CATs: Task Planning for Shared Control of Assistive Robots with Variable Autonomy

Bustamante, S.; Quere, G.; Leidner, D.; Vogel, J.; Stulp, F.


```
@INPROCEEDINGS{Bustamante2022,
author={Bustamante, S. and Quere, G. and Leidner, D. and Vogel, J. and Stulp, F.},
title={CATs: Task Planning for Shared Control of Assistive Robots with Variable Autonomy},
booktitle={2022 IEEE International Conference on Robotics and Automation ICRA},
year={2022},
volume={ },
number={ },
pages={8},
url={},
doi={}
}
```

CATs: Task Planning for Shared Control of Assistive Robots with Variable Autonomy

Samuel Bustamante, Gabriel Quere, Daniel Leidner, Jörn Vogel, Freek Stulp

***Abstract*— From robotic space assistance to healthcare robotics, there is increasing interest in robots that offer adaptable levels of autonomy. In this paper, we propose an action representation and planning framework that is able to generate plans that can be executed with both shared control and supervised autonomy, even switching between them during task execution. The action representation – *Constraint Action Templates (CATs)* – combine the advantages of Action Templates [1] and Shared Control Templates [2]. We demonstrate that CATs enable our planning framework to generate goal-directed plans for variations of a typical task of daily living, and that users can execute them on the wheelchair-robot EDAN in shared control or in autonomous mode.**

I. Introduction

In applications ranging from robotic space assistance [3] to healthcare robotics [4], there is an increasing need for robots with adaptable levels of autonomy, including direct control, shared control, supervised autonomy, and full autonomy. For instance, studies with users of wheelchair-mounted robots show that more robot autonomy is not always better, and flexible systems are recommended [5], [6]. As a concrete example on our own wheelchair-robot EDAN [7], a user may want to initiate the opening of a door in shared control [8], but let the crossing of the doorway be done autonomously.

In this paper, we propose a hybrid task-planning and motion-generation framework that enables plans to be executed with both shared control *and* supervised autonomy, as illustrated in Fig. 1. It combines the advantages of two approaches. From hybrid planning, it inherits hybrid symbolic/geometric action representations, which enable symbolic planning of actions sequences to achieve a given goal. From shared control, it inherits the feature that users are enabled to control a high-dimensional robotic system with low-dimensional user input commands with task-specific support.

Concretely, the main contributions of this paper are: 1) Proposing a novel action representation *Constraint Action Templates (CATs)*. As summarized in Fig. 2, CATs combine *Action Templates* [1] and *Shared Control Templates* [2]. 2) Introducing a novel hybrid task-planning framework for CATs, which enables goal-directed planning of shared control plans. 3) Demonstrating that CATs enable appropriate task plans to be automatically generated for variations of a typical task of daily living on EDAN [7]. 4) Show that the plans can be executed in shared control [9] or supervised autonomous mode [10], and even allows users to switch between them *during* task execution.

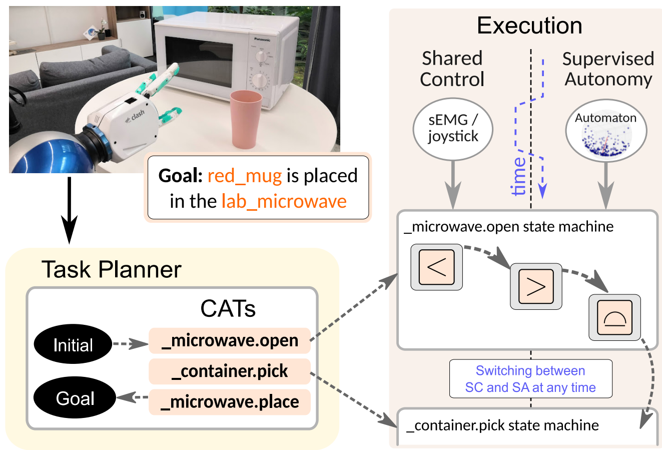

Fig. 1. Our task-planning framework in an example situation with our wheelchair-robot EDAN. **Left:** A symbolic planner generates a plan with all the necessary symbolic transitions to satisfy the goal. The shared control planner creates an SCT, which is a finite state machine. **Right:** During plan execution, the user can traverse the SCT with an interface in shared control, or request autonomous completion of the task.

The rest of this paper is structured as follows: in the next section, we present Action Templates and Shared Control Templates; further related work beyond these approaches is discussed later in Section VI. In Section III, we describe the CAT action representation, and Section IV explains how CATs are used for hybrid planning and plan execution. The validation on EDAN is described in Section V. We conclude with Section VII.

II. Background

We combine two action representations: Action Templates and Shared Control Templates, both illustrated in Fig. 2.

A. Action Templates (ATs)

ATs (Fig. 2, first row) are action representations that enable hybrid symbolic/geometric planning for autonomous robots [11], [1]. An example AT for a picking task (`object.pick`) is shown in Listing 1. The header of an AT is a declarative action definition specified with PDDL [12]. Its body is a sequence of robot operations that generates motion

All authors are with the German Aerospace Center (DLR), Robotics and Mechatronics Center (RMC), Münchner Str. 20, 82234 Weßling, Germany. Corresponding author: `Samuel.Bustamante@dlr.de`. This work is partly supported by the German Research Foundation (DFG) within the Collaborative Research Center EASE (SFB 1320) and the Bavarian Ministry of Economic Affairs, Regional Development and Energy (StMWi) by means of the project SMiLE2gether (LABAY102). The authors would like to thank Ribin Balachandran, Adrian Bauer, Annette Hagengruber, Florian Lay and Peter Lehner for continual support and motivating discussions.

