeting is schedulable.

4.2 Including diagnostic reasoning

With a huge number of subgroups of a research project, it often happens that no meeting is schedulable since the unavailabilities of persons are conflicting for every date. For this case, we develop a diagnostic model such that the reasons why no meeting is schedulable are determined by the diagnostic model. Furthermore, whenever a meeting is schedulable, the diagnostic model should give out all possible meetings. This idea of including diagnostic reasoning is closely related to the diagnostic model for the configuration of the Debian GNU/Linux system [7, 8]. There, the configuration problem gives suitable combinations of software packages which have to be installed in a Linux system. Software packages may interact in different ways, e.g. they are conflicting with each other or are requiring other software packages. In the case where no suitable configuration of software packages exists, the diagnostic model in [7, 8] determine an error set, a problem set, and an explanation set for analyzing why no configuration of the software packages is possible. The *error set* expresses why no configuration has been found, e.g. required software packages are

$$meet \leftarrow d(M), meeting(M) \tag{38}$$
$$meeting(M) \leftarrow d(M), not\ unpresentgroup(M) \tag{39}$$
$$unpresentgroup(D) \leftarrow d(D), g(G), not\ present\_group(G, D) \tag{40}$$
$$a(P, G, D) \leftarrow not\ na(P, D), p(P, G), g(G), d(D) \tag{41}$$
$$present\_group(G, D) \leftarrow rnb(R), N = \#count\{P : a(P, G, D)\}, N \geq R, g(G), d(D) \tag{42}$$
$$exists\_date \leftarrow d(D) \tag{43}$$
$$nodate \leftarrow not\ exists\_date \tag{44}$$
$$smallgroup(G) \leftarrow g(G), rnb(R), nbgroup(N, G), N < R \tag{45}$$
$$nbgroup(N, G) \leftarrow N = \#count\{P : p(P, G)\}, g(G) \tag{46}$$
$$unpresent(G, D) \leftarrow g(G), d(D), not\ present\_group(G, D), not\ meet \tag{47}$$
$$person\_unavailable(P, G, D) \leftarrow unpresent(G, D), na(P, D), p(P, G), g(G), d(D), not\ meet \tag{48}$$
$$incomplete \leftarrow d(M), not\ h(M), not\ meet \tag{49}$$
$$h(M) \leftarrow d(M), not\ unpresentgroup(M) \tag{50}$$

Fig. 2. Diagnostic Model of meeting scheduler

missing or selected packages are conflicting with each other. The *problem set* contains all software packages which are involved in a conflict. The *explanation set* points out why software packages causing errors are chosen to be in a configuration, e.g. the user has selected them or a package is required by another one.

In the following we want to carry over the diagnostic model for configuration problems to a meeting scheduling problem $\mathcal{M} = \langle D, X, NA, k\rangle$ for a research group. We have the following reasons, why no meeting is schedulable:

(R1) There are no dates for a meeting available.
(R2) For each date there exists at least one subgroup such that one of the following conditions holds:
- **(R2a)** the number of persons from that group is smaller than the number of required persons for that subgroup (without consideration of unavailabilities), or
- **(R2b)** there are too many persons unavailable for the date such that not enough persons can attend that meeting.

The logic program given in Figure 2 represents the encoding of the diagnostic model. Rules (38)–(42) are the same as in Figure 1 except for Rule (37), which is replaced by the following rules for the diagnostic output. Rule (43) and (44) handle reason (R1), whenever no date is available. Rule (45) and (46) handle reason (R2a), whenever one subgroup is smaller than the required number of persons. Rule (47)–(50) handle reason (R2b), where no meeting is schedulable since for every date at least one subgroup is not present due to unavailabilities of group members.

The logic program $\Pi^{\mathcal{D}}_{\mathcal{M}}$ consisting of rules (27)–(31) and (38)–(50) is called *diagnostic model* for the meeting scheduling problem $\mathcal{M}$. In contrast to [7], we define only the error set and the explanation set, since the problem set is needed in [7] to detect transitive relationships, which have no effect here.

Definition 2. *Let $\mathcal{M} = \langle D, X, NA, k\rangle$ be a meeting scheduling problem and $\Pi^{\mathcal{D}}_{\mathcal{M}}$ be the corresponding diagnostic model. A* diagnosis *is a triple $D = (X, E_X, R_X)$, where*

1. *X is an answer set of Π,*
2. *E_X is the error set*

$$E_X = \{nodate \in X\} \cup \{smallgroup(G) \in X\} \cup \{incomplete \in X\}$$

