2014 IEEE International Conference on Robotics & Automation (ICRA)
Hong Kong Convention and Exhibition Center
May 31 - June 7, 2014. Hong Kong, China

A New Approach to Combined Symbolic-Geometric Backtracking in the Context of Human-Robot Interaction

Lavindra de Silva Mamoun Gharbi Amit Kumar Pandey Rachid Alami

***Abstract*— Bridging the gap between symbolic and geometric planning has received much attention in recent years. An important issue in some of the works that combine the two approaches is finding the right balance between backtracking at the symbolic level versus at the geometric planning level. We present in this work a new approach to interleaved backtracking, where the symbolic planner backtracks to try alternative action branches that naturally map to different geometric solutions. This eliminates the need to "protect" certain symbolic conditions when backtracking at the geometric level, and addresses a completeness issue in our previous approach to interleaved backtracking. We discuss a concrete, non-trivial symbolic-geometric planning example in the context of Human-Robot Interaction, a full implementation of the combined planning technique, and an evaluation of performance as well as the effect of increasing the symbolic-action branching factor.**

I. Introduction

This paper concerns the underlying issues in bridging the gap between symbolic and geometric planning, an area that is continuing to gain attention in the AI and Robotics communities. Of particular interest is finding an intuitive combination of the two types of planners so that each may borrow strength from the other. Some primary concerns are where geometric entities "fit" within symbolic entities; what geometric solutions mean at the symbolic level; how context information available in symbolic planning can serve as heuristics in geometric planning; and how symbolic information can be inferred from complex geometrical world states.

An important issue that is implicit in some of these efforts is the "tension" between backtracking at the symbolic level versus the geometric planning level. Both systems are able to do so through their own local data structures to try alternative solutions for choices that were made earlier. Relying only on one system to backtrack would mean losing out on solutions, but backtracking in an interleaved manner [1], [2] opens up new issues such as respecting the choices made at the symbolic level when the geometric planner backtracks, and keeping the geometric and symbolic world states sufficiently orthogonal so that backtracking at one level does not interfere with the other's state.

In the interleaved backtracking approach proposed in [1], [3] the authors keep the symbolic state orthogonal to the geometric one. Hence changing geometrical information such as the pose of an object on a table does not affect the symbolic state. For symbolic planning they use the JSHOP2 [4] HTN planner and for geometric planning a specialised path planner. This makes their approach similar to our approach in [2], where we also combine HTN planning with an extended geometric planner. In both approaches the geometric planner backtracks when an action being planned at the symbolic level is not applicable, by reconsidering previous geometric choices so as to make the new action applicable, e.g., changing the orientation of an object placed earlier to make room for a new object. However, unlike [1], [3], our approach in [2] does allow changes in the geometric world state to also affect the symbolic one. This is realised by sharing with the symbolic planner symbolic facts ("shared facts") extracted from the geometric state. We also needed to address the ramifications of geometric backtracking on already pursued symbolic actions and their preconditions.

Many approaches rely on being able to call external procedures—encapsulating geometric reasoners—when evaluating preconditions [5], [6], [7]. In [5], [6] the authors introduce "semantic attachments" which link predicates in the symbolic planning domain to external procedures implemented using, for example, a trajectory planner that computes collision free trajectories. If one exists, then the corresponding semantic attachment evaluates to $true$, and $false$ otherwise. Similarly, "effect applicators" in effects of actions consult the geometric planner to set certain state variables (e.g. the pose of an object moved) in the symbolic domain. For this to work, they require that effect applicators make deterministic choices: basically, calling an effect applicator from a particular state will always return the same effect. In contrast, it is important in our work to give the geometric planner some leeway to make choices. More importantly, each choice amounts to a different branch in symbolic planning, specifically, a different symbolic action.

While the above works keep the symbolic component somewhat detached from the geometric one, there is work that exploits a tighter link between the two. In [8], a hierarchical planner plans all the way down to the level of geometric actions making the two systems plan in the same "space" of solutions. Their hierarchical planner is a specialised one where primitive actions are implemented by invoking external solvers such as RRTs, and states model low-level geometric details such as robot joint configurations. Similarly, [9], [10] describe a hierarchical planner combined with a geometric motion planner for planning and then executing the most basic actions. In their work, however, geometric actions are executed while the plan is being constructed. Consequently, while "suggesters" in [9], despite being approximate computations, are similar to how

This work was conducted within the EU SAPHARI project funded by the E.C. division FP7-IST under contract ICT-287513. We thank Malik Ghallab and the anonymous reviewers for their very useful feedback and Kim Wallrafe for help with statistical tests.

Address: CNRS, LAAS, 7 avenue du Colonel Roche, F-31400 Toulouse, France; Univ. de Toulouse, LAAS, F-31400 Toulouse, France
Emails: {ldesilva,magharbi,akpandey,rachid}@laas.fr

we map symbolic actions to multiple geometric solutions, [9] commits to an action when its precondition holds and executes it, whereas we may backtrack over the action's other geometric solutions if it does not lead to a symbolic solution. Moreover, while [9] reasons about details such as "locations" and "paths" at the symbolic level, we do so at the geometric planning level.