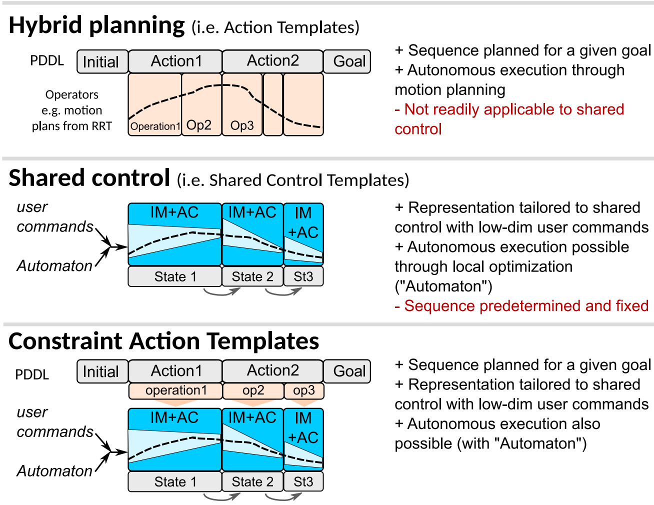

Fig. 2. Relation of CATs to previous work [1], [2].

plans and the required movements for the action, linked to geometric properties of the objects in the world.

```
;; Header: declarative action specification in PDDL

:parameters (?o - _object ?m  - _manipulator ?t  - _tray)
:precondition (and(free ?rn)  (on  ?o  ?t))
:effect  (and(bound ?o  ?m)  (not. (free ?m)) (not. (on
     ?o  ?t.)))


;; Body: a sequence of geometric operations that generate
;; motion plans and movements

(...)
operations = [
     ('move_hand', manip, graspset.approach_grasp) ,
     ('plan_to', manip, graspset.approach_frame, object_.
      frame),
     ('plan_to', manip, graspset.grasp_frame, object_.
      frame),
     ('bind', manip, object_.name),
     ('move_hand', manip, graspset.pre_grasp),
     ('move_hand', manip, graspset.grasp)]
```

Listing 1: AT for `_object.pick`, taken from [11].

Planning with ATs is a hybrid symbolic/geometric process. The AT planner retrieves the symbolic headers of all ATs, and a PDDL representation of the current state of the world. It then generates a symbolic plan, i.e. a chain of ATs, to satisfy the given goal. The geometric operations in the AT chain are then called in order (e.g. generating a motion plan with a rapidly-exploring random tree, RRT [13]), and executed in simulation to assert reachability and collision avoidance. If a simulation fails, the planner re-tries with different parameter sets; if all fail, a backtracking mechanism discards the action, and restarts the symbolic planner [1]. Finally, if the planner is able to simulate the whole plan, it is executed on the robot.

The implementations of the operations in an AT have not been made with shared control in mind, and the resulting plans can therefore not be readily used in the context of shared control. Our aim in this paper is to extend the AC framework, so that the resulting plans can (also) be used for shared control.

B. Shared Control Templates (SCTs)

SCTs are action representations that allow users to control high-dimensional robotic systems with only low-dimensional user input commands [2], [9], for instance from a 3D joystick or electromyography sensors. They provide task-relevant support in fields such as assistive robotics [2] or robotic surgery [14].

In an SCT, Input Mappings (IM) map the user commands to end-effector displacements. Different task phases require different IMs: when transporting a bottle, 3D input commands map to end-effector translational motions, with no orientation control to avoid spilling. However when pouring from the bottle, the bottle tip has a fixed position, but its orientation can be controlled, allowing the bottle to be tilted. IMs thus require knowledge about the relevant frames of reference of a task (e.g. the relative pose between a cup and a bottle tip), and which inputs map to which displacements in these frames. Additionally, Active Constraints may limit the range of motion provided by an IM, e.g. to limit the angle at which a bottle is tilted to avoid pouring too quickly.

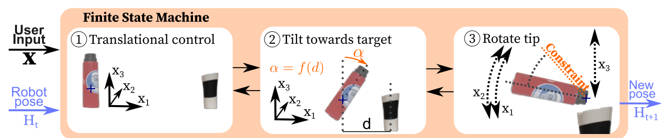

Fig. 3. An SCT with three states, each with their own Input Mapping and Active Constraints. Image adapted from [2].

As different phases of a task requires different mappings and constraints (e.g. transport of a bottle vs. pouring from it), an SCT combines several of them in a Finite State Machine, as illustrated in Fig. 3. State transitions occur when metrics of interest (such as the distance between the bottle and the mug while pouring, or the measured vertical force while releasing) reach established thresholds.

As we shall explain in more detail in Section IV-C, SCTs can be executed both with user commands as input (which motivated their design), but also with autonomously generated commands (which enables autonomous execution of SCTs, without user commands) [10].

So far SCTs have been hand-coded for tasks [2], or partially learned from demonstrations [9]. Our aim in this paper is to partially automate their generation with a hybrid task planning approach, as in the AT framework. See also the comparison in Fig. 2.

III. Constraint Action Templates (CATs)

CATs aim to have the best of both the AT and SCT worlds. That is, the task-planning functionality from ATs, and the shared control functionality from SCTs. How this combination is achieved is sketched in Fig. 2 (bottom), and explained in the following.