3. R_X is the explanation set

$$R_X = \{person\_unavailable(P, G, D) \in X\} \cup \{unpresent(G, D) \in X\}$$

The error set gives the reason why no meeting is schedulable. More precisely, it distinguishes the cases that no date time exists ($nodate \in E_X$), one subgroup is smaller than the required number of persons ($smallgroup(G) \in E_X$), or due to unavailabilities one subgroup is underrepresented ($incomplete \in E_X$). The explanation set contains information why no meeting is schedulable. Whenever $nodate \in E_X$ or $smallgroup(G) \in E_X$, we need no further explanations since these errors are self-explanatory. Whenever $incomplete \in E_X$, we add to the explanation set all subgroups G which are not present for each date time D, and corresponding to that all persons which are unavailable from these subgroups. Whenever a meeting is schedulable, there exists an answer set X, where the error set E_X and the explanation set R_X are empty. Otherwise, $\Pi^{\mathcal{D}}_{\mathcal{M}}$ has exactly one answer set, where the error set and the explanation sets are non-empty and are explaining why no meeting is schedulable

Theorem 3. *Let $\langle D, X, NA, k\rangle$ be a meeting scheduling problem and $\Pi^{\mathcal{D}}_{\mathcal{M}}$ be the corresponding diagnostic model. Then,*

1. *$m \in D$ is a meeting iff $\Pi^{\mathcal{D}}_{\mathcal{M}}$ has an answer set X, where $E_X = \emptyset, R_X = \emptyset$ and $meeting(m) \in X$,*
2. *there exists no meeting iff $\Pi^{\mathcal{D}}_{\mathcal{M}}$ has an answer set X such that $E_X \neq \emptyset$.*

Example 2. Let us reconsider Example 1. In the last case, where person p^3_1 became unavailable on Monday, no meeting was schedulable. Accordingly, our diagnostic model has one answer set X with a nonempty error set E_X and explanation set R_X. Our error set E_X contains *incomplete*, denoting the case that at least one subgroup can not attend to the meeting due to unavailabilities. Our explanations set R_X contains *unpresent(g3,D)* for all possible times $D \in \{d_1, d_2, d_3, d_4, d_5, d_6\}$. That is, no meeting was schedulable since group $g3$ is not present for each possible time. Furthermore, the explanation set R_X contains *person_unavailable(p^3_1,g3,D)* for $D = d_1, d_2, d_3$ and *person_unavailable(p^3_2,g3,D)* for $D = d_4, d_5, d_6$. That is, group $g3$ cannot attend to a meeting since p^3_1 is unavailable on Monday and p^3_2 is unavailable on Tuesday.

4.3 Selecting preferred meetings

In the case where multiple meetings are schedulable, we have to choose exactly one meeting out of the set of possible meeting times. In order to do so, every person expresses preferences for meeting times. Often, one can express preferences only as a partial order, e.g. one prefers a meeting in the morning over a meeting in the afternoon, instead of total orders. Hence, we use the voting procedures from Section 3 to determine upper and lower bounds for preferred meetings wrt different voting strategies.

Each person of our research group provides his preferred dates. That is, each voter p^i_k has a partial order $<^i_k$ among the set of dates. Since in general not all dates are schedulable as meetings, we have to restrict the partial orders to the set of schedulable meetings. Furthermore, we assume that each person expresses only preference relations on dates, where the person is available.

Let $\mathcal{M} = \langle D, X, NA, k\rangle$ be a meeting scheduling problem. For each person $p^i_j, 1 \leq j \leq k_i$ from subgroup X_i $(i = 1, \ldots, n)$, we have given a partial preference relation $<^i_j \subseteq D \times D$ for the set of all dates $D = d_1, \ldots, d_m$. Note that each voter expresses only preference relations on the set of for him available dates. That is, $<^i_j \subseteq D_a \times D_a$ where $D_a = \{d \in D \mid na(p^i_j, d) \notin NA\}$. We call $\mathcal{M}^< = \langle D, X, NA, k, <\rangle$ a *preferred meeting scheduling problem*, where $<$ comprises the preference relations given above. We have to restrict the partial orders to the set of all schedulable meetings. So, let $M \subseteq D$ be the set of all schedulable meetings according to the requirements given in Section 4.1. Then, we let $\prec^i_j \subseteq M \times M$ such that $d \prec^i_j d'$ holds if $d <^i_j d'$ holds.

We combine the logic programs from the basic meeting scheduling problem in Figure 1 and the encoding of voting procedures within ASP (Rules (1)- (26)) from Section 3 as follows: We define

$\Pi^{<}_{\mathcal{M}}$ as the union of the rules (27)-(37), rules (4)-(26), and the following rules, which replace rules (1)-(3):

$$c(M) \leftarrow meeting(M), d(M) \tag{51}$$

$$v(P) \leftarrow p(P,G), not\ na(P,M), meeting(M), g(G) \tag{52}$$

$$pref(P^i_j, X, Y) \leftarrow \text{where } Y \prec^i_j X \text{ holds for } X, Y \text{ being meetings} \tag{53}$$

With this logic program, we can compute *possible* and *necessary meetings*.