The Asymov [11] system is a combined task and motion planner for problems that are difficult to solve when the symbolic planner is in control of the geometric search. Unlike the above approaches, the geometric planner in Asymov uses the symbolic planner and its domain to get heuristics when choosing roadmaps during geometric search. Likewise, in [12] a symbolic planner guides a sampling-based motion planner's tree-based exploration of the continuous space, which in turn returns utility estimates to improve the guide in the next iteration. This interplay between symbolic and geometric planning is also stressed in [13], where the symbolic plan guides the motion planner to find a continuous collision-free trajectory, failing which the symbolic planning problem is adjusted to take the cause of failure into account before planning again. While in [13] a complete symbolic plan is found first, and then a collision-free trajectory computed, our work interleaves the two.

Overall, a key difference in our work compared to others is that we do not use a trajectory/path planner but a geometric *task* planner [14], which lets us define the interface to symbolic planning more meaningfully. In particular, the symbolic planner no longer needs to make the "finer" decisions such as what grasp to use when picking an object, or where an object should be placed so that it remains visible to agents.

The contributions of this paper are the following. First, we present a new, intuitive combined planning approach where the symbolic planner backtracks to try alternative geometric solutions. With this strategy we do not need to explicitly "protect" preconditions of actions already pursued at the symbolic level, as we did in [2], [15] to ensure that the conditions are not contradicted by new geometric solutions. Moreover, by naturally resorting to backtracking at the symbolic—and consequently geometric—level when a shared fact is false, our new approach does not lose out on solutions, which was a minor issue in our earlier approach [2]. Second, we present a concrete symbolic-geometric planning example in an HRI context, with actions involving multiple agents, preconditions that make good use of shared facts, and a reasonably challenging geometric world depicting agents in a library. This framework relies on a fully implemented "intermediate layer", like the one in [2], which coordinates the two planners. Finally, we provide empirical results focusing on the runtime performance of the combined system in the context of the library domain, and on the effect of varying the "branching factor"—the limit on the number of geometric solutions per symbolic action.

II. Background

In this paper we use a STRIPS-like planning language [16]. While classical planners such as STRIPS focus on achieving some goal state, Hierarchical Task Network (HTN) planners focus on solving *abstract tasks*. HTN planning is also generally more efficient than classical planning because the search is kept focussed by appealing to a given library of recipes. In this paper we use a popular type of HTN planning called "totally-ordered" HTN planning (henceforth simply called HTN planning), which unlike "partially-ordered" HTN planning allows calls to external functions [17], which is important in our work. We define an HTN *planning problem* as the 3-tuple $\langle d, s_0, \mathcal{D} \rangle$, where d, the "goal" to achieve, is a sequence of primitive or abstract tasks, s_0 is the initial state, and $\mathcal{D}$ is an HTN *planning domain*. This is the pair $\mathcal{D} = \langle \mathcal{A}, \mathcal{M} \rangle$ where $\mathcal{A}$ is a finite set of operators, and $\mathcal{M}$ is a finite set of HTN *methods*. A method is a 4-tuple consisting of: the name of the method, the abstract task that it needs to solve, a precondition specifying when the method is applicable, and a *body* realising the "decomposition" of the task associated with the method into more specific subtasks. Specifically, the method-body is a sequence of primitive and/or abstract tasks.

The HTN planning process works by selecting applicable methods from $\mathcal{M}$ and applying them to abstract tasks in d in a depth-first manner. In each iteration, this will typically result in d becoming a "more primitive" sequence of tasks. The process continues until d has only primitive tasks left, which map to action names. At any stage during planning if no applicable method can be found for an abstract task, the planner essentially "backtracks" and tries an alternative method for an abstract task refined earlier.

Our Geometric Task Planner (GTP) searches in a discrete space of candidate grasps and placements [14] for (GTP) tasks involving picking and placing. The GTP framework lets us define day-to-day tasks, e.g. showing and giving objects, in terms of different constraints, based on factors such as the ability to reach and grasp an object. It works by iterating in a four dimensional search space of *agent "effort" levels*, *discrete grasps*, and *object placement positions* and *orientations*. Grasps are precomputed for each hand type (e.g. the robot's gripper) and object, and later used and filtered online based on the environment and task to solve. The amount of symbolic "effort units" needed to perform tasks such as turning the head, extending an arm, or standing up, are ranked based on intuitive assumptions: e.g., extending an arm requires less effort than standing up to take something.

The planner computes sets of placement positions and possible orientations of objects online, based on the environment, the task, and limits on the amount of effort allowed for the task. The sets are then weighted with respect to the situation, taking into account factors such as the object's stability implied by a given grasp and placement, whether two agents can comfortably grasp the object, and whether it is visible to the human from his/her viewpoint. Based on such reasoning, candidate final configurations are computed online and used with an RRT-based planner [18] to find a feasible trajectory. When feasible, computed partial trajectories are reused for successive planning requests. While the GTP is not "complete" in the sense of planning in a continuous search space, it is able to find a solution (if one exists) in the task's discrete search space. A solution is found using a constraint

hierarchy based approach, by successively introducing constraints during different planning stages, thereby reducing the search space successively before introducing relatively more computationally expensive constraints.

In this paper we use the following conventions: symbol ϵ is the empty string, variables begin with upper case, and predicates and constants with lower case. We assume that symbolic and geometric states are fully observable, and allow negative literals and universal quantification in preconditions of actions and HTN methods. Finally, this paper only deals with planning: we do not interleave planning with execution.

