Toward Achieving Formal Guarantees for Human-Aware Controllers in Human-Robot Interactions

Rachel Schlossman[1], Minkyu Kim, Ufuk Topcu, Luis Sentis

***Abstract*— With the primary objective of human-robot interaction being to support humans' goals, there exists a need to formally synthesize robot controllers that can provide the desired service. Synthesis techniques have the benefit of providing formal guarantees for specification satisfaction. There is potential to apply these techniques for devising robot controllers whose specifications are coupled with human needs. This paper explores the use of formal methods to construct human-aware robot controllers to support the productivity requirements of humans. We tackle these types of scenarios via human workload-informed models and reactive synthesis. This strategy allows us to synthesize controllers that fulfill formal specifications that are expressed as linear temporal logic formulas. We present a case study in which we reason about a work delivery and pickup task such that the robot increases worker productivity, but not stress induced by high work backlog. We demonstrate our controller using the Toyota HSR, a mobile manipulator robot. The results demonstrate the realization of a robust robot controller that is guaranteed to properly reason and react in collaborative tasks with human partners.**

I. INTRODUCTION

As robots become more embedded into our everyday lives and begin to collaborate with humans, a large potential emerges to boost human productivity by eliminating unnecessary human chores in workplaces [1]. This potential can only be realized by robot control systems that process and react to human needs. A survey of human-aware robot navigation shows that many researchers have studied motion and task plan generation for socially-aware robots, for activities ranging from operating in human-occupied areas to engaging in social cues [2]. However, these methods often lack robustness to disturbances and/or formal guarantees of goal achievement. Formal methods have been leveraged in robotic applications, but there is a growing need to apply these techniques to robots that continuously and directly interact with humans. Our work takes a step toward addressing this need.

Reactive synthesis has been applied in contexts where disturbances are unavoidable, such as the DARPA Robotics Challenge [3] and other challenging setups [4], to provide formal guarantees for specification realizability. There is a gap in robotics and formal method literature with respect to applying these methods to direct human-robot interaction (HRI). In reactive synthesis problems, humans are often framed as randomly or periodically interfering with the robot's goal [4], [5]. In [6], a reactive synthesis problem is formulated to generate a policy for a robot to reach a goal position in a simulated kitchen scenario, but the only human interaction involves avoiding two moving chefs. We seek to be robust to a larger variety of disturbances, and to generate policies in which humans and robots continuously interact.

Fig. 1. (a) The Toyota Human Support Robot, which we use as our experimental platform for human-informed work delivery, picking up completed work. (b) The HSR dropping off completed work at the Inventory Station.

There is much interest in the HRI community for robot controllers to consider human factors to boost human productivity [7], [8]. Several research groups are exploring formal verification methods for robots to interactively support humans [9], but few groups have incorporated human factors into these methods. In [10], the authors verify whether a robot assistant can reach commanded positions and deliver medicine. In addition to these types of interactions, we also explore additional knowledge of human requirements to improve collaborative task execution. Ref. [11] takes a step in this direction: A cognitive model of trust is incorporated into a stochastic multi-player game and probabilistic rational temporal logic specifications are proposed, but probabilistic model checking is left as future work. In [12], social norms for a hand-off task are represented as transition systems, and model checking is performed to verify successful task completion. In contrast to this work, our framework uses reactive synthesis to prevent specification violations, while being robust to uncertainties.

Although it is impossible to generalize human behavior, works like [13], [14] still demonstrate the insight to be gained by considering human models for decision-making. For example, in [15], hypothetical human models that consider human proficiency and stress are employed to synthesize paths for semi-autonomous operation of drones with human operators. Similarly, we focus on studying the implications of incorporating human models. Our approach provides the

*This work was supported by NASA Space Technology Research Fellowship 80NSSC17K0188.

[1]rachel.schlossman@utexas.edu

Authors are with The Departments of Mechanical Engineering (R.S., M.K.) or Aerospace Engineering (U.T., L.S.), University of Texas at Austin, Austin, TX 78712-0292, USA

flexibility to update the human model for effective and personalized human-robot interactions.

The goal of this paper is to demonstrate a proof-of-concept for devising control policies via reactive synthesis that consider and improve human working behavior. In doing so, we generate a controller that is robust to disturbances and provides formal guarantees for specification satisfaction in an HRI scenario. The main contribution of our work is a study on reactive synthesis that incorporates human factors for human-aware robot cooperative tasks. We consider an HRI case study in which we construct a model of human workers in a workplace as transition systems to devise a robot controller to deliver and pick up work while considering the human's needs. To this end, we formalize system specifications using linear temporal logic. We use reactive synthesis to automatically construct a controller that meets all system specifications and a human's productivity needs. We then demonstrate the reactive controller on the Toyota Human Support Robot (HSR) (Fig. 1). Ultimately, we explore the question of how robots can make humans more productive by limiting unnecessary human tasks

II. PRELIMINARIES

Our notation employs the formalisms of [16], which are summarized below:

Definition 1: A transition system, TS, is a tuple $TS = (S, Act, \rightarrow, I, AP, L)$ where S is a set of states, Act is a set of actions, $\rightarrow \subseteq S \times Act \times S$ is a transition relation, $I \subseteq S$ is a set of initial states, AP is a set of (Boolean) atomic propositions, and $L : S \mapsto 2^{AP}$ is a labeling function.

