State-Space Reduction through Preference Modeling*

Radosław Klimek, Igor Wojnicki, and Sebastian Ernst

AGH University of Science and Technology
Department of Applied Computer Science
Al. Mickiewicza 30, 30-059 Kraków, Poland
{rklimek,wojnicki,ernst}@agh.edu.pl

Abstract. Automated planning for numerous co-existing agents, with uncertainty caused by various levels of their predictability, observability and autonomy, is a complex task. One of the most significant issues is related to explosion of the state space. This paper presents a formal framework which can be used to model such systems and proposes the use of formally-modeled agents' preferences as a way of reducing the number of states. A detailed description of preference modeling is provided, and the approach is evaluated by examples.

1 Introduction

Automated planning is an active research field, with applications ranging from motion planning [1], through resource allocation and scheduling, to coordination of autonomous agents [2]. This paper presents a formal framework which can be used to model sophisticated systems, featuring heterogenous entities (agents), characterized by varying levels of autonomy, predictability and observability, and proposes the use of preferences to reduce the size of the state space.

The paper is organized as follows. Section 2 provides a brief introduction to automated planning and the state-space representation of planning problems. Section 3 presents the framework, indicating certain and uncertain knowledge elements, and provides intuition how this model can be mapped to real-world cases. Section 4 introduces a formal tool which can be used to model preferences (or agent constraints) using temporal logic, and Section 5 provides an example how formally represented preferences can be used to reduce the state space.

2 Motivation and State-of-the-Art

The most general, intuitive definition of planning is that it is the *reasoning part of acting* [3]. Planning can be performed by people, either implicitly or explicitly. Automated planning is a branch of AI which is concerned with computation and execution of plans by machines.

* This work is supported by the Polish National Science Centre (NCN) grant 2011/01/D/ST6/06146.

L. Rutkowski et al. (Eds.): ICAISC 2013, Part II, LNAI 7895, pp. 363–374, 2013.

A common conceptual model for planning is a *state-transition system*, also called *discrete-event system*. It can be defined as a 4-tuple $\Sigma = (S, A, E, \gamma)$ [3,4], where:

- $S = \{s_1, s_2, \ldots\}$ is a set of *states*,
- $A = \{a_1, a_2, \ldots\}$ is a set of *actions*,
- $E = \{e_1, e_2, \ldots\}$ is a set of *events*,
- $\gamma : S \times A \times E \to 2^S$ is a state-transition function.

This definition is often used together with a graph model, where states $s \in S$ are nodes, and state transitions (given as pairs $(a, e), a \in A, e \in E$) are directed edges. The semantic difference between actions and events is that actions are *applicable* to states by the plan executor (if $\gamma(s, a) \neq \emptyset$), while events are *contingent* – they might occur due to the system's characteristics, and every event e which takes the system from state s to state s' must have a corresponding transition function $\gamma(s, e) = \{s'\}$. In a typical instance of a planning problem, the system is in some initial state $s_I \in S$, and the goal is to take it to one of the goal states $s_G \in \{s_{G1}, s_{G2}, \ldots\} \subset S$.

Based on that definition, derivation of a plan consists in finding a sequence of state transitions which take the system form the initial state to one of the goal states. Numerous methods can be used here – either "blind" (uninformed) ones, which do not take the characteristics of the system into account [4, sec. 3.4] or informed (heuristic) methods, which are especially useful for large state spaces [5]. Determination of heuristics for *domain-independent planning* is a problem which has recently received a lot of attention [6,7,8,9].