Each CAT has a PDDL action definition in the header, and geometric operations in its body. An example CAT file for opening a microwave (`_microwave.open`) is in Listing 2.

```
;; Header: declarative action specification in PDDL
;; Difference to AT: Effects distributed over blocks

@parameters
(?micro - _microwave ?rob - _manipulator)

@precondition
(and (free ?rob) (enclosed ?micro))


;; Body: a sequence of operations
;; Diff. to AT: operations are clustered in blocks
;; Diff. to AT: operations will be mapped to SCT states

@sets
use micro.sets.open[rob] as rmset

@block.approach_microwave
operation(rob, "move_fingers", rmset.open_hand)
operation(rob.frames.hand, "reach_full_pose", rmset.
      start_button)
effect = (and (not(free ?rob)))

@block.push_button
operation(rob, "move_fingers", rmset.microwave_pinch)
operation(rmset.fingertip, "reach_position", micro.frames
      .button_approach, use_constraint = "cone")
operation(rmset.fingertip, "reach_position", micro.frames
      .button_contact, use_constraint = "line",
      end_effector_force = force_button)
force_button = {axis:"x", value: micro.open_button_force}
effect = (and(not(enclosed ?micro)))

@block.go_back
operation(rob.frames.hand, "local_axis_motion", rob.
      frames.hand, axis = "-z", distance = 0.15)
operation(rob, "move_fingers", rmset.open_hand)
effect = (and(free ?rob))
```

Listing 2: CAT for `_microwave.open`.

CATs extend Action Templates by allowing for more fine-grained specification of effects of the individual operations in so-called "blocks". For instance, opening the microwave requires an approach, a push, and a go back block, which each have their own partial effects, see Listing 2. As we shall see in Section IV-B, each operation in the CAT maps to one state in the SCT. The block structure allows effects to be associated with the states in the SCT that achieve them.

The symbolic planner plans with PDDL action definitions, not with individual operations or blocks. Therefore, the overall effects of a CAT are determined by iterating from its last block to its first and obtaining the list of effects, as illustrated in Fig. 5 (left). When performing this so-called *effect tally*, the list should contain only non-repeated and non-negated state literals, e.g. one could not add `not(free ?robot)` from **`approach_microwave`**, because it is negated in the last block **`go_back`**.

As we are combining several action representations (PDDL, Action Templates, Shared Control Templates), let us clarify the etymology of our terminology before continuing. The term 'action' is taken from PDDL. In pure PDDL, there are only actions. Actions must have preconditions and completely specified effects. An action is the highest level of abstraction in Action Templates and CATs. The term 'operation' is taken from Action Templates, and represents one call to a motion or grasp planning algorithm (e.g. RRT) or a hand-coded procedure for motion generation. A 'block' is specific to CATS, and provides partial effect specifications for several operations.

A. Grounding of Objects and Frames of Reference

The operators in a CAT often refer to objects and frames of reference, e.g. `micro.frames.button_contact` is the frame of reference of the button of the microwave. All objects and their frames are stored in the Object DataBase (ODB), and updated in the World State Representation (WSR) [1], [7].

CATs and the ODB follow an object-centric paradigm: actions are defined around the objects classes that take part in them, according to a hierarchy illustrated in Fig. 4. During execution however, the parameters come from specific instances of the object classes. To give an example, the CAT in Listing 2 needs the parameters of a `_microwave` instance, such as the specific `lab_microwave`, and one `_manipulator` instance, such as the specific `edan_arm`.

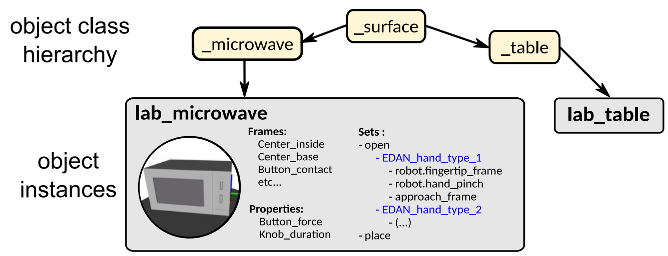

Fig. 4. A portion of the hierarchy for object classes, adapted from [1]. Object classes have a single leading underscore and are shown in yellow, while object instances (and their frames) are shown in grey.

Instance parameters (fig. 4) can have three types: *1) Instance properties:* For example the force expected from the `lab_microwave` button. *2) Frames:* explicit definitions of the object frames referenced to the instance. Each `_microwave` instance will store a `button_contact` frame with respect to its origin. *3) Object-Robot sets:* Some properties are specific to a combination of an object instance and a robot instance. Sets (**@sets** in Listing 2) contain an arbitrary number of robot-object properties on a given context. For instance, `lab_microwave` stores different *opening* finger configuration and fingertip frames, one for each hand type of EDAN.

IV. Planning and Execution with CATs

A. Symbolic planner

The parameters, preconditions and effects (acquired through effect tallies) of each CAT and the current state of the objects in the world (see Section III-A) completely specify what is known as the 'domain' in the PDDL language. Given the domain and the user goal (also expressed in PDDL e.g. `(on red_mug lab_micro)` [1]) , the planner produces the sequence of CATs that can turn the state of the world into a state containing the goal, if such a plan exists. We use Fast Downward [15] for end-to-end PDDL planning.

[1]Note: with `(on object microwave)` we model "on the microwave inner surface" and not "on top of the microwave".

In the example in Fig. 5, the goal `(on red_mug lab_micro )` can be achieved by first opening the door of the microwave (using the CAT in Listing 2), then grasping the mug, and finally placing the mug in the microwave. But the same actions could also fulfill other goals from different initial states; for example, the planner could also use use `_microwave.open` or `_container.pick` to achieve goals like `(opened lab_micro)` or `(on red_mug lab_table)`.