III. A NEW APPROACH TO INTERLEAVED BACKTRACKING

We shall first review our combined symbolic-geometric planning framework in [2], which forms the basis for this work. In [2] the HTN developer interacts with the GTP using evaluable predicates [5]; whether these evaluate to $true$ or $false$ are determined by associated external procedures. Specifically, every relevant GTP task t is associated with an evaluable predicate,[1] denoted $t^?$, in the HTN domain. For example, GTP task GIVE(B, H) has evaluable predicate $give(B, H)^?$ (where B is a book and H is a human), which evaluates to $true$ if t has a GTP solution from a given state and $false$ otherwise. We call an HTN operator (resp. action) a *Geometric-Symbolic* (GS) operator (resp. action) when such an evaluable predicate occurs in its precondition. Note that a GS operator maps to exactly one GTP task: only one evaluable predicate that corresponds to a GTP task can appear in a GS operator's precondition.

A GTP task t is also associated with an *add list function*, denoted t^+, and a *delete list function*, denoted t^-, which are the (possibly empty) add and delete lists for t, computed by the GTP based on the world state resulting from applying the solution that was found for t. For GTP task GIVE(b_1, h) for example, $give(b_1, h)^+$ might be the set $\{visible(b_1, h), reachable(b_1, h)\}$ and $give(b_1, h)^-$ the set $\{visible(b_3, h), reachable(b_3, h)\}$, indicating that after giving book b_1 to human h it is both visible and reachable to h, but book b_3 is neither visible nor reachable to h. Such predicates are based on geometrical properties and consequently modelled more accurately by the GTP. The $visible(O, H)$ predicate (for an object O) for instance, is computed based on whether the human's field of view in a 3D world overlaps with the object [19]. We call any such predicate that is computed by the GTP and "shared" with the HTN planner (via add and delete list functions) a *shared predicate* or *shared literal*. During HTN planning, the effects of a GS operator, then, are taken as the combination of its (pre-specified) add and delete lists together with its "dynamic" add and delete lists computed by the GTP.

The basic framework presented so far is inherently incomplete because the HTN planner assumes that applying a GS operator instance (GS action) to some state will yield only one possible outcome: that specified in its add and delete list, together with the one entailed by the GTP solution arbitrarily chosen on testing the action's precondition. If that GTP solution turns out to eventually not lead to an HTN solution, the HTN planner will, rightly, not backtrack to the GS action in question and reapply it. Intuitively, what we need then is to have multiple "instances" of the GS action, with each mapping to one possible GTP solution for the corresponding GTP task. The HTN planner would then naturally take into account the different GTP solutions for a GTP task by trying different GS action "instances". By abuse of terminology, we will continue to use the term *GS action instance* in this paper.

To this end, instead of changing the HTN planning algorithm, we apply a transformation to the set of HTN GS operators. Given the precondition ϕ of any GS operator (name) $o(v_1, \ldots, v_k)$ (where each v_i is a distinct variable), we replace ϕ with $\phi \wedge ins(id, N)$—and thus $o(v_1, \ldots, v_k)$ with $o(v_1, \ldots, v_k, N)$—where $ins(id, N)$ is a typed predicate, ins is any predicate symbol, and id is any constant symbol that uniquely identifies the operator;[2] moreover, we assume that the HTN initial state includes all elements in the set of facts $\{ins(id, n_1), \ldots, ins(id, n_m)\}$, where each n_i is a distinct object/constant of the same type as N, and $m \geq 1$ is the upper bound on the number of GS action instances.[3]

For example, given a GS operator GIVE(O, H) with precondition $give(O, H)^?$, we could modify the latter to $give(O, H)^? \wedge instance(give2, N)$ (and hence GIVE(O, H) to GIVE(O, H, N)) and add facts $instance(give2, 1), \ldots, instance(give2, 50)$ to the initial HTN state. Here we took symbol id as the concatenation of the operator's predicate symbol and arity, and we set the limit on the number of GS action instances to 50. This limit may be a theoretical maximum or one learned by analysing solution traces that occur in practice. So one possible GS action instance in our example is GIVE$(o, h, 1)$, and another is GIVE$(o, h, 44)$. If it happens that there are only 43 GTP solutions for the associated GTP task, then predicate $give(o, h)^?$ will evaluate to $false$ for all GS action instances GIVE$(o, h, 44)$ to GIVE$(o, h, 50)$. Indeed, this approach relies on some "bookkeeping" by an intermediate layer between the HTN planner and the GTP, particularly to find different GTP solutions for different GS action instances. We leave out the algorithms for a longer paper, but refer the reader to the similar algorithms in [2].

IV. AN EXAMPLE

In this section we present a non-trivial HTN example that makes use of GTP tasks.[4] It highlights: *(i)* how shared literals can be meaningfully used when writing an HTN domain, and their implications on the planning process; *(ii)* symbolic-geometric planning in a multi-agent HRI context,

[1] We assume that such predicates are non-negated and that there is no explicit quantification in their associated preconditions.

[2] symbol ins and variable symbol N do not appear anywhere else in the original HTN domain and in the operator, respectively

[3] Technically, to correctly keep track of the particular instance of the GS operator that the HTN planner is currently checking, if $gt(t_1, \ldots, t_n)$ is the GTP task in some transformed GS operator's precondition, we add as "dummy variables" to the task the subsequence $v_1, \ldots, v_m$ of variables occurring in the operator's name that are not in the set of terms $\{t_1, \ldots, t_n\}$, to get $gt(t_1, \ldots, t_n, v_1, \ldots, v_m)$.

[4] This example has been substantially adapted from [2], [15]; in particular, methods and many operator definitions have changed. The reader may still refer to those papers for details that we leave out in this section.

where certain GS actions require the HTN planner and the GTP to reason about not just the robot but also humans; *(iii)* a new combined symbolic-geometric backtracking approach where the HTN planner backtracks to try different GS action instances that essentially map to different GTP solutions.

