

HATP: Hierarchical Agent-based Task Planner

Raphaël Lallement, Lavindra de Silva, Rachid Alami

▶ **To cite this version:**

Raphaël Lallement, Lavindra de Silva, Rachid Alami. HATP: Hierarchical Agent-based Task Planner: Demonstration. International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2018), Jul 2018, Stockholm, Sweden. hal-01944178

HAL Id: hal-01944178
https://laas.hal.science/hal-01944178

Submitted on 4 Dec 2018

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

HATP: Hierarchical Agent-based Task Planner

Demonstration

Raphaël Lallement
LAAS-CNRS, Uni. of Toulouse,
Toulouse, France
Raphael.Lallement@laas.fr

Lavindra de Silva
Institute for Advanced Manufacturing,
Uni. of Nottingham, Nottingham, UK
Lavindra.deSilva@nottingham.ac.uk

Rachid Alami
LAAS-CNRS, Uni. of Toulouse,
Toulouse, France
Rachid.Alami@laas.fr

KEYWORDS

Multi-robot planning; HTN planning; Geometric planning

ACM Reference Format:
Raphaël Lallement, Lavindra de Silva, and Rachid Alami. 2018. HATP: Hierarchical Agent-based Task Planner. In *Proc.of the 17th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2018), Stockholm, Sweden, July 10–15, 2018,* IFAAMAS, 3 pages.

1 INTRODUCTION

Hierarchical Task Network (HTN) planning is a proven approach to solving complex, real world planning problems more efficiently than planning from first principles when "standard operating procedures" (or "recipes") can be supplied by the user [13, 14]. By planning for tasks in the same order that they are later executed, total-order HTN planners always know the complete state of the world at each planning step. This enables writing more expressive planning domains than what is possible in partial-order HTN planning, such as preconditions with calls to external procedures [14]. Such features have facilitated the use of total-order HTN planners in agent systems and seen them excel in AI games [5, 7, 12, 17].

This paper describes the Hierarchical Agent-based Task Planner (HATP), a total-order HTN planner. Since its first implementation [11], HATP has had various extensions and integrations over the years, such as support for splitting a solution into multiple streams and assigning them to the agents in the domain [1]; modelling their beliefs as distinct world states [18]; allowing "social rules" to be included by the user to define what kind of agent behaviour is appropriate [1, 11]; allowing tasks to be planned by taking the human's safety and comfort into account [16]; and to interleave HTN and geometric planning [2, 3, 6].

Since many of these implementations have remained prototypes, we have significantly enhanced them as well as HATP itself, and integrated them into a stand-alone HATP distribution, which is now available as open source software (under a BSD 2-Clause License). This paper presents some of our recent improvements to HATP, and gives an overview of its user-friendly language, which treats agents as distinct entities; its mechanisms for effective control over decomposition; and its integration into our robotics framework.

2 LANGUAGE FEATURES

The HATP syntax is a product of lessons learnt over years of practical integration with real agent/robotic systems and other applications. Like in standard HTN planning, an HATP domain consists of a set of user supplied HTN methods ("recipes") and operators. For defining these, however, HATP employs some useful structured programming concepts such as structures, conditionals, and typing. Moreover, the agent types and object types in an HATP domain/problem are defined separately as a collection of structures called *entities*. An entity-structure has a set of attributes, each of which represents either a data value, or a relationship between the entity and other entities. An attribute can be one of the standard data types boolean, integer, or string. An attribute can also correspond to a set of objects, which can be manipulated in preconditions and effects via standard set operations. For example, an entity representing a robot-agent may have the attribute `carry` of type `Container` declared as a *set*, denoting that the robot can carry multiple objects of type `Container`.

An HATP initial state is an instantiation of the defined entities, with value assignments to their attributes; e.g. the robot-agent above could be instantiated with "`robot = new Agent;`" and its containers added via the statements "`robot.carry` **add** `con1; robot.carry` **add** `con2;`" where `con1` and `con2` are instantiated container objects.