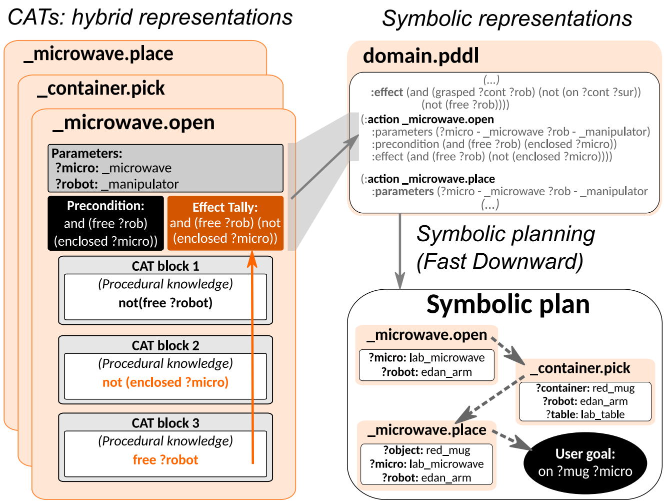

Fig. 5. Symbolic planning with CATs **Left:** Schematic of the CAT for `_microwave.open` and the procedure to obtain an effect tally. **Top right:** The PDDL information, included the effect tally, provides a PDDL action description, and is stored in a `domain.pddl` file. **Bottom right:** An end-to-end PDDL planner can generate a symbolic plan that solves the goal with the object instances in the world.

B. Shared Control planner

As mentioned in Section II-A, the implementation of operations in Action Templates cannot be readily used in the context of shared control. CATs solve this problem by mapping each operation to one state of a Shared Control Template (see Section II-B), which have been designed specifically for this purpose. The output of the planner is one large Shared Control Template. In comparison, such FSMs in previous work needed to be created and fine-tuned by hand [2].

Once there is a symbolic plan containing a list of CATs to be traversed by the robot, our task-planning framework aggregates all their operations and generates an new SCT from them. This SCT, shown in Fig. 6A., is a large linear FSM containing one state for every operation in the planned CATs, and only forward transitions. The planner creates the basic elements of an SCT state (input mappings, active constraints, and transitions), and includes the parameters of the object instances. Operations thus provide a blueprint for the robot to interpret *how* a movement should be performed regardless of the autonomy level.

The structure of a CAT operation is shown on Fig. 6B. It contains three required arguments and an unlimited number of optional keyword arguments. The required arguments are, in order, the motion reference (meaning the frame or item

TABLE I

OPERATIONS INCLUDED IN OUR CATS IMPLEMENTATION

Operation name **Implementation**	**Transition**
`reach_full_pose` The user starts or stops the motion, which is a simple point-to-point motion in $SE(3)$.	Target position and orientation are reached with a small tolerance.
`reach_position` The user controls the actuated frame in a translational motion, and the motion uses constraints defined by the axes of the target frame (e.g. a cone on the Z axis).	Position (but not orientation) is reached with a small tolerance.
`move_fingers` The robot moves the fingers while the user is waiting.	Timeout of one second to finish the finger motion.
`local_axis_motion` The user controls a 1D translational motion, in one of the axes of the target frame.	Actuated frame displacement reaches a treshold.

actuated by the robot, like the hand configuration or the fingertip frame), the type of the operation, and a target (*viz.* an object frame or a finger target). Keyword arguments can add adjustments to the motion, most notably constraints (cones, lines, etc.) from our constraint model collection [9]. Our first implementation of CATs contains four operation types, explained in Table I, each of which has a prefabricated state primitive. Each generated SCT state is thus assembled from the primitives and the keyword arguments.

Example: To press a microwave button in Listing 2 (**`@block.push_button`**) the SCT would first contain a state in which the fingers would move to a pinch (first op.); then, a state in which the user could steer the joystick-controlled fingertip in 3D while staying in a cone volume towards an approach position (relative to the button, second op.); finally, in another state the SCT would assist the button press by applying a line constraint to the fingertip (third op.).

Force transitions: SCTs states and the Automaton have support for force transitions [2], [10]. CATs wrap this functionality in the keyword arguments, and allow to override the original primitive transitions. To name one example: in **`@block.push_button`** of Listing 2, the transition where the fingertip frame reaches the contact button (third op.) only happens after the robot senses a horizontal force, using an instance-specific treshold `micro.open_button_force`.

C. Execution of the generated SCT

The SCT generated by the planner can be executed in shared control, supervised autonomy, or a mixture of both. For example, while opening a microwave, the user could start moving with a 3D joystick, but then click to let the robot finish the task when it gets close to the button. The user can also trade back autonomy at any time, and continue with the task in shared control. Although the original SCTs were designed for shared control with user inputs, an *Automaton* (presented in [10]) can execute them autonomously by generating the commands, rather than a human. Furthermore,

