Planning Human Centered Robot Activities

Vincent Montreuil†‡, Aurélie Clodic†, Rachid Alami†
LAAS-CNRS†
7, avenue du Colonel Roche
31077 Toulouse Cedex 4, France
Université Paul Sabatier‡
118, route de Narbonne
31062 Toulouse, France
firstname.lastname@laas.fr

***Abstract*— This paper addresses high-level robot planning issues for an interactive cognitive robot that has to act in presence or in collaboration with a human partner. We describe a task planner called HATP (for Human Aware Task Planner). HATP is especially designed to handle a set of human-centered constraints in order to provide "socially acceptable" plans that are oriented toward collaborative task achievement. We provide an overall description of HATP and discuss its main structure and algorithmic features.**

I. INTRODUCTION

Task planning is treated here as a mean to endow the robots with high-level decisional autonomy. Given a goal state (or a task) the robot is able to find a plan i.e. a (partially ordered) sequence of ground operators that satisfies it. Once a plan has been found the robot uses it to execute actions associated to planned operators. Chosen task planning formalism must be adapted to the application the robot will be involved in.

In this paper we address interactive and collaborative robots. In this kind of applications a robot and human partner act collaboratively in a shared environment implying that the robot must act in a safe way to ensure physical safety of its partners but also to ensure their mental comfort. This last characteristic must be considered during execution (for example by producing adapted movements [1], [2]) but also during high-level decisions. We adopt here a three-layered architecture (see [3] for an example), the highest layer of such an architecture is called deliberative layer and is composed of two components: the supervisor and the planner. The supervisor is responsible for decisions during execution and calls the planner when it needs to get a "procedure" to realize goals. We are developing both a supervisor called SHARY [4], [5] and a task planner called HATP (Human Aware Task Planner) especially designed for human-robot collaborative task achievement. SHARY embeds an explicit representation of the robot human partner as well as a context-dependent task refinement process that integrates collaborative task execution.

II. CONTEXT AND OBJECTIVES

Human-Robot Interaction (HRI) raises a number of challenges for robots at different levels (perception, decision making, motion planning and execution, etc.). It is thus important to identify the requirements that are specific to task-planning in this context. In [6] ten challenges for human-robot teamwork have been identified. Among these challenges several can be considered as associated to the robot planning abilities. Indeed, an interactive robot must be able to:

- signal in what tasks it can/wants participate,
- act in a predictable way to ensure human understanding of what it is doing,
- reveal its status and its intentions,
- negotiate about tasks with its human partner in order to determine roles and select how to perform the tasks,
- deal with "costs" and "utility" by taking into account time, resources, social rules, as well as human partner abilities and preferences.

Intentions and status expression: In order for the robot to act by "expressing" intentions we have designed a hierarchical execution controller based on Joint Intention Theory [7]. This controller is based on a task hierarchy. Hierarchical task decomposition enables the robot to associate monitors to different task abstraction levels and to detect situations that invalidate or may cause the irrelevance of the current plan. Starting from this, we have chosen the Hierarchical Task Network (HTN) formalism which is particularly adapted in this case. In this formalism, each non atomic task can be decomposed into subtasks permitting thus to get a task hierarchy. This hierarchical aspect has been also used in dialogue planning [8]. This approach uses the "recipes" of the SharedPlans formalism [9], [10] to produce a hierarchical structure showing intentions of agents involved in the dialogue allowing, thus, to produce adapted segments of discourse. Another advantage of the HTN formalism is its procedural aspect which features have been largely shown in programming language such as OpenPRS [11] or in multiagent reactive systems such as STEAM [12] and ABL [13].

Agent abilities and preferences: Considering HTN formalism allows us to provide adapted structures to the supervisor but it does not guarantee that the produced plans are optimal according to agents abilities/preferences which is a key point to produce acceptable plans. Indeed, HATP must consider that a human partner H is capable of but dislikes doing a specific action a by promoting plans in which a is not done by H.

Social aspects of the plan: Consideration of abilities and preferences is not sufficient to ensure that produces

plans are "socially acceptable". Indeed, agent abilities and preferences permit a local optimization of the plan but do not ensure that undesirable situations or combinations of actions will raise. To prevent that HATP must promote plans avoiding such situations.

