HATP: Hierarchical Agent-based Task Planner

Raphael.Lallement @gnosco.net
Lavindra.deSilva @nottingham.ac.uk
Rachid.Alami @laas.fr

LAAS CNRS

University of Nottingham
UK | CHINA | MALAYSIA

HTN Planning

Hierarchical Task Network (HTN) planning is a proven approach to solving complex, real world planning problems

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

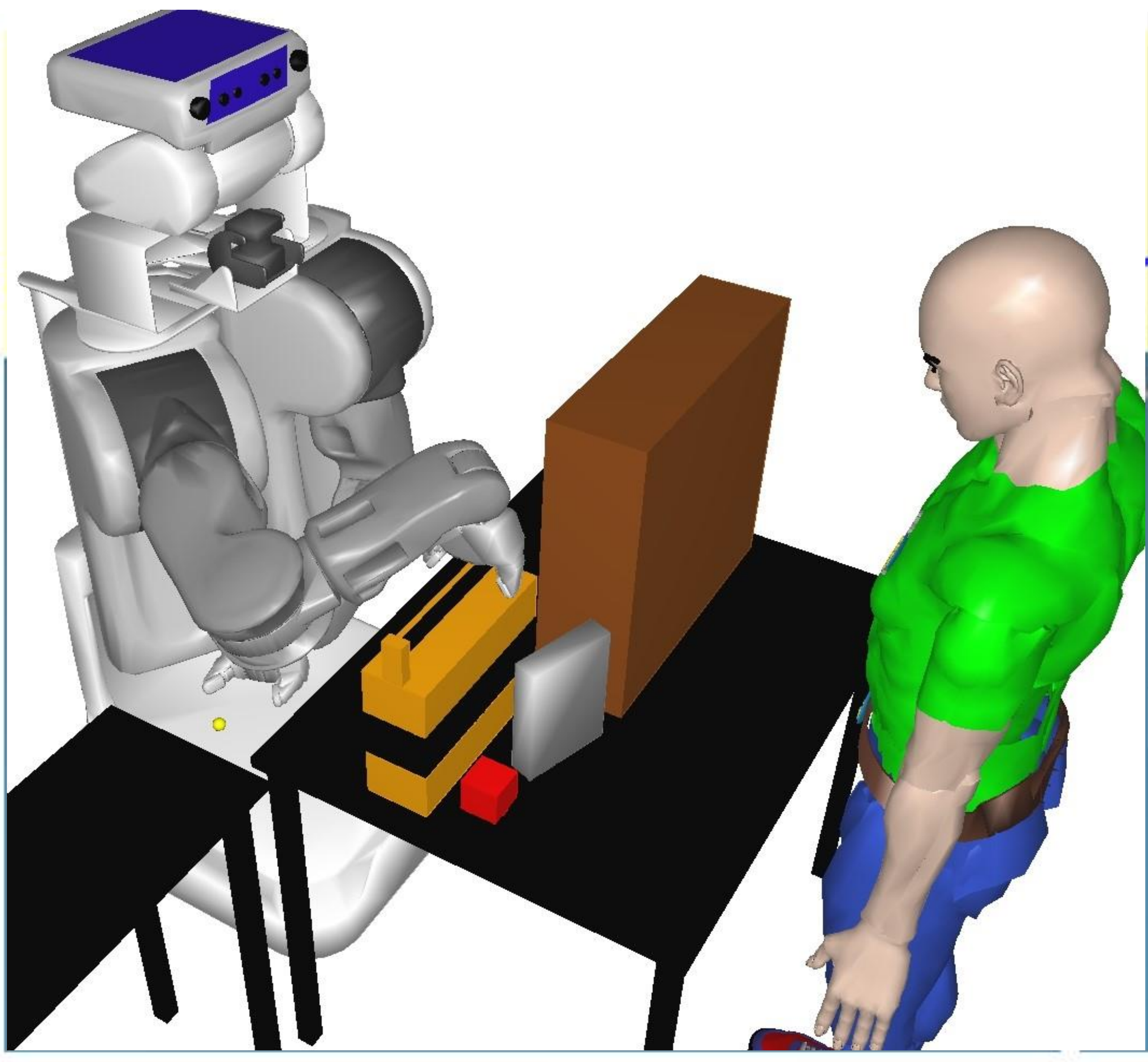
Such features have facilitated the use of total-order HTN planners in agent systems and seen them excel in AI games