Consider a PR2 working as a library receptionist alongside a human receptionist at an adjacent counter. Library members reserve books via the Internet with their membership ID, which can also be used to top up their library credit, etc. To collect the books reserved, the member must go to the library in person. Soon after a book is reserved, (human) librarians find those books and make them accessible (i.e. both *visible*, and *reachable* without navigating from the current position) to the PR2 on a shelf adjacent to it.

The part of the HTN domain we are interested in is illustrated graphically in Figure 1 and detailed in Table I. The top-level HTN task MANAGEORDER(M) for member M has one method (named m_1) with three subtasks: LENDBKS(M), TAKEPAYMENT(M) and HELPTAKEBKS(M). The first task, which lends all held (reserved) books, is associated with the two methods m_2 and m_3; these are tried in that order. Method m_2 is trivially applied if no (more) books are held by the member. If it is not applicable, m_3 is tried, which has the following primitive tasks: pick from the adjacent shelf a book held by the member; make it accessible to the member on the desk; verbally confirm the name of the book and the number of books that are yet to be made accessible to the member, but only if the new book was not placed at an "awkward" spot—on top of the POS (Point-of-Sale) machine stand—and does not hide from the member books that were previously made accessible;[5] and then recursively call LENDBKS(M).

The TAKEPAYMENT(M) task is associated with methods m_4 and m_5. The first debits the member's library-account directly if it has enough credit C; otherwise, the member must pay by credit card. The PLACEPOSM(M) task refines into two (ordered) methods for taking payment: if the POS machine $posm$ is (likely) reachable it is simply picked up, but if not—because it was taken by the human librarian working at the adjacent counter—he/she (lib) picks it up and gives it to the PR2, who takes it.[6] The POS machine is then shown and made accessible to the member either directly on the POS machine stand or even somewhere on the desk, if based on how the books were placed, the desk is likely to be more convenient for the member (require less effort).

Next, the member is asked to pay by swiping his/her card and entering the PIN, but only if the machine was not grouped too closely together with the books lent—it should be placed at a comfortable distance from them. The member then pays, and the PR2 puts away the POS machine. We make a simplifying assumption here that the member does not move the POS machine while paying. The PUTAWAYPOSM(M) task basically puts the machine somewhere that is away from the person's reach and sight. Finally, the HELPTAKEBKS(M) task directly hands the books to the member starting from those that are graspable, i.e. have a collision-free grasp. While the books may have been grouped very closely together when they were made accessible, the one that was made accessible last, at least, is guaranteed to be graspable.[7] In method m_{11} the SAY$(take)$ action simply speaks out a generic phrase asking the person to take the book, and TAKEBK(M, B) amounts to planning for member M to take book B from the PR2's gripper.

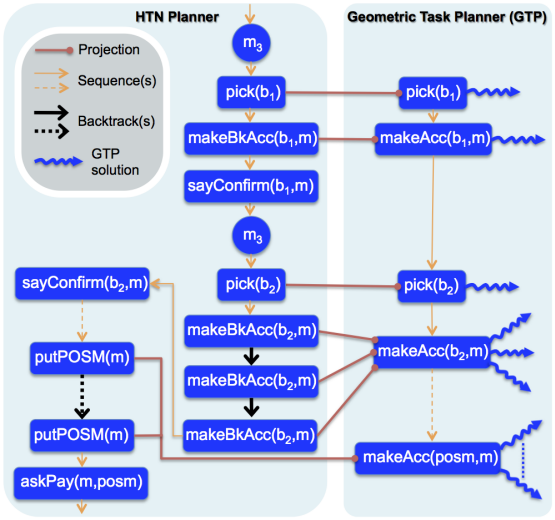


Fig. 2: A combined planning scenario highlighting the one-to-one mapping between GS action instances and GTP solutions

Note that predicates with symbols $visible$, $reachable$, $graspable$, $near$, and on are all shared literals. Note also that this domain leaves room to combine GTP tasks to create more "compound" ones. Although in the HTN domain consecutively picking an object and then showing it (or making it accessible) are shown as two separate HTN primitive tasks, these can be combined to form a single HTN primitive task, e.g. PSHOW(B, M), associated with a single GTP task that suitably picks the object so that it can be shown to the member M afterward [2], instead of picking it up with an arbitrary GTP solution and then triggering HTN backtracking when there is no corresponding GTP solution to show it.

Combined backtracking approach. Now suppose we have an initial world state where the reception desk is cluttered with objects, considerably reducing the available space on the desk, and that member m has reserved a small book b_1 and a big book b_2. Consider the following scenario (partly depicted in Figure 2). The PR2 successfully picks b_1 and makes it accessible to m (i.e. there are solutions for GTP tasks corresponding to the PICK(b_1) and MAKEBKACC(b_1, m) GS actions), and then after recursively calling LENDBKS(m), does the same with b_2. The pose of b_2, however, hides b_1 from m, making SAYCONFIRM(b_2, m)'s precondition not applicable, and triggering HTN backtracking, which picks a different "instance" of the MAKEBKACC(b_2, m) action

[5]We assume it is acceptable to place books on top of other objects such as boxes—just not on things such as telephones and POS machine stands.

[6]While planning the GTP task where the librarian gives the machine to the PR2 the GTP would pick *any* point in free space that requires the least amount of effort from the person, so we assume that during execution the librarian adapts their motion to coincide with the PR2's gripper.