Plan Negotiation: A task planner used by an interactive robot will have to produce plans for both the robot and its human partners. This does not mean that the robot will assign tasks to humans but it will propose a way to do them and wait for a validation from its human partners. In this context, there are two important key points to allow a good plan negotiation: (1) how the robot will introduce the plan to its partners and (2) how permitting a partial validation of the plan. Beyond interface issues, point (1) can be considered in HATP by giving it the feature to make plan abstractions. These abstractions are produced using notions of public/private plan parts as in Joint Activities [14] or in SharedPlans theory [9], [10]. Point (2) corresponds to the case of partial validation of the plan. Considering this, we want HATP to be able to consider partial plan structure as an input which will be completed to find a plan complying with the human requests.

Real time constraints: HATP is created to be used on board of robots. So, it is clear that it has to satisfy real time constraints. Indeed, if the robot takes too much time to produce high-level decisions, it would lead to the non-reactivity of the robot. This "passivity" in front of human requests may make the human partner feel that the robot misunderstand the request or is lost in its thoughts.

III. HATP FORMALISM

A. Agent Model

In order to represent abilities and preferences, the model of an Agent a_i is a set of couples $\{< A_k^{a_i},\ C_k^{ctxt} >\}$. $A_k^{a_i}$ is an action that can be done by Agent a_i and C_k^{ctxt} is the context dependent associated cost. This cost represent a mix of action utility and difficulty to realize it. For each action $A_k^{a_i}$ in a plan we add to the plan score C_k^{ctxt} evaluated in the current context. A cost is always positive.

B. Social rules

As said in section II, we have to take into account social rules in social evaluation of plans. So, we have defined a social rule as a couple $< S_k, P_k^{ctxt} >$. S_k is the description of the social rule and P_k^{ctxt} the associated context dependent penalty. For each violated social rule S_k in a plan we add to the plan score P_k^{ctxt} evaluated in the current context. A penalty is always positive. In HATP, social rules are defined as patterns in the plan structure or in the fact database. We have defined different kinds of social rules:

- undesirable states,
- undesirable sequences of actions,
- bad decompositions,
- effort balancing,
- control of intricacy,
- abstraction legibility.

Each kind of social rule has a specific description.

1) Undesirable state: This social rule describes states of the world which can be socially unacceptable. For example, an undesirable state can be the situation in which the robot has in hands a cleaning object and food at the same time. More formally, an undesirable state is a conjunction of facts.

2) Undesirable sequence: This social rule describes specific combinations of actions in the plan which can conduct to a feeling of unpleasantness for the human partner. For example, a plan in which the robot puts down an object and its human partner picks it up immediately after can make a bad feeling on human side. More formally, an undesirable sequence is a couple $< Seq, CO >$ in which Seq is the sequence by itself (i.e. actions and links between them) and CO a set of conditions that must be true during Seq.

3) Bad decompositions: This social rule describes the fact that specific ways to do given tasks must remain possible but they must be used only if it is necessary. For example, if the robot has to put down an object for someone on a piece of furniture, it is better to do it on the furniture instead of putting it inside the furniture. A bad decomposition is a pattern to detect during refinement process.

4) Effort balancing: This social rule is associated to the fact that to be acceptable a plan must be almost optimal without implying that one agent has to do everything. In this sense, we penalize plans in which there is an unbalance between agent efforts.

5) Control of intricacy: This social rule tries to avoid intricate human-robot task achievement with a high number of inter-dependencies and synchronization steps. Indeed, such plans might be fragile and uncomfortable from the human point of view who will feel "dependent" on synchronization with the robot and "locked-in" an automaton. To avoid that we penalize plans containing too much synchronization steps.

6) Abstraction legibility: This social rule is linked to plan negotiation. If the produced plan abstraction used for negotiation is too complicated it will make a bad HRI and a difficulty for the human partner to understand robot intentions. So, we penalize plans whit complicated abstractions.

We dot not claim that we have integrated all the necessary notions in HATP. Our aim is to design a generic platform that will allow to integrate other notions. For instance, an interesting idea is the notion of empathy raised in [15]. This could be useful to allow the task planner to produce plans taking into account human partners personalities.

C. Plan social score

The social score of a plan P is evaluated as follows:

$$S(P) = \sum_{a_i \in P} C_{a_i}^{ctxt} + \sum_{s_k \in P} P_{s_k}^{ctxt}$$

In this equation action a_i is a part of P and s_k is a violated social rule k in P. The social score of a plan P is the sum of costs of all its actions added to the sum of penalties of all violated social rules in P. Therefore the goal is to find plans with the sufficient low score.