The size of the state space and the complexity of the planning process can grow rapidly, due to several reasons:

1. **Partially predictable action (and event) results.** If it is unclear which state the system will be in after an action is taken or an event occurs, the planner will have to consider all possible outcomes, which may result of a combinatorial explosion of the search tree.
2. **Partially determinable world state.** If the world state cannot be fully observed, the planner needs to assume all possible states which match the observations.
3. **Multiple agents.** The size of the state space grows exponentially with the number of entites (agents), as the local state of an agent can be combined with almost all states of every other agent. Planing for such a case (multi-body) is not different than for a single agent though[4].
4. **Multi-variant plans.** Sometimes, it may be preferable to provide multiple possible actions to the agent and let it choose the most favorable option. This is true especially in situations, where the planner's observations of the system state are less detailed than (local) observations made by the agents.

In needs to be noted, that the state space is not a Cartesian product of all possible states. Some states can be unreachable due to world constraints (obstacles) or agent constraints restricting its behavior. Defining these constrains decreases planning process complexity.

3 Considered World

This section presents the general model of a world of coexisting entities and their supervisor. The entities differ with regard to the following parameters:

- *controllability* – the extent to which the supervisor may influence the entities; inversely proportional to the entities' *autonomy*,
- *predictability* – certainty that the entity will act according to the prediction, either due to its intents or its abilities to fulfill the orders,
- *observability* – the ability of the supervisor to observe the actions of an entity, whether autonomous or performed to fulfill an order.

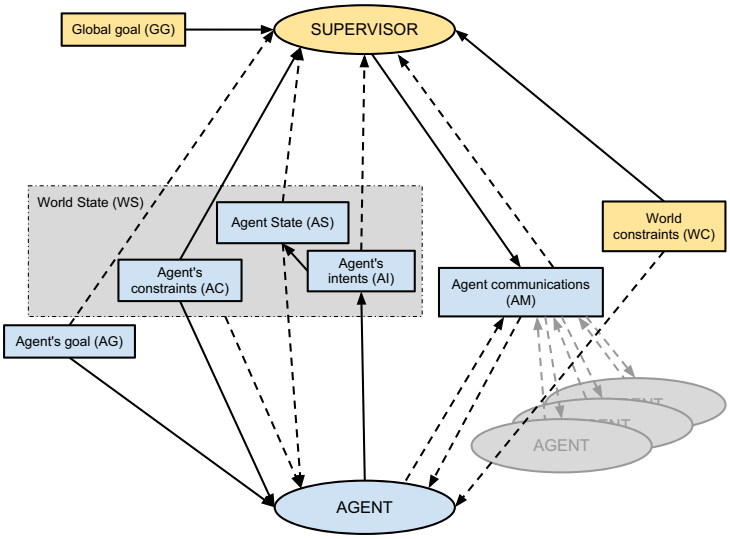


Fig. 1. Knowledge in the world model. Solid lines indicate certain knowledge, while dashed lines indicate knowledge that is subject to an arbitrary level of uncertainty.

An overview of the knowledge existing in the world and its availability to particular entities has been presented in Figure 1. The knowledge elements are as follows:

Global Goal (GG). The desired state of the world, known to the *Supervisor*, irrelevant to *Agents*.

World State (WS). State of other agents (AC, AS, AI) as perceived or inferred by the *Agent* under consideration.

Agent's State (AS). Current state of an individual agent, might be known to the *Agent* or *Supervisor*. It is an effect of the agent's actions, which result from the Agent's Intents (AI).

Agent Communications (AM). May occur:

- between the *Supervisor* and the *Agent*, typically including *orders* (downstream) and *order fulfilment acknowledgements* (upstream),
- between individual agents, typically including negotiations and knowledge broadcast.

Agent's Constraints (AC). Limitations of an *Agent* with regard to actions it can perform; known to the *Agent* and the *Supervisor*.

Agent's Goal (AG). Goal of an individual *Agent*. Known to the *Agent*, might be known to the *Supervisor*.

Agent's Intents (AI). Generated by the *Agent* based on its goal (AG), received orders (subset of AM) and state of other agents (WS).

World Constraints (WC). Known to the *Supervisor*, may be known (e.g. observed) by the *Agent*.