[7]Indeed, being graspable may still not eventually permit a GTP solution for GIVEBK(B, M), as a collision-free trajectory may not exist.

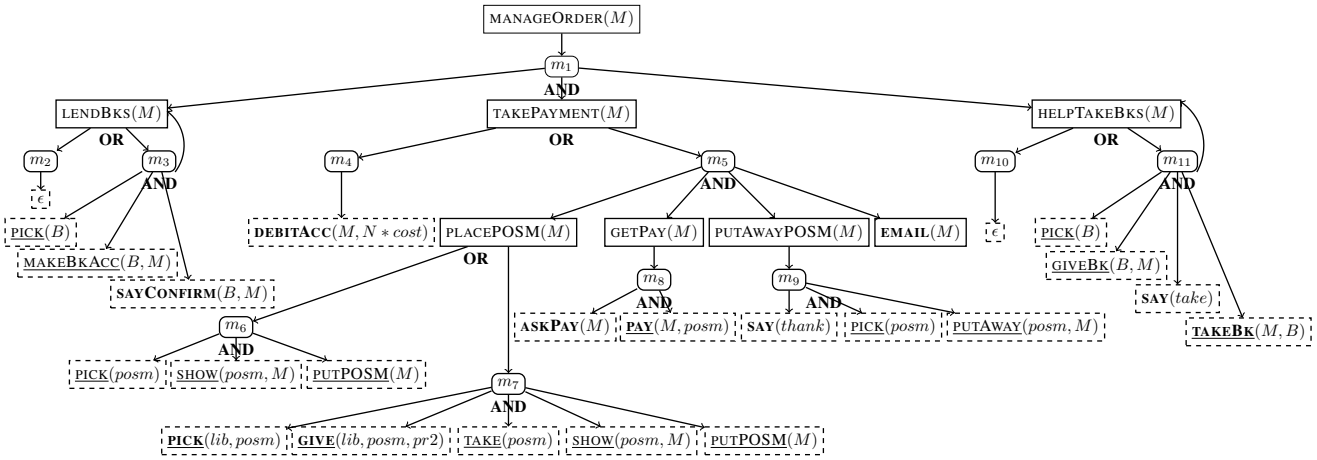


Fig. 1: The part of the HTN domain that handles online book reservations. Solid rectangles are HTN abstract tasks, rounded rectangles methods (where the incoming vertex is the task that the method solves, and outgoing vertices are the tasks in the method's body), and dashed rectangles primitive tasks. GS primitive tasks realised by humans are underlined and boldfaced and have the human as the first argument, those done by the PR2 are simply underlined, and non-GS actions are simply boldfaced.

ACTION/HTN TASK	M	PRECONDITION	METHOD-BODY/ACTION-EFFECTS
MANAGEORDER(M)	m_1	$visible(M, pr2) \wedge held(B, M)$	LENDBKS(M) · TAKEPAYMENT(M) · HELPTAKEBKS(M)
LENDBKS(M)	m_2	$\forall B, \neg held(B, M)$	ϵ
	m_3	$held(B, M) \wedge title(B, T)$	PICK(B) · MAKEBKACC(B, M) · **SAYCONFIRM**(B, M) · LENDBKS(M)
TAKEPAYMENT(M)	m_4	$nLnt(M, N)$	**DEBITACC**$(M, N * cost)$
	m_5	$nLnt(M, N) \wedge cred(M, C) \wedge (C < N * cost)$	PLACEPOSM(M) · GETPAY(M) · PUTAWAYPOSM(M) · **EMAIL**(M)
PLACEPOSM(M)	m_6	$reachable(posm, pr2)$	PICK$(posm)$ · SHOW$(posm, M)$ · PUTPOSM(M)
	m_7	$true$	**PICK**$(lib, posm)$ · **GIVE**$(lib, posm, pr2)$ · TAKE$(posm)$ · SHOW$(posm, M)$ · PUTPOSM(M)
GETPAY(M)	m_8	$true$	**ASKPAY**(M) · **PAY**$(M, posm)$
PUTAWAYPOSM(M)	m_9	$true$	**SAY**$(thank)$ · PICK$(posm)$ · PUTAWAY$(posm, M)$
HELPTAKEBKS(M)	m_{10}	$\forall B, \neg lent(B, M)$	ϵ
	m_{11}	$lent(B, M) \wedge graspable(B, pr2)$	PICK(B) · GIVEBK(B, M) · **SAY**$(take)$ · **TAKEBK**(M, B) · HELPTAKEBKS(M)
SAY(T)		$true$	$\{spoke(T)\}$
MAKEBKACC(B, M)		$held(B, M) \wedge makeAcc(B, M)^?$	$\{\neg held(B, M),\ lent(B, M)\},\ makeAcc(B, M)^-,\ makeAcc(B, M)^+$
TAKEBK(M, B)		$gave(B, M) \wedge take(M, B)^?$	$\{\neg gave(B, M), \neg lent(B, M)\},\ take(M, B)^-,\ take(M, B)^+$
SAYCONFIRM(B, M)		$(\forall B', lent(B', M) \models visible(B', M) \wedge \neg on(B', stnd)) \wedge title(B, T) \wedge lent(B, M) \wedge nLnt(M, N)$	$\{spoke(N+1), spoke(T), \neg nLnt(M, N),\ nLnt(M, N+1)\}$
GIVEBK(B, M)		$held(B, M) \wedge give(B, M)^?$	$\{gave(B, M)\},\ give(B, M)^-,\ give(B, M)^+$
PICK(O)		$visible(O, pr2) \wedge reachable(O, pr2) \wedge graspable(O, pr2) \wedge pick(O)^?$	$pick(O)^-,\ pick(O)^+$
DEBITACC$(M, Cost)$		$cred(M, C) \wedge (C \geq Cost)$	$\{\neg cred(M, C),\ cred(M, C - Cost)\}$
EMAIL(M)		$lent(B, M)$	$\{emailed(M)\}$
PUTPOSM(M)		$makeAcc(posm, M)^?$	$makeAcc(posm, M)^-,\ makeAcc(posm, M)^+$
ASKPAY(M)		$\forall B, lent(B, M) \models \neg near(B, posm)$	$\{spoke(swipe)\}$
PUTAWAY(O, M)		$putAway(O, M)^?$	$putAway(O, M)^-,\ putAway(O, M)^+$