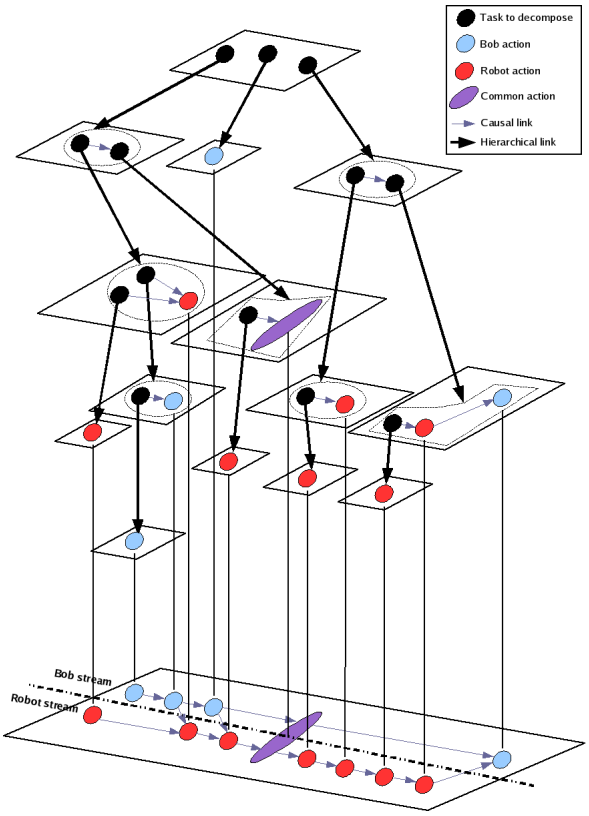

Fig. 1. HATP plan structure: the *Refinement Tree* and the *Time Projection*

IV. HATP PLAN STRUCTURE

A HATP plan is composed of two elements. The first one is used to keep a trace of selected decompositions during the HTN exploration, we call it *Refinement Tree* (it will be called only *Tree* in section V). The second one is used to determine the course of actions for each involved agents, it is composed of several time-lines of actions with causal links between these actions, we call it *Time Projection* (it will be called only *Projection* in section V). These two structures are illustrated by Fig. 1.

A. The Refinement Tree

The role of its structure is to maintain the instance of the HTN exploration for the current problem. Each node of the *Refinement Tree* is composed of partially ordered tasks. Each task in a *Refinement Tree* node is decomposed in its turn until all leaves are actions. Each task in the *Refinement Tree* is associated with a tag illustrating its current status (SATISFIED or UNSATISFIED).

B. The Time Projection

The *Time Projection* corresponds to the lowest level plan. It is composed of several time-lines of actions, one for each agent involved in plan realization. We make the assumption that one agent can do only one action at a given moment. Using HTN formalism to make plans with parallel actions has been already studied. An useful solution is based on combination of a HTN and a Simple Temporal Network

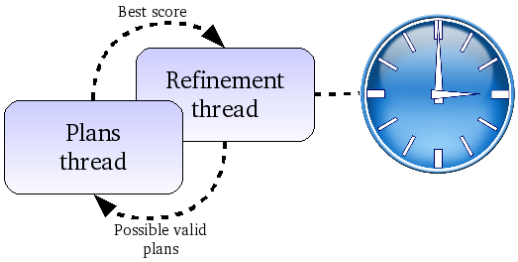

Fig. 2. HATP planning process threads

(STN). This approach has been recently studied in [16] and in SIADEX planner [17]. The possible simultaneity of two actions a_1 and a_2 is characterized by a lack of time constraints between their start and end timepoints i.e. (1) by the lack of causal links between them in the *Refinement Tree* and (2) by the lack of common resources needed by these two actions.

V. HATP ALGORITHM

HATP planning process is composed of three threads as illustrated on Fig. 2. One thread is responsible of the refinement search, when it has found a possible valid plan it transmits it to the thread responsible of complete evaluation and storage of plans. The refinement thread is also connected to a chronometer which role is to stop refinement process when available time has been consumed. This chronometer is necessary to ensure satisfaction of HATP real time constraints as seen in section II. Algorithm described in this paper is that of the refinement thread.