Orders, received within AM, are usually plans, often multi-variant ones, i.e. consisting of more than one path leading to the destination. Since the proposed model is generic, potential applications regard many different domains, both abstract and very particular.

For more abstract solutions, multi-agent systems are targeted. These include problem decomposition and agent scheduling. The issue of problem decomposition regards subdividing the problem into manageable sub-problems and ascribing them to given agents. Depending on the nature of the problem, it may be required to perform subdivision in a planned way. Synthesizing multiple plans and choosing the optimal or suboptimal one poses a challenge. Similarly, while managing multiple agents towards reaching a global goal, there is a need to plan their actions. In such case, the aforementioned issues of controllability, observability and predictability come to play. Furthermore, in many cases, such planning should also be multi-variant, especially if communication with some agents is not reliable.

There are many similarities between pure software multi-agent systems and real-world applications. One of the cases to be considered here is fleet management (in terms of a supply chain, consisting of trucks, trains, cargo ships, etc.). While there might be a well defined goal in existence, which formulates optimization criteria from the perspective of managing shipments, goals of individual agents can fluctuate in time and space. Such fluctuations might be caused by a wide range of factors, including technical issues with vessels, natural disasters, communication interruptions, or human factor issues.

Search and rescue operations can also be considered here. They form a complex system which includes information about the environment under consideration, rescue team members, supporting hardware (such as autonomous search and mapping robots), domain experts, other personnel (e.g. volunteers), commanding centers, etc. Since such operations are carried out under extreme variability of factors regarding the system components, it is not safe to assemble a single plan. Instead, multiple plans for such cooperative activities should be established to compensate for lack of controllability, observability or predictability.

Yet another application are team sports, both these performed by humans (e.g. basketball) or robots (such as RoboCup soccer). The ultimate goal here is

to win. There are multiple agents (the players) performing actions and carrying out orders and suggestions and the couch, or programmer, as the supervisor. Depending whether it is a simulated game, a RoboCup competition or a real one, the proposed World still holds and expresses the system.

4 Preference Modeling

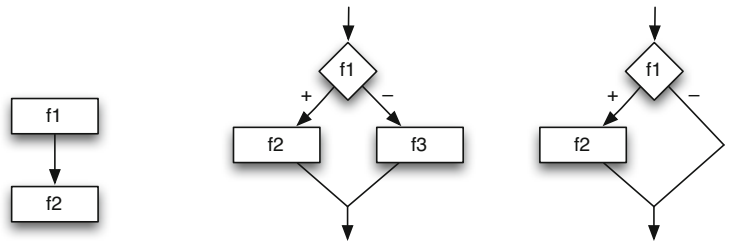
Preference modeling and choice prediction is crucial in the world of agents. *Preference* is an act of selecting something over others possibilities of choice. It is equivalent to the principle of giving advantages to some options over others. It also enables customization of the agents' behavior and preferences could be used to choose from a multi-variant plan. Decisions of agents require a mechanism for representing and reasoning about the possible consequences of their choices. Logical models seem especially appropriate for representing preference models and their exploration. Flexible models should provide the potential for interpreting preference choices in a way that does not depend on the underlying utility model.

Preference modeling and preference models need formalization and are discussed in some works, e.g. work by Öztürk et al. [10]. The preference models might be constructed using fuzzy sets, classical logic and many-valued logics. Classical logic, particularly rule-based systems, are especially popular, c.f. [11]. Non-classical logics, especially temporal logic, are less popular here. However, let us note that temporal logic is a well established formalism for describing system reactiveness. On the other hand, many applications are characterized by reactivity and flexibility in adapting to changes on the user side. These changes may result from recognized and predefined preferences which are applied in practice. The variability and change in valuation of logical statements over time flows are difficult to describe in classical logic. This is a reason important enough to include temporal logic in the considerations for preference models. Temporal logic creates new possibilities for analysis of preferences by going beyond the static world of classical logic. It also allows to illustrate the dynamic aspect of preferences which describe situations of preference valuations that vary over flows of time. The temporal approach seems to be underestimated in preference modeling. After building a preference model in temporal logic, one can analyze it using a deductive approach. The goal is searching for contradictions, if any, in a model. It is also possible to inference about the correctness of preference objectives.