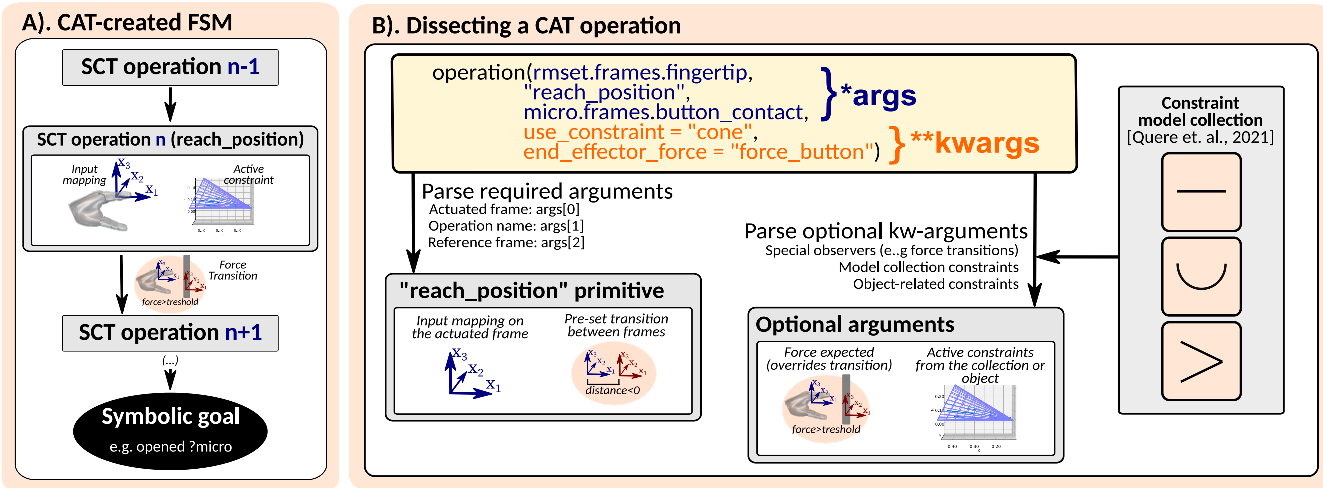

Fig. 6. **Left:** Schematic of an SCT generated by our planning framework. **Right**: Procedure for generating one SCT state from a CAT operation.

we demonstrated previously that smooth transitions between shared control and autonomous execution are possible *during* the task [10].

The Automaton is capable of traversing all the states required by a plan, effectively providing supervised autonomy after a button press, and within an SCT. However, before CATs these plans had been provided *ad-hoc* for every SCT, as the robot did not have any symbolic knowledge of the tasks and was not able to reason about goals.

V. Validation on EDAN

Robot: All experiments were conducted on EDAN, which consists of a wheelchair with a 8-DOF light-weight robot with a three-fingered hand shown in fig. 7. The wheelchair base, arm, and hand can be controlled through electromyography, a 3D joystick, and/or a tablet graphical user interface (GUI). As a fallback to SCT execution, EDAN offers a manual mode interface with mode switches. More details about EDAN and these interfaces can be found in [7]. Only the 3D joystick interface was used in the experiments.

Task: We consider an activity of daily living involving the placement of a cup in and out of a microwave. The different variations of this task, i.e. the different initial conditions and goals, are listed Table II. Both are provided to the planning algorithm as PDDL specifications.

Planning process: For each demo, the position of the objects and the initial symbolic state[2] were specified before the experiment, and loaded into the World State Representation (WSR). Based on the goal, the initial state and the library of CATs, symbolic plans were generated with Fast Downward [15] just-in-time before the plan execution. The different CATs that were implemented and used by the planner are listed in the final column of Table II.

Plan execution: Plans were executed either in shared control, supervised autonomy, or switching between the two during the task (see the fourth column in Table II). Shared control was performed by an expert user, using a 3D joystick for shared control. Plan executions are illustrated both in Fig. 7, as well as in the video attached as supplementary material.

Previous work has shown that executing SCTs (without CATs) in autonomous mode is not slower than in shared control [10], and that a human in shared control is faster than in direct manual control [2].

Planning time duration: In a separate test (not executed on the robot), we show on table III the planning times for 10 runs of a `(on red_mug lab_microwave)` task on an Intel(R) Xeon(R) with 8 cores. The times reported in the *Symbolic Plan* row include the generation of the `domain.pddl` file from the CAT files and the call to Fast Downward. We also show the time needed to convert the symbolic plan to an SCT. To assess how well planning times scale with the number of objects in the world, we generated plans for scenes containing 10 and 40 mugs, where only one was involved in the plan.

Discussion and limitations: The experiments demonstrate that our approach is able to generate symbolic plans for variations of a task, convert the plans to SCTs, and execute these SCTs with different levels of autonomy. Planning times in the experiments are below 1s. Although we cannot prove that an SCT is always successfully executed autonomously, in previous work we have shown a success rate of 93-98% for SCT task execution in an obstacle-free scenario [10]. We do not yet implement joint space planning nor simulations with feasibility checks and backtracking mechanisms as in the Action Templates approach. We expect to implement these features in future work to improve the robustness of the CATs framework, albeit with longer planning times.

VI. Related Work

There is a need for goal-guided interaction in Human-Computer interaction, particularly for non-expert users [16]. However, while there is a large body of work in shared control of assistive robotic systems, for instance on systems

[2]We use a closed-world assumption, meaning that a semantic state that is not specified in the WSR is considered as False.

Demo	Goal	Starting position	Autonomy level	CATs queried
1	`(on red_mug lab_microwave)` Video frames: Fig. 7 (top row)	Closed microwave and the mug on the table.	Shared control with multiple supervised autonomy triggers during task execution.	`_microwave.open(lab_microwave, edan_arm)`, `_container.grasp(lab_table, red_mug, edan_arm)`, `_microwave.place(lab_microwave, red_mug, edan_arm)`.
2	`(on red_mug lab_table)`. Video frames: Fig. 7 (bottom row).	Closed microwave and mug inside the microwave.	Shared control.	`_microwave.open(lab_microwave, edan_arm)`, `_container.grasp(lab_microwave, red_mug, edan_arm)`, `_table.place(lab_table, red_mug, edan_arm)`.
3	`(not (enclosed lab_microwave))`	Closed microwave.	Full supervised autonomy.	`_microwave.open(lab_microwave, edan_arm)`

TABLE II

OVERVIEW OF THE PLANS GENERATED AND EXECUTED ON EDAN.