HATP algorithm presented in this section is considerably inspired from SHOP2 procedure [18]. The main difference is that in HATP we do not manipulate only a task set but also a tree. HATP main procedure is provided by algorithm 1. In this procedure we start by making a partial evaluation of the current couple (*Tree*, *Projection*) (line 1). If necessary we apply branch and bound optimization (lines 2-4). If not we search all *Tree* nodes with the UNSATISFIED status and without UNSATISFIED predecessors (line 6). If resulting set is empty we call storePlan(*Tree*, *Projection*) procedure and we make a backtrack to search a new plan (lines 8-9). If resulting set is not empty, we create branches in exploration tree if necessary (lines 14), we select a task and apply the appropriate procedure (lines 15-21).

The *storePlan* procedure makes a complete evaluation of the new possible plan and, if it is valid and better than the best plan actually in memory, stores it in *Stock*. *Stock* is just a sorted list of found plans, the sort criteria of this list is the plan score. The *applyAction* procedure checks action preconditions, if they are false we *backtrack* procedure is called, else modifications linked to the action are applied and *Projection* is updated. The *applyTask* procedure checks *task* preconditions and determines possible decompositions making branches in the exploration tree for each of them, one of this branch is chosen and *Tree* is updated accordingly.

Algorithm 1 HATP main procedure

Require: *Tree*, *Projection*

```
1: c ← partial_evaluation(Tree,Projection)
2: if ∃ plan P in Stock and c > evaluation(P) then
3:    cut current branch in exploration tree
4:    backtrack and goto 1
5: end if
6: TL ← {t ∈ Tree nodes | t is UNSATISFIED and t has
   no UNSATISFIED predecessors}
7: if TL = ∅ then
8:    call storePlan(Tree,Projection)
9:    backtrack and goto 1
10: else
11:    if cardinality(TL) = 1 then
12:       task ← member of TL
13:    else
14:       create branches in exploration tree
15:       select a branch and update task accordingly
16:    end if
17:    if task is an action then
18:       call applyAction(task,Tree,Projection)
19:    else
20:       call applyTask(task,Tree,Projection)
21:    end if
22:    goto 1
23: end if
```

VI. HATP IMPLEMENTATION

Using first promising tests [19] done with SHOP2 planner [18] we have validated our approach and have made some implementation choices. HATP implementation is done in C++. To get a fast planner able to be used on line, we have made the choice to use the same compilation process as JSHOP2 [20]. We make a domain-specific planner from domain-independent templates combined with a domain description. The user describes his HTN[1] and compiles it to produce C++ specific code and make a specific executable. To permit problems definition on line, HATP is designed as a multi-threaded system with a main server translating orders coming from a client thread.

In the current state, several domain-independent templates have been implemented and tested: the hierarchical system of decompositions is ready, the system is able to determine an abstraction of the plan for negotiation. Plan evaluation based on social constraints is under development. HATP will be soon connected to a scheduler able to provide a numerical estimation of plan execution. This will be done through an interface between HATP and a temporal planning library borrowed from IxTeT [21].

VII. RESULTS

An illustrative scenario: We have defined a simulation scenario containing "fetch-and-carry" aspects with several objects. We start at the situation illustrated by Fig. 3, robot is at the door, bob is at the sofa and wants to drink something on the sofa. This task involves a conjunction of three goals: (1) bob must have a glass, (2) bob must have a bottle and (3) bob must reach the sofa if he is not at it. The glass is in a closed cupboard and the bottle is on the table. Social rules in this small example can be: we do not want to have a bottle put on the sofa, to have the cupboard opened after getting the glass is not desired and we penalize the fact that the robot puts down an object and the human picks it up just after (we prefer the robot to give the object to the human). This small example is more difficult as it appears, indeed, there is a lot of choices to do about agent roles, task realization way and order in which task will be realized. This last point makes the search space exploded and so increases a lot computational time.