To show that HATP's syntax is sound with respect to traditional HTN languages, we have developed in [4] a formal mapping from HATP's syntax into that of SHOP—the dominant HTN planner in the literature. Intuitively, attributes of entities correspond to predicate symbols in SHOP, and the entity names and attribute values to the parameters of predicates. For example, if variables `Rob`, `Loc1`, and `Loc2` represent robots and locations, then, the assignment "`Rob.at = Loc2`" will map to the two predicates "`at(Rob,Loc2)`" and "`¬at(Rob,Loc1)`" in SHOP (where `Loc1` is the robot's previous location), and "`Loc2.empty = false`" to "`¬empty(Loc2)`".

HATP supports developing agent-oriented domains by making a distinction between agents and other types of objects. Agents are treated as "first class citizens", and it is possible to define different types of agents. By virtue of making this distinction, HATP is able to split a solution (sequence of action nodes) found into multiple solution "streams", one per agent in the solution, and to add causal links between streams for synchronisation [1]. The various streams may then be executed concurrently in a multi-agent system by synchronising where necessary.

3 CONTROL OVER SEARCH

HATP provides ways to control how variables occurring in methods are bound at runtime, via the `SELECT` (which we abbreviate to `SEL`), `SEL-ORDERED`, and `SEL-ONCE` constructs. The `SEL` construct binds the given variable in the usual way. In essence, the construct amounts to a "backtrack point" that allows all values of the associated variable—and thus all ground instances of the method—to

Proc.of the 17th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2018), M. Dastani, G. Sukthankar, E. André, S. Koenig (eds.), July 10–15, 2018, Stockholm, Sweden. © 2018 International Foundation for Autonomous Agents and Multiagent Systems (www.ifaamas.org). All rights reserved.

be considered. SEL-ORDERED binds the variable in some given order, governed by a user-supplied ordering relation; the variable can be bound in ascending or descending order with respect to the relation. Finally, SEL-ONCE binds the variable only once—any remaining bindings are disregarded. This offers a reduction in the branching factor at the expense of completeness, as some of the ignored bindings may also yield HATP solutions. As an example of the value of SEL-ONCE, consider the setting where if one robot cannot navigate to a destination then none of the others will be able to either. Here, there is no need to consider all possible robot-object bindings: trying a single binding will suffice.

Recently, we have enabled the encoding of domain-dependent backtrack preferences by a developer, allowing the more "promising" backtrack points to be explored first, resulting in faster search. These preferences can be conditional, and the next backtrack point to explore can be at a different "level" in the search tree. For example, if it is impossible to plan a particular route to a restaurant (e.g. due to a road closure), it might be faster in some settings, before trying the other (possibly numerous) routes, to "jump" to an earlier backtrack point in the search tree and pick a different restaurant.

4 SYMBOLIC-GEOMETRIC PLANNING

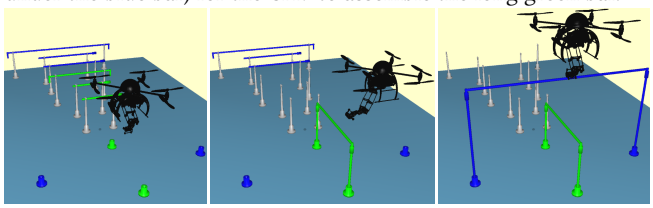
While an HTN hierarchy allows one to intuitively reason about high-level tasks in terms of more specific tasks, and eventually in terms of basic actions, the latter are still "abstractions" of the lowest possible level of detail, and they make certain assumptions about the world. For example, an HATP operator move(Robot,From,To) might assume that as long as location To is adjacent to location From, that the robot at From *will* be able to navigate to location To. In reality, however, this will not work when certain geometrical characteristics of the robot and the route make the move physically impossible. Combining HATP with geometric planning algorithms used in robotics is therefore crucial to be able to obtain primitive solutions that are viable in the real world.

To this end, we have developed a tighter integration with a geometric planner (GP) by taking inspiration from the prototype integrations in [3, 6]. Interleaved HTN and geometric planning is now a built-in feature of HATP. The main improvements are the following new syntactic constructs (which concretise the concepts discussed in [9]) for action definitions, which can be used to: *(i)* consult the GP, during HTN planning, to determine against the current 3D world state whether there *exists* a viable trajectory for an action; *(ii)* apply such a trajectory to the 3D world; *(iii)* compute the symbolic facts resulting from the trajectory, including any relevant side effects such as distanceMoved(robot1,15); *(iv)* transparently add such facts into HATP's symbolic world state; and *(v)* constrain the geometric goal states (positions and orientations) considered.