Temporal Logic TL is a formalism, e.g. [12], which has strong application for specification and verification of models. It exists in many varieties; however, considerations in this paper are limited to the *Linear Temporal Logic* LTL, i.e. logic for which the time structure is considered linear. Considerations in this paper are limited to the *smallest temporal logic*, e.g. [13]. This logic is considered a classical propositional calculus' extension of axiom $\Box(P \Rightarrow Q) \Rightarrow (\Box P \Rightarrow \Box Q)$ and inference rule $\frac{\vdash P}{\vdash \Box P}$. The following formulas may be considered as significant examples of the minimal temporal logic: $LowVoltageDetection \Rightarrow \Diamond Alarm$,

$\Box(Trigger \Rightarrow \Diamond Action)$, $\Diamond Live$, $\Box\neg(BadEvent)$ or $\Box\neg(Event1 \wedge (Event2 \vee Event3))$, etc.

Semantic tableaux is a decision procedure for checking formula satisfiability. The method is well known in classical logic, but it can also be applied in modal and temporal logics [14]. The method is based on formula decompositions. At each step of a well-defined procedure, formulas are decomposed and have fewer components since logical connectives are removed. Finding a contradiction in all branches of the decomposition tree means there are no valuations that satisfy a formula placed in the root. When all branches of the tree have contradictions, it means that the inference tree is *closed*. If the negation of the initial formula is placed in the root, this leads to the statement that the initial formula is true. This method has some advantages over the traditional axiomatic approach. In the classical reasoning approach, starting from axioms, longer and more complicated formulas are generated and derived. Formulas become longer and longer step by step, and only one of them will lead to the verified formula. The method of semantic tableaux is characterized by a reversed strategy. The inference structure is represented by a tree and not by a sequence of formulas. Expansion of any tree branch may be halted after finding a contradiction. In addition, the method provides, through so-called *open* branches of the semantic tree, information about the source of an error, if one is found; that is another and very important advantage of the method. The work [15] shows an example of the truth tree of the semantic tableaux method for minimal temporal logic.



(a) Sequence (b) Branch (c) SimpleBranch

Fig. 2. Patterns for preferences

The proposed preference model is based on preference patterns. A *pattern* is a predefined solution for a special context of preference issues. Preference patterns are shown in Fig. 2 and are introduced in the work [15]. Every preference model, perhaps elicited during the requirements engineering phase, consists only of these patterns. Preference patterns can be nested and form complex models. Preferences are in the form of logical rules expressed in temporal logic. Suppose that well-formed and syntactically correct temporal logic formulas are already defined, c.f. [12].

Patterns constitute a kind of primitives. They are indicated as $pat()$, where pat is a name of a given pattern, and their parameters, if any, are included in parentheses. The following three patterns are considered: *Branch*, *SimpleBranch* and *Sequence*. They constitute a kind of illustration of the if-then scheme. Pattern nesting enables projection of a multi-stage decision-making. Say, a *basic set of patterns* Σ is a set of temporal logic formulas describing properties of a pattern. Thus, a set of three patterns, i.e. $\Sigma = \{Branch, SimpleBranch, Sequence\}$, is considered. Now, let us define *temporal properties* $\Pi(\Sigma)$ over predefined patterns Σ. Hence, set $Branch(f_1, f_2, f_3) = \{c(f_1) \Rightarrow \Diamond f_2 \wedge \neg\Diamond f_3, \neg c(f_1) \Rightarrow \neg\Diamond f_2 \wedge \Diamond f_3, \Box\neg(f_1 \wedge (f_2 \vee f_3))\}$ describes properties of the Branch pattern and set $SimpleBranch(f_1, f_2) = \{c(f_1) \Rightarrow \Diamond f_2, \neg c(f_1) \Rightarrow \neg\Diamond f_2, \Box\neg(f_1 \wedge f_2)\}$ the SimpleBranch pattern. Set $Sequence(f_1, f_2) = \{f_1 \Rightarrow \Diamond f_2, \Box\neg(f_1 \wedge f_2)\}$ defines the Sequence pattern. Formulas that constitute set for a patterns describe both liveness and safety property of a pattern. Formulas f_1, f_2 etc. are atomic formulas for a pattern. They are a kind of formal arguments for a pattern. $\Diamond f$ means that sometime activity f is completed, i.e. when the token left the activity, i.e. when the falling edge (or the negative edge) of the activity is transited. $c(f)$ means that the logical condition associated with activity f has been evaluated and is satisfied. This logical condition is satisfied when the falling edge is transited. There is a standard taxonomy of system properties, i.e. liveness and safety, which are adapted here to the field of preferences:

- *liveness* means that some preferences might be achieved, if desired, in the whole preference model, e.g. $p_1 \Rightarrow \Diamond p_5$ or $\Diamond p_4$;
- *safety* means that some preferences, if any, perhaps a logical combination of a subset of the entire preference world, are avoided; e.g. $\Box\neg(p_2 \wedge \neg p_6 \wedge p_{10})$.

where $\{p_1, ..., p_{10}\}$ is a whole identified preference world.

The entire preference model can be written in the form of logical expressions in order to write preferences in a concise and literal notation. The *logical expression* W_L is a structure created using the following rules:

- every elementary set $pat(a_i)$, where $i > 0$ and every a_i is an atomic formula, is a logical expression,
- every $pat(A_i)$, where $i > 0$ and every A_i is either (a) an atomic formula, or (b) a logical expression $pat()$, is also a logical expression.

Any logical expression may represent an arbitrary structure of patterns and an example of this are the following expressions: $Branch(a, SimpleBranch(f, g), c)$ and $Sequence(Branch(a, b, c), SimpleBranch(d, e))$. In the first case, the combination (and nesting) of two branched patterns is considered, a and f are conditions in this expression. In the second case, the sequence of two branched patterns is considered. Individual preferences expressed as a logical expression may belong to a set of preferences R, i.e. $R = \{r_1, r_2, ...r_n\}$, where every r_i is a preference which is expressed as a single logical expression.

When building a logical model for preferences, two important aspects of a logical system can be analyzed: (1) semantic contradiction of a model, or (2)

correctness of the model due to some properties. Formal verification of properties for a preference model leads to the analysis of the formula $s_1 \wedge \ldots \wedge s_n \Rightarrow Q$, where Q is a desired property for the preference model $\{r_1, r_2, ...r_m\}$.

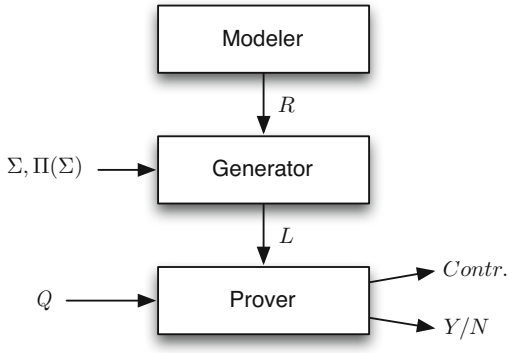


Fig. 3. Deduction system for preference models

The architecture of a system for automatic inference on preference models is proposed in Fig. 3. The Modeler module allows to prepare a preference model using preference patterns shown in Fig. 2. The output of the Modeler are preference models R expressed as logical expressions. The next module is the Generator module. The inputs of the Generator are logical expressions R and a predefined set of basic preference patterns Σ, together with their predefined temporal properties $\Pi(\Sigma)$. The output is a logical specification L understood as a set of temporal logic formulas. The sketch of the generating algorithm is the following:

1. At the beginning, the logical specification is empty, i.e. $L = \emptyset$;
2. The most nested pattern or patterns are processed first, then, less nested patterns are processed one by one, i.e. patterns that are located more towards the outside;
3. If the currently analyzed pattern consists only of atomic formulas, the logical specification is extended, by summing sets, by formulas linked to the type of the analyzed pattern $pat()$, i.e. $L = L \cup pat()$;
4. If any argument is a pattern itself, then the logical disjunction of all its arguments, including nested arguments, is substituted in place of the pattern.