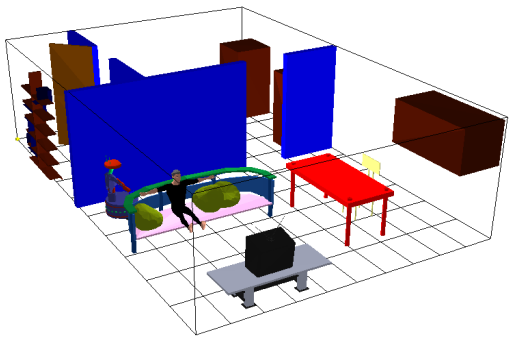

Fig. 3. An illustrative scenario

Produced plans: In order to illustrate the gain obtained by taking into account explicitly human-centered constraints, we compare HATP with an efficient task planner. We have run this scenario with SHOP2 with cost notions but without penalties linked to social rules. The best plan found by SHOP2 in 30s is illustrated by Fig. 4. It is important to note that plan parallelization is done in a second step. The produced plan seems to be composed of disordered actions making a feeling of confusion. If we look at Bob time-line we can see that he will participate superficially to several tasks without achieving anyone. Moreover we can see that there is a lot of synchronization steps in the plan making it fragile. Finally, we can see that the robot transmits the bottle to Bob by putting it down on the sofa forcing Bob to pick it up whereas it would have been better (for richness of the HRI) to give it.

Using HATP approach we have obtained the plan illustrated by Fig. 5. HATP takes about 16.5s to explore all possibilities and it produces a more legible plan, easier to understand and in which efforts seem to be more directly oriented toward goal realization. The different issues raised by the first plan we studied have disappeared especially because HATP has allocated Bob to goal (1) of the plan and Robot does almost everything in goal (2). The abstraction provided by HATP for plan negotiation is illustrated by Fig. 6. In this abstraction we have a subtree of the *Refinement Tree*

[1]More details about HATP syntax are available in `http://www.laas.fr/~vmontreu`.

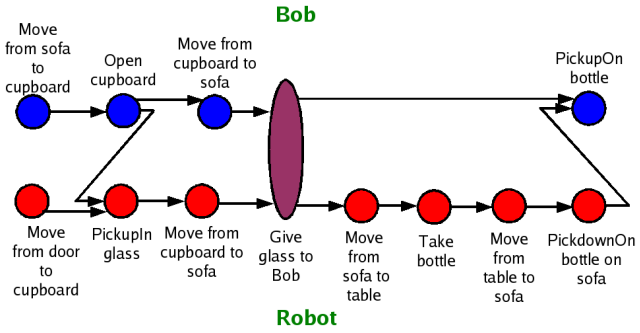

Fig. 4. a plan produced by a classical HTN planner

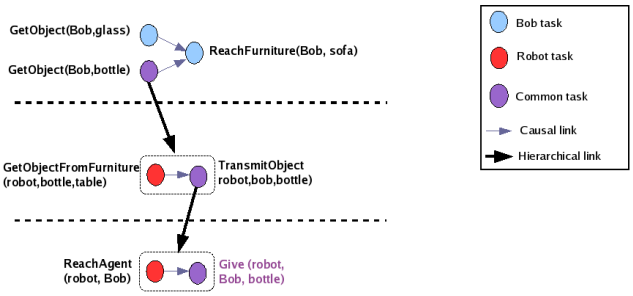

Fig. 6. An abstraction produced by HATP

in which individual plans have been cut (they are considered as private) and shared plans are public. It is important to note that although only the abstraction is presented to the human partner, the robot has in memory all the plan until *Time Projection*. This will be useful to introduce monitors to check and to influence human partner commitment during plan execution.

Partial structure as input: In order to illustrate the feature of HATP to take as input partial structures, consider the case where Bob said "I will get the glass by myself and you will bring me the bottle", high level decompositions of part (1) and (2) are imposed and we produce the same plan in only 3.9s.

Plan quality during planning process: Fig. 7 shows the number of plans added in the stock during planning time with and without human constraints. We remind that plans are added to the stock only if they are better than the best plan currently in memory. It means that each new plan added in the stock increases the quality of the final plan.

VIII. CONCLUSIONS AND FUTURE WORKS

We have described in this paper a task planner called HATP designed for interactive and collaborative robotic applications. It is able to produce socially acceptable plans for several agents by making a social evaluation of plans. We have provided several details on its implementation and have illustrated its results with an example. All HATP features are not implemented yet, but the implementation has been designed to facilitate future improvements. Although HATP is actually not used on board of a robot, it has been designed

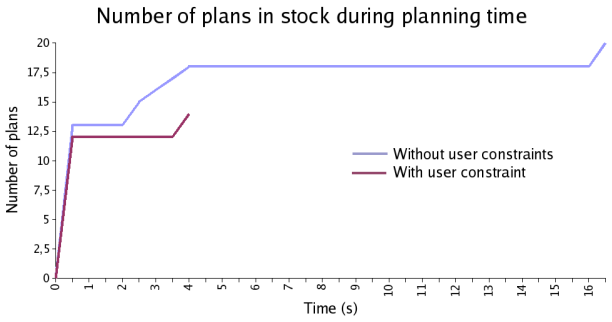

Fig. 7. Plans stock during planning time

for this and it will be done as soon as all its features will be implemented.