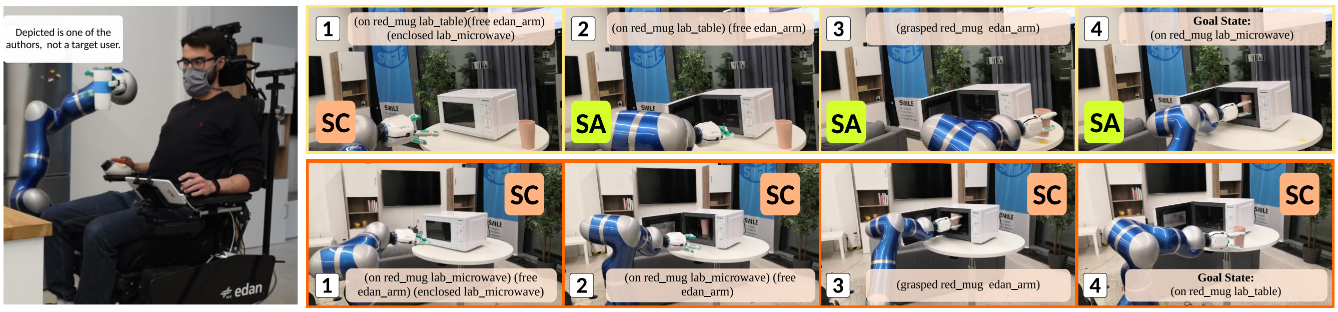

Fig. 7. **Left:** EDAN. **Right:** Two task examples, achieving goals `(on red_mug lab_microwave)` (top) & `(on red_mug lab_table)` (bottom). At every snapshot we annotate the symbolic state, as well as the level of autonomy (shared control [SC], or supervised autonomy [SA]).

	1 mug	**10 mugs**	**40 mugs**
Symbolic plan	0.446 ± 0.021	0.574 ± 0.047	0.837 ± 0.055
SCT generation	0.375 ± 0.037	0.422 ± 0.035	0.570 ± 0.034
Total	0.820 ± 0.037	1.000 ± 0.065	1.407 ± 0.085

TABLE III

TASK PLANNING COMPUTATION TIMES ($\mu \pm \sigma$, TEN RUNS, SECONDS)

that blend the command of the user with an autonomy module [17], [18], research about creating a sequence of shared control tasks towards a goal is rare. While planned shared control systems exist [19], [20], [21], they usually do not plan an explicitly long sequence of actions while providing the option to switch to supervised autonomy.

In the brain-machine interface (BMI) community there are inference systems that leverage knowledge to discover and exploit the user goal while executing a manipulation task [22], [23], and planning systems where the user inputs a goal that the robot plans and later executes by itself [24]. In a similar direction as us, albeit in the domain of wheelchair navigation, Lopes et al. [25] use a hybrid-planning framework to create a grid the user can traverse with a BMI. In the teleoperation literature there has been research in adaptive movement constraints (virtual fixtures) [26] and on-the-fly goal-oriented pedagogical task demonstrators for a space robot [27].

The paradigm for simultaneously creating symbolic high-level and geometric low-level plans in robotics is usually called hybrid planning. There are many practical robotics applications, like in space [28], search-and-rescue [29] and household robotics [30]. On the symbolic side, one of the most famous formalisms for symbolic descriptions of actions is the Planning Domain Definition Language (PDDL) [12]. On the geometric side, our work is inspired by the task frame formalism [31], which modeled robot contacts in terms of frames of interest within a task. This approach is close to the one of Bartels and Beetz [32], where they use an object-frame constraint representation for planning in an autonomous system. Workspace limits and task-related constraints have also been studied by Berenson et al. [33] and Pérez-D'Arpino et al. [34].

VII. CONCLUSION

In this paper we propose the Constraint Action Template framework, which combines Action Templates and Shared Control Templates, with the main aim of enabling task-planning for shared control. To the best of our knowledge, CATs are the first action representation that allows symbolic planning of action sequences that can be used for shared control *and* autonomous execution (using the Automaton [10]), even allowing smooth switches between the two during execution.

This paper has introduced CATs and validated that the plans they generated can be used to successfully complete tasks. In future work, we will use a more extensive set of CATs and operations, and port the missing features from ATs. It was also beyond the scope of this paper to explain how perception can be used to ground the objects and their PDDL symbols under uncertainty. This is being studied in parallel in our robotics institute [35] and the EDAN team [7].

Finally, with a more diverse repertoire of actions and tasks, we will also be able to conduct user studies. Preliminary studies on SCTs and ATs have confirmed their usability in assistive [2] and space robotics [3].

REFERENCES

[1] D. S. Leidner, *Cognitive Reasoning for Compliant Robot Manipulation*, ser. Springer Tracts in Advanced Robotics. Cham: Springer International Publishing, 2019, vol. 127. [Online]. Available: http://link.springer.com/10.1007/978-3-030-04858-7

[2] G. Quere, A. Hagengruber, M. Iskandar, S. Bustamante, D. Leidner, F. Stulp, and J. Vogel, "Shared Control Templates for Assistive Robotics," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, Paris, France, 2020, p. 7.