TABLE I: The table for Figure 1. The top half (above the thick line) are methods (M) and the bottom half are operators. $nLnt$ stands for *number of books lent*. Empty sets have been omitted from the table. Omitted operators are defined like PUTAWAY(O, M).

that maps to a different GTP solution for MAKEACC(b_2, m). Due to limited space on the desk, the new solution places b_2 on the POS machine stand, which again triggers HTN backtracking due to SAYCONFIRM(b_2, m)'s precondition not being applicable. Next, (suppose) b_2 is properly placed and the GS actions PICK$(posm)$ and SHOW$(posm, m)$ of method m_6 are applied (not shown in the figure),[8] and then PUTPOSM(m) is applied, but because of its corresponding GTP solution, ASKPAY(m)'s precondition is not applicable: the POS machine was placed too close to b_2. This will again trigger HTN backtracking and subsequently finding alternative solutions for PUTPOSM(m) until a spot/pose for the machine is found that is not too close to the books lent.

Notice from this scenario that, like standard HTN actions, a GS action in the HTN domain may also bring about side-effects in the form of ground shared literals that are undesirable for, or contradict the precondition of, some action occurring later in the domain, leading to HTN backtracking. For example, $\neg visible(b_1, m)$ and $on(b_2, stnd)$ contradict the precondition of SAYCONFIRM(b_2, m). Moreover, backtracking is caused by shared literals only in some (generally minority) of the cases. For example, while MAKEBKACC(b_1, m) definitely brings about $visible(b_1, m)$, and MAKEBKACC(b_2, m) definitely brings about $visible(b_2, m)$, the latter action will only *possibly* bring about $\neg visible(b_1, m)$. Likewise, MAKEBKACC(b_2, m) will only possibly bring about

[8]We assume here that the member has no credit in their library-account and that the POS machine is reachable to the PR2.

GTP Task	T	S%	Pts	Gr	Orts	CL	TC
pMakeAcc	6.89	97	3.64	21.93	55.48	8.04	14.80
MakeAcc	1.31	100	8.57	0	18.70	2.32	1
pShow	8.54	95	4.19	51.65	12325	1.29	17.46
pGive	6.73	90	1.04	16.73	221.5	2.54	19.45
pPutAway	9.73	91	21.34	57.49	107.6	1.24	12.54

TABLE II: GTP tasks (with parameters omitted) planned along with: the time taken in seconds to find a solution (T), what percentage of their planning attempts were successful ($S\%$), and the total number of points tested in 3D space (Pts), grasps tried (Gr), orientations tried ($Orts$), calls to the GTP task, including calls to find alternative solutions and calls that resulted in no GTP solutions (CL), and calls to the trajectory planner (TC). Except $S\%$ all other columns are averages values.

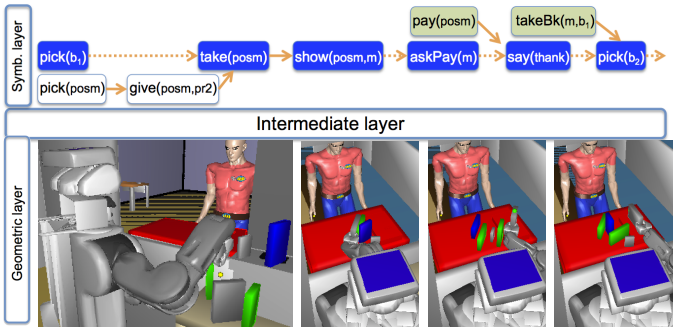
$on(b_2, stnd)$—if from all the available places the GTP plans to put b_2 on the POS machine stand.

V. Implementation and Evaluation

To analyse the runtime performance of the combined system we implemented the algorithms discussed in this paper, and a large part of the example in Figure 1. Specifically, we integrated the GTP tasks implemented in [14], and assumed that GTP tasks performed by the human always succeed; in the future we intend to integrate an existing system that plans human actions. Consequently, the GS actions we focus on in this section are those that are performed by the PR2. For the two planners we have chosen the HATP [20] HTN planner and our GTP [14], both of which have been used extensively in the LAAS architecture [21] for HRI experiments. Figure 3 shows an HATP partial plan (top), and screenshots of some GTP world states in our experiments. The former shows some of the actions of the PR2, librarian and the member, and how they are split into multiple "streams" by HATP and connected via causal links [20]. Notice from the GTP screenshots that we have simplified the shape of the POS machine to look like a larger book to facilitate tag based object identification and localisation.