The listing below illustrates our new constructs. The precondition adds constraints to ensure that all robots in the same travel group as Rob are visible to it after the move; the constraints are considered by the GP when building goal states corresponding to To. The precondition then simulates a move in the 3D world to check whether a trajectory exists for one such goal state. If so, the "projects" line applies the trajectory to the 3D world, and the "effects" line computes and adds the resulting symbolic facts into HATP's state. The "cost" line retrieves the actual moved distance, computed by the GP.

```
action movWithGroup(Agent Rob, Location From, Location To, Type T) {
  preconditions {
    FORALL(Agent R, { R isIn Rob.travelGroup; }
      { addConstraint(Rob.ID, T.isVisible, R.ID, true) == true; });
    Rob.at == From;
    To isIn From.adjacent;
    checkMove(Rob, From, To) == true; // simulate to check that a move is possible
  };
  projects { applyMove(Rob, From, To) }; // do the move within the 3D world
  effects { Rob.at = To; }; // any side effect is added transparently
  cost { moveCost(From, To) }; // used for finding the optimal plan
  duration { moveDur(From, To) }; // used to estimate a plan's execution period
```

We evaluated our tighter integration of symbolic and geometric planning with a new case study, which required extending the GP to be able to model the dynamics of quadrotor UAVs, in order to simulate the assembly of complex structures by multiple cooperating UAVs. The screenshots below illustrate the assembly of coloured bars by a single UAV with an attached robotic arm. Each bar is moved from a holder (top left) and inserted into a matching slot, or into two vertical bars. The high-level tasks (e.g. 'pick' and 'insert') are planned by HATP, and their trajectories by the GP. Any correct high-level plan involves assembling the long blue bar last; otherwise, there would not exist a trajectory (due to a lack of space under the blue bar) for the UAV to assemble the long green bar.



Our second scenario is collaborative manipulation involving a robot and human [9]. At one level of abstraction, the robot must find and pick the objects on a table and move them onto another; at the lower level of abstraction, the robot must check that objects are graspable, and visible and reachable to the human after the move.

5 LAAS ROBOTICS ARCHITECTURE (LRA)

While HATP can be used as a stand-alone planner, it is also part of the LRA [8]: an interconnected set of components responsible for different aspects of the overall system. An important component is the Move3D [15] GP, which represents the real world in 3D and uses this as input for geometric planning. The robot uses various sensors to update its 3D state in real-time, e.g. a tag-based stereovision system is used for object identification and localisation, and a Kinect (Microsoft) sensor is used for tracking humans. The execution controller is the Procedural Reasoning System [10], responsible for invoking HATP when a task needs to be planned, and for executing the solution returned, by invoking the corresponding actuators on the (possibly simulated) robot. More recently, we have made HATP accessible to the robotics community by making each of HATP's constituent modules a ROS node. This enables standard and safe interoperability between HATP modules (e.g. via strongly typed data structures), and between HATP modules and other ROS nodes, e.g. the ones that control robots in the LRA.

REFERENCES

[1] Samir Alili, Rachid Alami, and Vincent Montreuil. 2009. A Task Planner for an Autonomous Social Robot. In *Distributed Autonomous Robotic Systems 8*. Springer Berlin Heidelberg, 335–344.

[2] Samir Alili, Amit Kumar Pandey, Emrah Akin Sisbot, and Rachid Alami. 2010. Interleaving Symbolic and Geometric Reasoning for a Robotic Assistant. In *ICAPS Workshop on Combining Action and Motion Planning*.

[3] Lavindra de Silva, Mamoun Gharbi, Amit Kumar Pandey, and Rachid Alami. 2014. A New Approach to Combined Symbolic-Geometric Backtracking in the Context of Human-Robot Interaction. In *Proc. of the International Conference on Robotics and Automation (ICRA)*. 3757–3763.