[3] P. Schmaus, D. Leidner, T. Krüger, A. Schiele, B. Pleintinger, R. Bayer, and N. Y. Lii, "Preliminary insights from the meteron supvis justin space-robotics experiment," *IEEE Robotics and Automation Letters*, vol. 3, no. 4, pp. 3836–3843, 2018. http://dx.doi.org/10.1109/LRA.2018.2856906

[4] J. Vogel, D. Leidner, A. Hagengruber, M. Panzirsch, B. Bauml, M. Denninger, U. Hillenbrand, L. Suchenwirth, P. Schmaus, M. Sewtz, A. S. Bauer, T. Hulin, M. Iskandar, G. Quere, A. Albu-Schaffer, and A. Dietrich, "An ecosystem for heterogeneous robotic assistants in caregiving: Core functionalities and use cases," *IEEE Robotics Automation Magazine*, vol. 28, no. 3, pp. 12–28, 2021. http://dx.doi.org/10.1109/MRA.2020.3032142

[5] D.-J. Kim, R. Hazlett-Knudsen, H. Culver-Godfrey, G. Rucks, T. Cunningham, D. Portee, J. Bricout, Z. Wang, and A. Behal, "How Autonomy Impacts Performance and Satisfaction: Results From a Study With Spinal Cord Injured Subjects Using an Assistive Robot," *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans*, vol. 42, no. 1, pp. 2–14, Jan. 2012. [Online]. Available: http://ieeexplore.ieee.org/document/5941028/. http://dx.doi.org/10.1109/TSMCA.2011.2159589

[6] T. Bhattacharjee, E. K. Gordon, R. Scalise, M. E. Cabrera, A. Caspi, M. Cakmak, and S. S. Srinivasa, "Is More Autonomy Always Better?: Exploring Preferences of Users with Mobility Impairments in Robot-assisted Feeding," in *Proceedings of the 2020 ACM/IEEE International Conference on Human-Robot Interaction*. Cambridge United Kingdom: ACM, Mar. 2020, pp. 181–190. [Online]. Available: https://dl.acm.org/doi/10.1145/3319502.3374818. http://dx.doi.org/10.1145/3319502.3374818

[7] J. Vogel, A. Hagengruber, M. Iskandar, G. Quere, U. Leipscher, S. Bustamante, A. Dietrich, H. Hoeppner, D. Leidner, and A. Albu-Schäffer, "Edan - an emg-controlled daily assistant to help people with physical disabilities," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2020.

[8] M. Iskandar, G. Quere, A. Hagengruber, A. Dietrich, and J. Vogel, "Employing Whole-Body Control in Assistive Robotics," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Macau, China: IEEE, Nov. 2019, pp. 5643–5650. [Online]. Available: https://ieeexplore.ieee.org/document/8967772/. http://dx.doi.org/10.1109/IROS40897.2019.8967772

[9] G. Quere, S. Bustamante, A. Hagengruber, J. Vogel, F. Steinmetz, and F. Stulp, "Learning and interactive design of shared control templates," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2021, pp. 1887–1894. http://dx.doi.org/10.1109/IROS51168.2021.9636047

[10] S. Bustamante, G. Quere, K. Hagmann, X. Wu, P. Schmaus, J. Vogel, F. Stulp, and D. Leidner, "Toward seamless transitions between shared control and supervised autonomy in robotic assistance," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 3833–3840, 2021.

[11] D. Leidner, C. Borst, and G. Hirzinger, "Things are made for what they are: Solving manipulation tasks by using functional object classes," in *2012 12th IEEE-RAS International Conference on Humanoid Robots (Humanoids 2012)*. IEEE, 2012, pp. 429–435.

[12] M. Ghallab, A. Howe, C. Knoblock, D. McDermott, A. Ram, M. Veloso, D. Weld, and D. Wilkins, "PDDL, The Planning Version Domain Definition Language. Version 1.2," Yale Center for Computational Vision and Control, AIPS-98 Planning Competition Committee, Tech. Rep., 1998.

[13] J. Kuffner and S. LaValle, "Rrt-connect: An efficient approach to single-query path planning," in *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No.00CH37065)*, vol. 2, 2000, pp. 995–1001 vol.2. http://dx.doi.org/10.1109/ROBOT.2000.844730

[14] K. Hagmann, A. Hellings-Kuss, J. Klodmann, R. Richter, F. Stulp, and D. Leidner, "A digital twin for contextual assistance in surgical robotics training," *Frontiers in AI and Robotics*, 2021.

[15] M. Helmert, "The fast downward planning system," *J. Artif. Intell. Res.*, vol. 26, pp. 191–246, 2006.

[16] A. L. Carrillo and J. A. Falgueras, "Proposal and testing goals-guided interaction for occasional users," *Human-centric Computing and Information Sciences*, vol. 10, no. 1, 2020. http://dx.doi.org/10.1186/s13673-020-0209-2

[17] A. D. Dragan and S. S. Srinivasa, "A policy-blending formalism for shared control," *The International Journal of Robotics Research*, vol. 32, no. 7, pp. 790–805, Jun. 2013. [Online]. Available: http://journals.sagepub.com/doi/10.1177/0278364913490324. http://dx.doi.org/10.1177/0278364913490324

[18] D. Gopinath, S. Jain, and B. D. Argall, "Human-in-the-loop optimization of shared autonomy in assistive robotics," *IEEE Robotics and Automation Letters*, vol. 2, no. 1, pp. 247–254, Jan. 2017. [Online]. Available: https://doi.org/10.1109/lra.2016.2593928. http://dx.doi.org/10.1109/lra.2016.2593928

[19] W. Zhang, F. Sun, C. Liu, W. Su, C. Tan, and S. Liu, "A hybrid eeg-based bci for robot grasp controlling," in *2017 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, 2017, pp. 3278–3283. http://dx.doi.org/10.1109/SMC.2017.8123134

[20] E. A. M. Ghalamzan, F. Abi-Farraj, P. R. Giordano, and R. Stolkin, "Human-in-the-loop optimisation: Mixed initiative grasping for optimally facilitating post-grasp manipulative actions," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2017, pp. 3386–3393. http://dx.doi.org/10.1109/IROS.2017.8206178

[21] M. Behery, "A knowledge-based activity representation for shared autonomy teleoperation of robotic arms," RWTH-Aachen University, Master's Thesis, 2016.