The above algorithm refers to similar ideas in work [16]. Let us supplement the algorithm by some examples. The example for the step 3: $Seq(p, q)$, gives $L = \{p \Rightarrow \Diamond q, \Box\neg(p \wedge q)\}$ and $Branch(a, b, c)$ gives $L = \{c(a) \Rightarrow \Diamond b \wedge \neg\Diamond c, \neg c(a) \Rightarrow \neg\Diamond b \wedge \Diamond c, \Box\neg(a \wedge (b \vee c))\}$. The example for the step 4: $Sequence(Branch(a, b, c), d)$ leads to $L = \{c(a) \Rightarrow \Diamond b \wedge \neg\Diamond c, \neg c(a) \Rightarrow \neg\Diamond b \wedge \Diamond c, \Box\neg(a \wedge (b \vee c)), (a \vee b \vee c) \Rightarrow \Diamond d, \Box\neg((a \vee b \vee c) \wedge d)\}$.

The Prover module works using the semantic tableaux method described above. The inputs for the Prover include a logical specification $L = \{s_1, ..., s_n\}$ and a query Q which might be a simple temporal logic formula expressing the desired property for the preference model. (This formula can be introduced using a simple text editor.) The Prover provides two kinds of answers which originate from two following cases:

1. Semantic contradiction, i.e. analysis of the formula:

$$s_1 \wedge \ldots \wedge s_n \tag{1}$$

2. Correctness of the model due to some properties, i.e. the formal verification of the formula:

$$s_1 \wedge \ldots \wedge s_n \Rightarrow Q \tag{2}$$

In the case of contradiction, formula 1 is placed in the root of the reference tree and the information about the semantic contradiction is produced. In the case of correctness, the negation of the formula 2 is placed in the root of the reference tree and the *Yes/No* output is produced.

Let us consider a simple example for the house domain. Preferences can be aggregated over groups of two objectives. The first one is the monitoring and alarm system. When danger is detected, a security group or the fire brigade is called depending on the type of danger; however, in both cases the police is called. After modeling preferences, one can obtain the following logical expression:

$$\begin{aligned} r_1 = Sequence(Danger, Sequence(Branch(\\ Intrusion, SecurityGroup, SimpleBranch(\\ Fire(FireBrigade)))), Police) \end{aligned} \tag{3}$$

The second objective is related to the method of house heating. Let us consider two different intervals of hours. Nights have lower range of temperatures in contrast to a higher range of temperatures for mornings and afternoons. Finally, one can obtain:

$$\begin{aligned} r_2 = Branch(Weekend, Branch(Night1, Lower, \\ Higher), Branch(Night2, Lower, Higher)) \end{aligned} \tag{4}$$

Thus, $P = \{Danger, Intrusion, ..., Weekend, Night1, ...\}$ are atomic preferences which were identified when preparing preference model. Next, $R = \{r_1, r_2\}$ are preferences considered as a set of logical expressions. Finally, the logical specification L is generated using the algorithms described above with two inputs, i.e. preferences R and the set of predefined patterns Σ together with related temporal properties $\Pi(\Sigma)$.

5 Examples of State-Space Reduction Cases

Let us consider a fully observable, predictable, and controllable world with given constraints, the Grid World. Knowledge regarding it is considered as follows, particular world components given in parentheses refer to Fig. 1.