[4] Lavindra de Silva, Raphaël Lallement, and Rachid Alami. 2015. The HATP Hierarchical Planner: Formalisation and an Initial Study of its Usability and Practicality. In *Proc. of the International Conference on Intelligent Robots and Systems (IROS)*. 6465–6472.

[5] Lavindra de Silva and Lin Padgham. 2005. Planning on Demand in BDI Systems. In *Proc. of the International Conference on Automated Planning and Scheduling (ICAPS) Poster Session*. 37–40.

[6] Lavindra de Silva, Amit Kumar Pandey, and Rachid Alami. 2013. An Interface for Interleaved Symbolic-Geometric Planning and Backtracking. In *Proc. of the International Conference on Intelligent Robots and Systems (IROS)*. 232–239.

[7] Jürgen Dix, Héctor Muñoz-Avila, Dana S. Nau, and Lingling Zhang. 2003. IMPACTing SHOP: Putting an AI Planner Into a Multi-Agent Environment. *Annals of Mathematics and Artificial Intelligence* 37, 4 (2003), 381–407.

[8] Sara Fleury, Matthieu Herrb, and Raja Chatila. 1997. GenoM: A Tool for the Specification and the Implementation of Operating Modules in a Distributed Robot Architecture. In *Proc. of the International Conference on Intelligent Robots and Systems (IROS)*. 842–848.

[9] Mamoun Gharbi, Raphaël Lallement, and Rachid Alami. 2015. Combining Symbolic and Geometric Planning to Synthesize Human-Aware Plans: Toward More Efficient Combined Search. In *Proc. of the International Conference on Intelligent Robots and Systems (IROS)*. 6360–6365.

[10] François Félix Ingrand, Raja Chatila, Rachid Alami, and Frédéric Robert. 1996. PRS: A High Level Supervision and Control Language for Autonomous Mobile Robots. In *Proc. of the International Conference on Robotics and Automation (ICRA)*. 43–49.

[11] Vincent Montreuil, Aurélie Clodic, Maxime Ransan, and Rachid Alami. 2007. Planning Human Centered Robot Activities. In *Proc. of the International Conference on Systems, Man and Cybernetics (SMC)*. 2618–2623.

[12] Dana Nau, Tsz-Chiu Au, Okhtay Ilghami, Ugur Kuter, Héctor Muñoz-Avila, J. William Murdock, Dan Wu, and Fusun Yaman. 2005. Applications of SHOP and SHOP2. *IEEE Intelligent Systems* 20, 2 (2005), 34–41.

[13] Dana Nau, Tsz-Chiu Au, Okhtay Ilghami, Ugur Kuter, J. William Murdock, Dan Wu, and Fusun Yaman. 2003. SHOP2: An HTN Planning System. *Journal of Artificial Intelligence Research (JAIR)* 20, 1 (2003), 379–404.

[14] Dana Nau, Yue Cao, Amnon Lotem, and Hector Munoz-Avila. 1999. SHOP: Simple Hierarchical Ordered Planner. In *Proc. of the International Joint Conference on Artificial Intelligence (IJCAI)*. 968–973.

[15] Thierry Simeon, Jean-Paul Laumond, and Florent Lamiraux. 2001. Move3D: A Generic Platform for Path Planning. In *Proc. of the International Symposium on Assembly and Task Planning*. 25–30.

[16] Emrah Akin Sisbot, Aurélie Clodic, Rachid Alami, and Maxime Ransan. 2008. Supervision and Motion Planning for a Mobile Manipulator Interacting with Humans. In *Proc. of the International Conference on Human Robot Interaction (HRI)*. 327–334.

[17] Stephen J. J. Smith, Dana S. Nau, and Thomas A. Throop. 1998. Success in Spades: Using AI Planning Techniques to Win the World Championship of Computer Bridge. In *Proc. of the National Conference on Artificial Intelligence and Innovative Applications of Artificial Intelligence Conference (AAAI/IAAI)*. 1079–1086.

[18] Matthieu Warnier, Julien Guitton, Séverin Lemaignan, and Rachid Alami. 2012. When the Robot Puts Itself in Your Shoes. Managing and Exploiting Human and Robot Beliefs. In *Proc. of the International Conference on Robot and Human Interactive Communication (RO-MAN)*. 948–954.