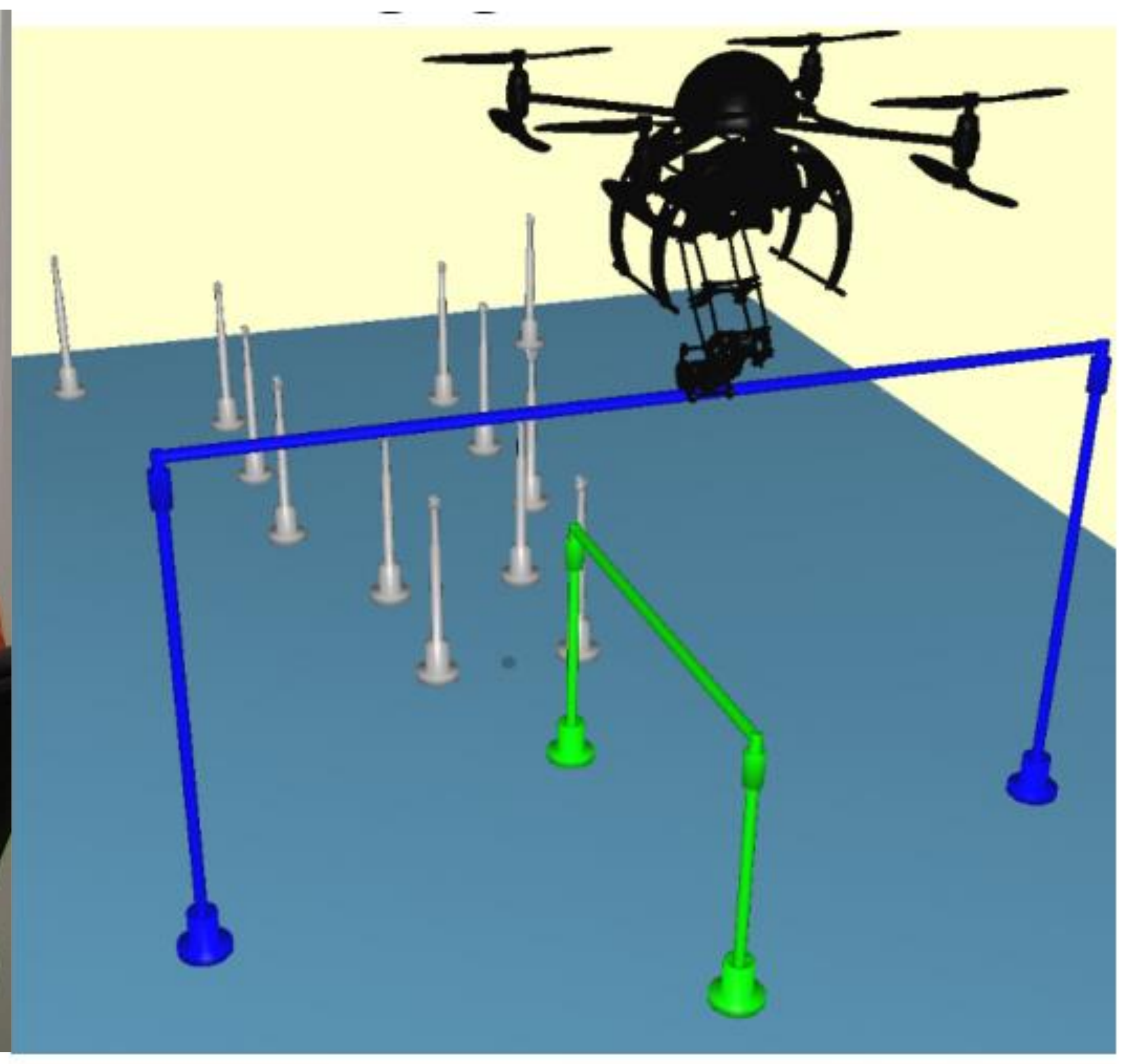
HATP

HATP is a total-order HTN planner

Since its first implementation (Montreuil et al., '07), 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
- allowing "social rules" to be included by the user to define what kind of agent behaviour is appropriate
- modelling their beliefs as distinct world states
- support for interleaved HTN and geometric planning





Features & Search Control

As in standard HTN planning, an HATP domain comprises 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, agent types and object types of an HATP domain or problem are defined separately as a collection of **entities**

HATP supports **search control**, e.g. how variables occurring in methods should be bound at runtime, via SELECT, SEL-ORDERED, and SEL-ONCE constructs

We have recently enabled encoding domain-dependent backtrack preferences, allowing the more "promising" backtrack points to be explored first

Finally, HATP supports developing agent-oriented domains by making a distinction between agents and other objects

Agents are treated as **first class citizens**, and it is possible to define different types of agents

Symbolic-Geometric Planning

An HATP operator such as *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 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

Interleaved HTN and geometric planning is now a built-in feature of HATP, as illustrated below in the task to move a group of UAVs

```
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
```