Future work on HATP will be on three aspects: (1) introduction of heuristics in the hierarchical decomposition system to explore the most promising parts of the solution space first, (2) the ability for the user to define a set of various constraints associated to the problem to influence choices made by HATP and (3) the introduction of monitoring in the plan to anticipate task performance control that will be used during execution. The aspect (1) is very important. Relevance of a heuristics is shown by results on Fig. 7, we can see on it that best plans are found at the end of exploration, so we can expect than with a heuristics we would be able to search in that part of the solution space first reducing thus time needed to plan. The aspect (2) will be very useful to increase richness of plan negotiation. The aspect (3) will allow the robot to anticipate task realization and so, to insert task performance monitors. Thus, the robot will be able to check at specific moments that its human partner is still involved in task realization and so that the task is still relevant.

IX. ACKNOWLEDGMENTS

The work described in this paper was conducted within the EU Integrated Project COGNIRON and was funded by the European Commission Division FP6-IST Future and Emerging Technologies under Contract FP6-002020.

REFERENCES

[1] S. Nonaka, K. Inoue, T. Arai, and Y. Mae, "Evaluation of human sense of security for coexisting robots using virtual reality," *IEEE Int. Conf. on Robotics and Automation*, 2004.

[2] E. A. Sisbot, R. Alami, T. Simeon, K. Dautenhahn, M. Walters, S. Woods, K. L. Koay, and C. Nehaniv, "Navigation in presence of humans," *IEEE-RAS Int. Conf. on Humanoid Robots, Humanoids2005, Tsukuba, Japan*, 2005.

[3] R. Alami, R. Chatila, S. Fleury, M. Ghallab, and F. Ingrand, "An architecture for autonomy," *Int. Journal of Robotics Research, Special Issue on Integrated Architectures for Robot Control and Programming*, vol. 17, no. 4, 1998.

[4] A. Clodic, V. Montreuil, R. Alami, and R. Chatila, "A decisional framework for autonomous robots interacting with humans," *IEEE Int. Workshop on Robot and Human Interactive Communication (RO-MAN)*, 2005.

[5] R. Alami, R. Chatila, A. Clodic, S. Fleury, M. Herrb, V. Montreuil, and E. A. Sisbot, "Towards human-aware cognitive robots," *The Fifth Int. Cognitive Robotics Workshop (The AAAI-06 Workshop on Cognitive Robotics, COGROB)*, 2006.