Experiments were done on a machine with Ubuntu 10.04, one quad-core i7 Xeon processor, 8 GB of RAM, and a 460 GB hard drive. We also ran our implementation on a PR2 robot, which then let us extract real object and human locations in real-time to construct the initial states for planning, and execute GTP solutions found. The PR2 uses the Move3D [22] integrated planning and visualisation platform, and through various sensors maintains and updates the 3D world state in real time. For localising and tracking the human it uses data from an external RGB-D sensor. The human's gaze is simplified to his/her head orientation.

In our experiments there was one library-member, and the number of books reserved by him varied from 1 to 4, making it 2 to 5 manipulable objects in total including the POS machine. For each total number of objects there were 5 different initial states, and each of them had a different arrangement of objects making it 20 different initial states, which were picked in turn. In the HTN initial state the member had to always pay by credit card, which forced method m_5 (Figure 1) to be selected—the one that relies on the GTP. We also varied the "branching factor", or the maximum number of GS action instances to try, from 1 to 8: a branching factor of n corresponds to the upper bound m in Section III. For each branching factor we called HATP 60 times to solve the ManageOrder(M) top-level task.



(a) (b) (c) (d)

Fig. 3: An HATP solution being pursued (top) by communicating, via the intermediate layer, with the GTP, some of whose intermediate states are shown at the bottom. Actions in the topmost stream are the member's, in the middle the PR2's, and in the bottom the librarian's. At the bottom is: *(a)* an initial state with a small grey book next to a large blue one, 2 large green books, a grey POS machine, and a red desk; *(b)* the machine placed too close to the other books; *(c)* a partial GTP plan with all books placed so that none are blocked from the human's view, and with the machine placed on its stand—the red platform on the desk; and *(d)* another partial plan with the small book only just visible to the person.

The first set of results is summarised in Table II.[9] All tasks starting with *p* involved picking the object first, which is a separate GTP task as discussed in Section IV. We set a 40 second timeout for the GTP, because of which it sometimes failed to find a solution. Table II shows that all GTP tasks took a similar amount of time to plan and had a similar success rate, even with varying numbers of points, grasps, and orientations. Any relatively higher value in a column for one GTP task (e.g. $Orts$ for pGive) was roughly leveraged by a lower value for that task, in a different column (e.g. Pts). One value worth noting is the large number of orientations for pShow. This is because, to make it look natural, this task has additional constraints requiring the object and the gripper to be facing the human [14], which made many orientations fail when analysed from the human's perspective. The relatively short period of time taken to plan MakeAcc (which requires that the object be in the gripper) was because there was no need to compute all the trajectories related to grasping and picking up the object.

The length of the sequence of GS actions in a final HTN plan varied from 5 to 11 (with an average of 7), although actually it was almost double this length as most of the tasks listed in Table II include picking an object. The average time taken to find an HTN solution, including the communication overhead with the GTP, time taken to compute shared literals, and the HTN planning time, was approximately 90 seconds for 3 manipulable objects and a branching factor of 6.

Varying the branching factor. As expected, a lower branching factor meant losing out on solutions: the failure rate of ManageOrder(M) decreased as the branching factor increased, as shown in the first row of Table III. A branching

[9]The results are from an improved version of the GTP discussed in [15].

B	1	2	3	4	5	6	7	8
S%	50	83	90	100	100	100	100	100
N	15.73	12.41	13.51	11.71	13.71	12.06	15.42	15.63

TABLE III: The effect of increasing the branching factor (B) on success rate ($S\%$) of the MANAGEORDER(M) HTN task, and the total number of times the GTP was called (to plan some task) on average for each time the HTN planner found a solution.

factor of 1 also meant more calls to the GTP to find a solution because the higher rate of failure implied exhaustively trying the HTN options more often. The second row (N) of Table III shows how as soon as the branching factor exceeded 1 the number of calls to the GTP reduced and remained relatively stable until the branching factor got close to 8. We found this decrease and increase in values in row N to be statistically significant.[10] This implies that in our domain, and possibly other similar ones, after reaching a "good" branching factor(s) (anywhere from 2 to 6 in our domain), trying more will have a detrimental effect. We have observed, for instance, that if a book b was placed on the POS machine stand (which is unacceptable) because the desk was full of other books, instead of trying, say, 8 alternative poses for b, which will likely still be on the POS machine stand, it is sometimes better to resort to HTN backtracking and try alternative poses for the other books to make room for b (albeit with the increased risk of not finding a successful HTN solution). Note that this observation involving the $on(O, O')$ literal was not seen with the $visible(B, M)$ literal—the system never needed to try more than 2 alternative GTP solutions before the given ground instance of the literal could be satisfied.

VI. Discussion and Future Work

We presented an approach for interleaved symbolic-geometric backtracking, with an HRI example that exploits our HATP and GTP planners' ability to deal with multiple agents. Our approach is based on the notion of GS action "instances", which map to unique GTP solutions. In this intuitive combination HTN backtracking amounts to trying different GTP solutions. We presented a non-trivial domain that yields HATP solutions with reasonable numbers of GS actions, and poses challenges via obstacles, limited space for objects, and various shared literals that trigger backtracking.