Theorem 4. *Let $\mathcal{M}^{<} = \langle D, X, NA, k, < \rangle$ be a preferred meeting scheduling problem, $\Pi^{<}_{\mathcal{M}}$ the corresponding logic program, and let $VP \in \{Borda, plurality, Condorcet\}$ be a voting procedure. Then, one of the following holds*

1. *$\Pi^{<}_{\mathcal{M}}$ has either no answer set, expressing that no meetings are schedulable, or*
2. *$\Pi^{<}_{\mathcal{M}}$ has exactly one answer set Y, where*
   (a) *the set $\{X : possible(VP, X) \in Y\}$ is the set of all possible meetings wrt voting procedure VP, and*
   (b) *the set $\{X : necessary(VP, X) \in Y\}$ is the set of all necessary meetings wrt voting procedure VP.*

Example 3. Let us reconsider Example 1. Assume that no person of the research group has unavailabilities. That is, all dates are schedulable as meetings. Consider the following preference relations: group g_1 prefers Monday morning over all other dates ($d_1 > d_i, i = 2, ..., 6$); g_2 prefers Monday over Tuesday and the afternoon and evening over the morning ($d_2 > d_1 > d_5 > d_4, d_3 > d_1 > d_6 > d_4$); and g_3 prefers Monday morning over Monday afternoon and Monday evening and Monday over Tuesday ($d_1 > d_2, d_1 > d_3$ and $d_i > d_j$ for $i = 2, 3$ and $j = 4, 5, 6$).

For the Borda voting procedure, we get d_1, d_2, d_3 as possible winners and no necessary winners. For the plurality and for the Condorcet procedure, we get d_1 as possible and as necessary winner. Hence, a meeting should be scheduled on Monday morning.

5 Conclusions and Further Work

We have linked voting theory to answer set programming for the first time. We have considered the voting procedures for incomplete preference profiles defined in [5] and have presented in Section 3 an embedding in answer set programming. Furthermore, we have presented the meeting scheduling problem for a research group, where we have integrated these voting procedures for computing preferred meetings. First, we have defined the basic problem (Section 4.1), then we have included diagnostic reasoning in Section 4.2. Whenever no meetings are schedulable, the diagnostic model determines why no solution to a problem exists. In Section 4.3, we have integrated voting procedures into our meeting scheduling problem. Voters, members of the research group, can express their preferences among a set of dates, and the voting procedures provide possible and necessary preferred meetings. Hence, we have shown the usefulness of voting procedures for incomplete preference profiles within an application. The example presented in Section 4.1 is in polynomial time solvable. Although ASP is able to handle complexer problems (e.g. NP-complete ones), we have taken a polynomial problem to demonstrate how efficient voting procedures can be integrated into logic programming. In further work, we want to integrate voting procedures within more complex, NP-complete, timetabling problems. The voting procedure for incomplete preference relations are used as a "filter" for determing preferred solutions. Since they are computable in polynomial time [5], the complexity of the underlying problem does not increase.

Acknowledgements

The author was supported by the German Science Foundation (DFG) under grant SCHA 550/6-4, TP C and by the EC under project IST-2001-37004 WASP.

References

1. S. Brams and P. Fishburn. *Handbook of Social Choice and Welfare*, volume 1, chapter Voting procedures. Elsevier, 2003.
2. T. Dell'Armi, W. Faber, G. Ielpa, N. Leone, and G. Pfeifer. Aggregate functions in disjunctive logic programming: Semantics, complexity, and implementation in dlv. In G. Gottlob and T. Walsh, editors, *IJCAI-03*, pages 847–852. Morgan Kaufmann, 2003.
3. `DLV. http://www.dbai.tuwien.ac.at/proj/dlv/`.
4. M. Gelfond and V. Lifschitz. The stable model semantics for logic programming. In *Proceedings of the International Conference on Logic Programming*, pages 1070–1080. The MIT Press, 1988.
5. K. Konczak and J. Lang. Voting procedures with incomplete preferences. In R. Brafman and U. Junker, editors, *IJCAI-05 Workshop on Advances in Preference Handling*, pages 124–129, 2005.
6. J. Lang. Logical preference representation and combinatorial vote. *Annals of Mathematics and Artificial Intelligence*, 42(1):37–71, 2004.
7. T. Syrjänen. Including diagnostic information in configuration models. In J. Lloyd, V. Dahl, U. Furbach, M. Kerber, K. Lau, C. Palamidessi, L. Pereira, Y. Sagiv, and P. Stuckey, editors, *First International Conference on Computational Logic (CL 2000)*, volume 1861 of *Lecture Notes in Computer Science*, pages 837–851. Springer, 2000.
8. Tommi Syrjänen. A rule-based formal model for software configuration. Research Report A55, Helsinki University of Technology, Laboratory for Theoretical Computer Science, 1999.