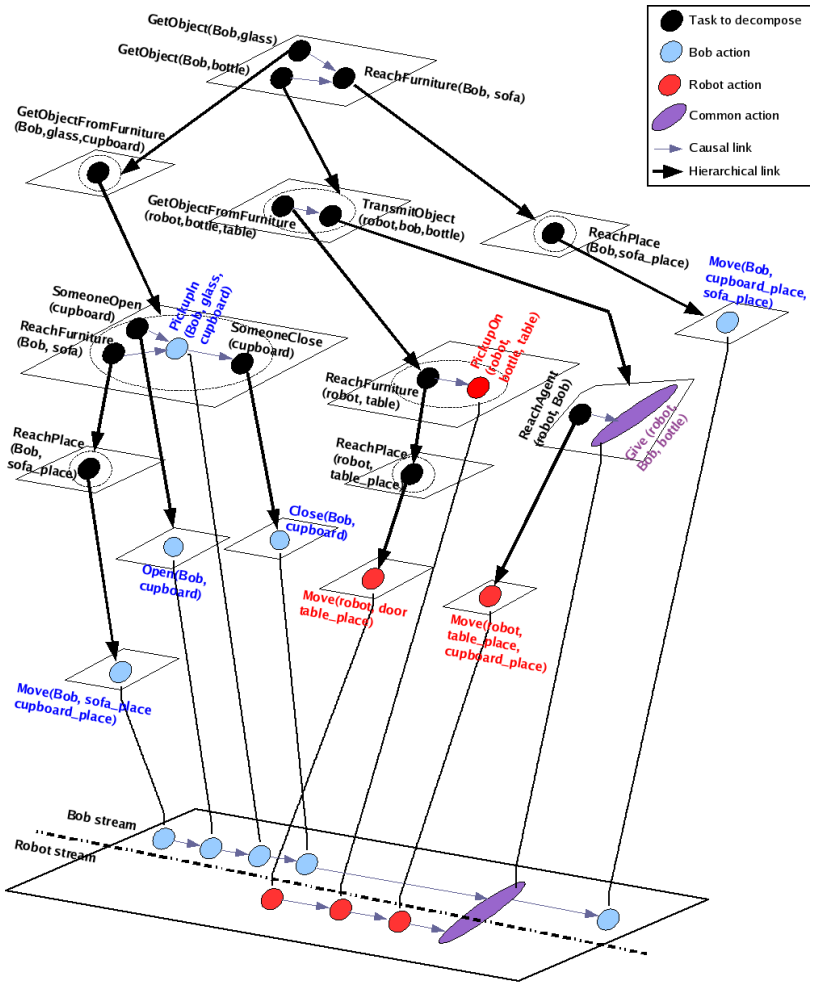

Fig. 5. a plan produced by HATP

[6] G. Klein, D. D. Woods, J. M. Bradshaw, R. R. Hoffman, and P. J. Feltovich, "Ten challenges for making automation a "team player" in joint human-agent activity," *IEEE Intelligent Systems*, vol. 19, no. 6, pp. 91–95, 2004.

[7] P. R. Cohen and H. J. Levesque, "Intention is choice with commitment," *Artificial Intelligence*, vol. 42, no. 2-3, pp. 213–361, 1990.

[8] K. E. Lochbaum, "A collaborative planning model of intentional structure," *Comput. Linguist.*, vol. 24, no. 4, pp. 525–572, 1998.

[9] B. Grosz and S. Kraus, "The evolution of SharedPlans," *Foundations and Theories of Rational Agencies*, 1999.

[10] B. J. Grosz and S. Kraus, "Collaborative plans for complex group action," *Artificial Intelligence*, vol. 86, pp. 269–358, 1996.

[11] F. F. Ingrand, R. Chatila, R. Alami, and F. Robert, "PRS: A high level supervision and control language for autonomous mobile robots," in *Proc. of the IEEE Int. Conf. on Robotics and Automation*, Minneapolis, USA, 1996, pp. 43–49.

[12] P. Scerri, D. V. Pynadath, L. Johnson, P. Rosenbloom, N. Schurr, M. Si, and M. Tambe, "A prototype infrastructure for distributed robot-agent-person teams," *The Second Int. Joint Conf. on Autonomous Agents and Multiagent Systems*, 2003.

[13] M. Mateas and A. Stern, "A behavior language: Joint action and behavioral idioms," *Book chapter in Life-like Characters. Tools, Affective Functions and Applications, eds H. Prendinger and M. Ishizuka, Springer*, 2004.

[14] H. H. Clark, *Using Language*. Cambridge University Press, 1996.

[15] A. Tapus and M. J. Mataric', "User personality matching with hands-off robot for post-stroke rehabilitation therapy," in *Proc. of the 10th Int. Symposium on Experimental Robotics*, Rio de Janeiro, Brazil, 2006.

[16] N. Yorke-Smith, "Exploiting the structure of hierarchical plans in temporal constraint propagation," in *Proc. of the National Conf. on Artificial Intelligence (AAAI)*, Pittsburgh, USA, 2005, pp. 1223–1228.

[17] L. Castillo, J. Fdez.-Olivares, O. Garca-Prez, and F. Palao, "Efficiently handling temporal knowledge in an HTN planner," *16th Int. Conf. on Automated Planning and Scheduling*, 2006.

[18] D. Nau, T. Au, O. Ilghami, U. Kuter, J. W. Murdock, D. Wu, and F. Yaman, "SHOP2: An HTN planning system," *Journal of Artificial Intelligence Research*, pp. 379–404, december 2003.

[19] V. Montreuil and R. Alami, "Report on paradigms for decisional interaction," LAAS-CNRS, Tech. Rep., 2005.

[20] O. Ilghami, "Documentation for JSHOP2," Department of Computer Science, University of Maryland, Tech. Rep., 2006.

[21] M. Ghallab and H. Laruelle, "Representation and control in IxTeT, a temporal planner," in *Proc. of Artificial Intelligence Planning Systems*, Chicago, USA, 1994, pp. 61–67.