This level of complexity naturally had an impact on the total time it took to find a complete HTN plan. One avenue to pursue, then, is to interleave planning with execution, as done in [9], [10] for instance, so that actions in a partial plan may be executed even before a complete HTN solution is found. Another direction to make our combined planning approach more practical is to develop heuristics, such as automatically finding symbolic literals in the HTN hierarchy that should be respected by GTP solutions. In our domain, $\neg on(b_1, stnd)$ is one such literal—it should not be removed by GTP solutions for MAKEACC(b_1, m). It would also be interesting to empirically compare our combined planning approach to the ones presented in [2], [3], where if a GTP task has no solution the GTP first backtracks (by itself) to find alternative solutions for previously planned GTP tasks. Since [2] does not, however, provide a fully implemented system, the implementation would need to be completed first. Finally, to have more comprehensive empirical results, future work will need to incorporate at least one other example such as a modified Towers of Hanoi problem [11].

[10] A t-test gave p-values below 0.05 for the data we gathered for columns $B = 1$ and $B = 2$, and likewise for $B = 6$ and $B = 7$. As expected, the p-values for our data corresponding to columns $B = 2$ to $B = 6$ indicated no significant difference in performance among those branching factors.

References

[1] L. Karlsson, J. Bidot, F. Lagriffoul, A. Saffiotti, U. Hillenbrand, and F. Schmidt, "Combining task and path planning for a humanoid two-arm robotic system," in *ICAPS Workshop on Combining Task and Motion Planning for Real-World Applications*, 2012, pp. 13–20.

[2] L. de Silva, A. K. Pandey, and R. Alami, "An interface for interleaved symbolic-geometric planning and backtracking," in *IROS*, 2013, pp. 232–239.

[3] F. Lagriffoul, D. Dimitrov, A. Saffiotti, and L. Karlsson, "Constraint propagation on interval bounds for dealing with geometric backtracking," in *IROS*, 2012, pp. 957–964.

[4] D. Nau, H. Muñoz Avila, Y. Cao, A. Lotem, and S. Mitchell, "Total-order planning with partially ordered subtasks," in *IJCAI*, 2001, pp. 425–430.

[5] C. Dornhege, M. Gissler, M. Teschner, and B. Nebel, "Integrating symbolic and geometric planning for mobile manipulation," in *IEEE International Workshop on Safety, Security and Rescue Robotics*, 2009.

[6] C. Dornhege, P. Eyerich, T. Keller, S. Trüg, M. Brenner, and B. Nebel, "Semantic attachments for domain-independent planning systems," in *ICAPS*, 2009, pp. 114–121.

[7] A. Gaschler, R. P. A. Petrick, T. Kröger, A. Knoll, and O. Khatib, "Robot task and motion planning with sets of convex polyhedra," in *RSS Workshop on Combined Robot Motion Planning and AI Planning for Practical Applications*, 2013.

[8] J. Wolfe, B. Marthi, and S. J. Russell, "Combined task and motion planning for mobile manipulation," in *ICAPS*, 2010, pp. 254–258.

[9] L. P. Kaelbling and T. Lozano-Pérez, "Hierarchical task and motion planning in the now," in *ICRA*, 2011, pp. 1470–1477.

[10] ——, "Unifying perception, estimation and action for mobile manipulation via belief space planning," in *ICRA*, 2012, pp. 2952–2959.

[11] S. Cambon, F. Gravot, and R. Alami, "A robot task planner that merges symbolic and geometric reasoning," in *ECAI*, 2004, pp. 895–899.

[12] E. Plaku and G. Hager, "Sampling-based motion and symbolic action planning with geometric and differential constraints," in *ICRA*, 2010, pp. 5002–5008.

[13] E. Erdem, K. Haspalamutgil, C. Palaz, V. Patoglu, and T. Uras, "Combining high-level causal reasoning with low-level geometric reasoning and motion planning for robotic manipulation," in *ICRA*, 2011, pp. 4575–4581.

[14] A. K. Pandey, J.-P. Saut, D. Sidobre, and R. Alami, "Towards planning human-robot interactive manipulation tasks: Task dependent and human oriented autonomous selection of grasp and placement," in *IEEE RAS/EMBS BioRob*, 2012, pp. 1371–1376.

[15] L. de Silva, A. K. Pandey, M. Gharbi, and R. Alami, "Towards combining HTN planning and Geometric Task Planning," in *RSS Workshop on Combined Robot Motion Planning and AI Planning for Practical Applications*, 2013.

[16] R. Fikes and N. Nilsson, "STRIPS: A new approach to the application of theorem proving to problem solving," *Artificial Intelligence*, vol. 2, no. 3-4, pp. 189–208, 1971.

[17] D. Nau, Y. Cao, A. Lotem, and H. Muñoz-Avila, "SHOP: Simple hierarchical ordered planner," in *IJCAI*, 1999, pp. 968–973.

[18] M. Gharbi, J. Cortes, and T. Simeon, "A sampling-based path planner for dual-arm manipulation," in *IEEE/ASME Int. Conf. on Advanced Intelligent Mechatronics*, 2008, pp. 383–388.

[19] A. K. Pandey and R. Alami, "Mightability maps: A perceptual level decisional framework for co-operative and competitive human-robot interaction," in *IROS*, 2010, pp. 5842–5848.

[20] S. Alili, R. Alami, and V. Montreuil, "A task planner for an autonomous social robot," in *Distributed Autonomous Robotic Systems*, 2009, pp. 335–344.

[21] S. Fleury, M. Herrb, and R. Chatila, "Genom: A tool for the specification and the implementation of operating modules in a distributed robot architecture," in *IROS-97*, 1997, pp. 842–848.

[22] T. Simeon, J.-P. Laumond, and F. Lamiraux, "Move3d: a generic platform for path planning," in *4th Int. Symp. on Assembly and Task Planning*, 2001, pp. 25–30.