[22] S. Javdani, H. Admoni, S. Pellegrinelli, S. S. Srinivasa, and J. A. Bagnell, "Shared autonomy via hindsight optimization for teleoperation and teaming," *The International Journal of Robotics Research*, vol. 37, no. 7, pp. 717–742, Jun. 2018. [Online]. Available: http://journals.sagepub.com/doi/10.1177/0278364918776060. http://dx.doi.org/10.1177/0278364918776060

[23] K. Muelling, A. Venkatraman, J.-S. Valois, J. E. Downey, J. Weiss, S. Javdani, M. Hebert, A. B. Schwartz, J. L. Collinger, and J. A. Bagnell, "Autonomy infused teleoperation with application to brain computer interface controlled manipulation," *Autonomous Robots*, vol. 41, no. 6, pp. 1401–1422, Aug. 2017. [Online]. Available: http://link.springer.com/10.1007/s10514-017-9622-4. http://dx.doi.org/10.1007/s10514-017-9622-4

[24] D. Kuhner, L. Fiederer, J. Aldinger, F. Burget, M. Völker, R. Schirrmeister, C. Do, J. Boedecker, B. Nebel, T. Ball, and W. Burgard, "A service assistant combining autonomous robotics, flexible goal formulation, and deep-learning-based brain–computer interfacing," *Robotics and Autonomous Systems*, vol. 116, pp. 98–113, 2019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0921889018302227. http://dx.doi.org/https://doi.org/10.1016/j.robot.2019.02.015

[25] A. Lopes, J. Rodrigues, J. Perdigao, G. Pires, and U. Nunes, "A new hybrid motion planner: Applied in a brain-actuated robotic wheelchair," *IEEE Robotics Automation Magazine*, vol. 23, no. 4, pp. 82–93, 2016. http://dx.doi.org/10.1109/MRA.2016.2605403

[26] D. Aarno, S. Ekvall, and D. Kragic, "Adaptive virtual fixtures for machine-assisted teleoperation tasks," in *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*, 2005, pp. 1139–1144. http://dx.doi.org/10.1109/ROBOT.2005.1570269

[27] K. Belghith, B. Auder, F. Kabanza, P. Bellefeuille, and L. Hartman, "Automatic animation generation of a teleoperated robot arm," in *ECAI 2008*. IOS Press, 2008, pp. 931–932.

[28] J. Martínez-Moritz, I. Rodríguez, K. Nottensteiner, J.-P. Lutze, P. Lehner, and M. A. Roa, "Hybrid planning system for in-space robotic assembly of telescopes using segmented mirror tiles," in *2021 IEEE Aerospace Conference (50100)*, 2021, pp. 1–16. http://dx.doi.org/10.1109/AERO50100.2021.9438399

[29] P. Bechon, C. Lesire, and M. Barbier, "Hybrid planning and distributed iterative repair for multi-robot missions with communication losses," *Autonomous Robots*, vol. 44, pp. 505–531, 2020.

[30] C. R. Garrett, R. Chitnis, R. Holladay, B. Kim, T. Silver, L. P. Kaelbling, and T. Lozano-Pérez, "Integrated task and motion planning," *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 4, no. 1, pp. 265–293, 2021. [Online]. Available: https://doi.org/10.1146/annurev-control-091420-084139. http://dx.doi.org/10.1146/annurev-control-091420-084139

[31] H. Bruyninckx and J. De Schutter, “Specification of force-controlled actions in the ”task frame formalism”-a synthesis,” *IEEE Transactions on Robotics and Automation*, vol. 12, no. 4, pp. 581–589, 1996. http://dx.doi.org/10.1109/70.508440

[32] G. Bartels, I. Kresse, and M. Beetz, “Constraint-based movement representation grounded in geometric features,” in *2013 13th IEEE-RAS International Conference on Humanoid Robots (Humanoids)*. Atlanta, GA: IEEE, Oct. 2013, pp. 547–554. [Online]. Available: http://ieeexplore.ieee.org/document/7030027/. http://dx.doi.org/10.1109/HUMANOIDS.2013.7030027

[33] D. Berenson, S. S. Srinivasa, D. Ferguson, A. Collet, and J. J. Kuffner, “Manipulation planning with workspace goal regions,” in *2009 IEEE International Conference on Robotics and Automation*. IEEE, 2009, pp. 618–624.

[34] C. Pérez-D’Arpino and J. A. Shah, “C-learn: Learning geometric constraints from demonstrations for multi-step manipulation in shared autonomy,” in *2017 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2017, pp. 4058–4065.

[35] J. Feng, M. Durner, Z. Marton, F. Balint-Benczedi, and R. Triebel, “Introspective robot perception using smoothed predictions from bayesian neural networks,” in *International Symposium on Robotics Research (ISRR), 06-10 Oct 2019, Hanoi, Vietnam*, 01 2019.