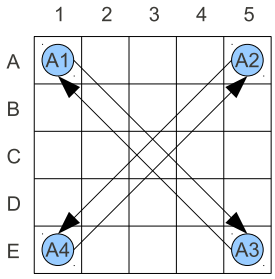


Fig. 4. Initial state of the proposed Grid World

The world is two-dimensional, based on a five by five grid (WC). The agents, denoted as A_n can move from one location to another orthogonally only, in accordance with the von Neumann neighborhood: north, south, east and west (WC). No more than one agent is allowed at a single location. There are four agents initially located in the corners (AS) (see Fig. 4). There is no inter-agent communication (AM). Each agent's goal it to get to the diagonally opposite corner (AG); A_1 to $(E, 5)$, A_2 to $(E, 1)$, A_3 to $(A, 1)$, A_4 to $(A, 5)$. Global goal (GG) is to guide all the agents to their goals corners. Since the world is fully controllable, the agents' intents comply with orders given by the supervisor (AM).

Assuming the above, a state consists of information about agent locations, thus it is a tuple: $s = (x_1, y_1, x_2, y_2, x_3, y_3, x_4, y_4)$. Assuming that each agent can be at any location, having 5×5 grid, cardinality of S, number of states, is given as:

$$|S_1| = (5 * 5)^4 = 390625.$$

Applying the world constraints (no two agents sharing the same location) reduces number of states slightly, having a product of arithmetic progression:

$$|S_2| = (5 * 5) * (5 * 5 - 1) * (5 * 5 - 2) * (5 * 5 - 3) = 303600.$$

Let us now introduce preferences in order to demonstrate how they can reduce the state space. Some agents suffer from *demophobia*[1], i.e. avoid being in a crowd. The consequence of this is an appropriate model of preferences. However, let us consider the whole world of atomic preferences

$$P = \{Go(N), Go(E), Go(S), Go(W), Crowd(d), ...\} \tag{5}$$

where $Go(N)$ means the agent preference to go North (or East, South, West), etc. Function $Crowd(d)$ is a Boolean function that tests the presence of the crowd, where d means a direction North, East, South or West and express the desired direction for agent.

Let us express the following preference model for an agent. He examines all directions, and only as a last resort go to fourth direction. Suppose that d is the current direction which expresses the agent desire and its will.

[1] A fear of crowds, masses, or people.

$$r_1 = Branch(Crowd(d), Branch(Crowd(d+1), \\ Branch(Crowd(d+2), Go(d+3), Go(d+2)), Go(d+1)), Go(d)) \quad (6)$$

After examining all the directions, we go in the last possible one.

If we take the preference described above into account, the state space will be reduced to at most $|S_3| = (5*5)*(5*5-3)*(5*5-6)*(5*5-9) = 167200$. Please note that the number above is the upper bound of the number of states, based on the assumption that the presence of an agent renders at most three of its neighboring cells unfavorable for other agents, in order to simplify calculations.

More detailed analysis of all possible combinations has revealed that the actual number of states fulfilling the above preference is 120576.[2] This indicates significant state reduction compared to $|S_2|$ ($\approx 60\%$) and $|S_1|$ ($\approx 69\%$). As the complexity and number of preferences is arbitrary, the deduction system (see Section 4) can use them to identify states which are unreachable and eliminate them, thus reducing the state space.

6 Conclusions and Future Work

Planning for multiple entities causes an explosion of the number of states in the state space. Variations of agent controllability, observability and predictability add to the problem by introducing belief states and states which would not otherwise be considered by the supervisor.

The number of states can be reduced by identifying unreachable states, which in turn can be eliminated from the state space. This can be achieved by defining and analyzing agent's preferences. As shown in Section 5, some preferences can influence the number of states in terms of their reduction.

Moreover, even if states cannot be eliminated from the state space, formally defined preferences can be useful if the agent receives multi-variant instructions and needs to decide which path to follow.

Other preferences may not eliminate the possible system states, but can influence the transitions between them. An intuitive example could be a preference of an agent not to turn left. While the number of states would not necessarily be reduced (as one left turn could be compensated by multiple right turns), they can be used by the planner to prune the search tree, thus reducing its branching factor.

Future work includes perfecting identification of unreachable states and improvement of the formalism used to model preferences, as well as research related to preference-driven search or planning algorithms and extraction of heuristics from preference sets.

References

1. Turek, W., Marcjan, R., Cetnarowicz, K.: Software agent systems for improving performance of multi-robot groups. Fundamenta Informaticae 112(1), 103–117 (2011)

[2] Value obtained by means of numerical analysis.

2. Cetnarowicz, K.: From algorithm to agent. In: Allen, G., Nabrzyski, J., Seidel, E., van Albada, G.D., Dongarra, J., Sloot, P.M.A. (eds.) ICCS 2009, Part II. LNCS, vol. 5545, pp. 825–834. Springer, Heidelberg (2009)
3. Ghallab, M., Nau, D., Traverso, P.: Automated Planning: Theory & Practice. Morgan Kaufmann Publishers Inc., San Francisco (2004)
4. Russell, S.J., Norvig, P.: Artificial Intelligence: A Modern Approach, 3rd edn. Pearson Education (2010)
5. Bonet, B.: Planning as heuristic search. Artificial Intelligence 129(1-2), 5–33 (2001)
6. Keyder, E., Geffner, H.: Trees of shortest paths vs. Steiner trees: Understanding and improving delete relaxation heuristics. In: Proceedings of the 21st International Joint Conference on Artificial Intelligence (IJCAI 2009), pp. 1734–1749 (2009)
7. Karpas, E., Domshlak, C.: Optimal Search with Inadmissible Heuristics. In: International Conference on Automated Planning and Scheduling, pp. 92–100 (2012)
8. Haslum, P.: hm (P)= h1 (Pm): Alternative characterisations of the generalisation from hmax to hm. In: Proc. ICAPS, vol. 1, pp. 354–357 (2009)
9. Helmert, M., Domshlak, C.: Landmarks, critical paths and abstractions: Whats the difference anyway. In: Proc. ICAPS, vol. 9 (2009)
10. Öztürk, M., Tsoukiàs, A., Vincke, P.: Preference modelling. In: Figueira, J., Greco, S., Ehrgott, M. (eds.) Multiple Criteria Decision Analysis: State of the Art Surveys, pp. 27–72. Springer, Boston (2005)
11. Fong, J., Indulska, J., Robinson, R.: A preference modelling approach to support intelligibility in pervasive applications. In: 8th IEEE Workshop on Context Modeling and Reasoning (CoMoRea 2011), Seattle, USA, March 21-25, pp. 409–414. IEEE (2011)
12. Wolter, F., Wooldridge, M.: Temporal and dynamic logic. Journal of Indian Council of Philosophical Research XXVII(1), 249–276 (2011)
13. Chellas, B.F.: Modal Logic. Cambridge University Press (1980)
14. d'Agostino, M., Gabbay, D.M., Hähnle, R., Posegga, J.: Handbook of Tableau Methods. Kluwer Academic Publishers (1999)
15. Klimek, R.: Temporal preference models and their deduction-based analysis for pervasive applications. In: Benavente-Peces, C., Filipe, J. (eds.) Proceedings of 3rd International Conference on Pervasive and Embedded Computing and Communication Systems (PECCS 2013), Barcelona, Spain, February 19-21, pp. 131–134. SciTePress (2013)
16. Klimek, R.: Proposal to improve the requirements process through formal verification using deductive approach. In: Filipe, J., Maciaszek, L. (eds.) Proceedings of 7th International Conference on Evaluation of Novel Approaches to Software Engineering (ENASE 2012), Wrocław, Poland, June 29-30, pp. 105–114. SciTePress (2012)