Definition 2: An infinite path fragment, $\pi = s_0 s_1 s_2 ...$, for $s_i \in S$, is an infinite sequence of states such that $s_{i+1} \in \{s'_i \in S : \exists \alpha \in Act \mid s_i \xrightarrow{\alpha} s'_i\}\ \forall i \geq 0$. An infinite path fragment is a path if the initial state, $s_0 \in I$. The set of paths in TS is denoted as $Paths(TS)$.

Definition 3: The trace of π, $trace(\pi) = L(s_0)L(s_1)L(s_2)...$, is a sequence of sets of atomic propositions that are true in the states along the path. The set of traces of TS is defined by $Traces(TS) = \{trace(\pi) : \pi \in Paths(TS)\}$

Definition 4: A linear-time (LT) property, P, over atomic propositions in AP, is a set of infinite sequences over 2^{AP}. TS satisfies P, represented by $TS \models P$, iff $Traces(TS) \subseteq P$.

Definition 5: Linear-temporal logic (LTL) is a formal language to represent LT properties. The operators used in this paper to construct LT formulas are conjunction ($\wedge$), disjunction ($\vee$), next ($\bigcirc$), eventually ($\Diamond$), globally ($\Box$), implication ($\rightarrow$), and negation ($\neg$). Let Φ be an LTL formula over AP. TS satisfies Φ, represented by $TS \models \Phi$, iff $\pi \models \Phi$ for all $\pi \in Paths(TS)$.

III. WORK DELIVERY WITH HUMAN BACKLOG MODEL

This case study examines a robot operating in a work environment and is inspired by [17]. The robot drops off new work ("deliverables") at the Human Workstation, picks up completed work from the Human Workstation, and drops the completed work off at the Inventory Station. The robot's contributions thereby eliminate unnecessary movement by the human to pick up and drop off work. The goal is for the robot to operate with an awareness of the human's backlog, defined as the amount of uncompleted work at the Human Workstation. Assessing human backlog is an ongoing area of research [18]. We take backlog to be an indicator of stress levels, as too little work can cause boredom and too much work can result in higher levels of frustration [19]. We seek to synthesize a controller that guarantees, despite system disturbances, that the human always has work to complete and is not over-stressed by work demands.

A. Modeling

1) Human Model: In this scenario, the human is always present in the Human Workstation. (Work breaks are addressed in Sec. IV.) The human's backlog, BL, can range from 0% to 100%, relative to the maximum amount of uncompleted work that can be present in the workstation. When the robot is not present in the Human Workstation, the human works whenenever there is uncompleted work. We consider a simple, discrete linear BL model that is a function of ΔT time steps that each last t_d seconds:

$$BL(\Delta T) = BL_{init} - \gamma \Delta T,\ \Delta T \in \{0, 1, 2, ...\}, \quad (1)$$

where BL_{init} is the initial amount of backlog at the Human Workstation, and γ is the work reduction rate per t_d seconds. The value of BL_{init} is updated each time the robot comes to the Human Workstation to deliver work.

We now formalize the way BL may change between time steps. There is uncertainty in how much BL decreases during each state transition, as the reduction value depends on how much time the robot requires to transition between states in the real-world execution. The worker's BL can decrease by integer multiples of γ each time step. We assume that BL may decrease by up to 5γ, based on the maximum amount of time the robot requires for its most challenging manipulation task. We consider two possibilities for how BL may shrink. When the robot is traveling between locations in the workspace, we define the formula v_1 such that the following holds:

$$\bigcirc v_1 \triangleq \bigvee_{k=0}^{2} (BL - k\gamma). \quad (2)$$

It may require more time for the robot to drop off completed work at the Inventory Station than to travel between locations, and so we allow for greater BL reductions between time steps for this task:

$$\bigcirc v_2 \triangleq \bigvee_{k=0}^{5} (BL - k\gamma). \quad (3)$$

The human transitions from the "work" state to the "wait" state if the human has completed all of her work and the robot has not yet arrived to deliver more work. The human transitions to the "refill" state when the robot is present in the Human Workstation and delivers more work. The robot being

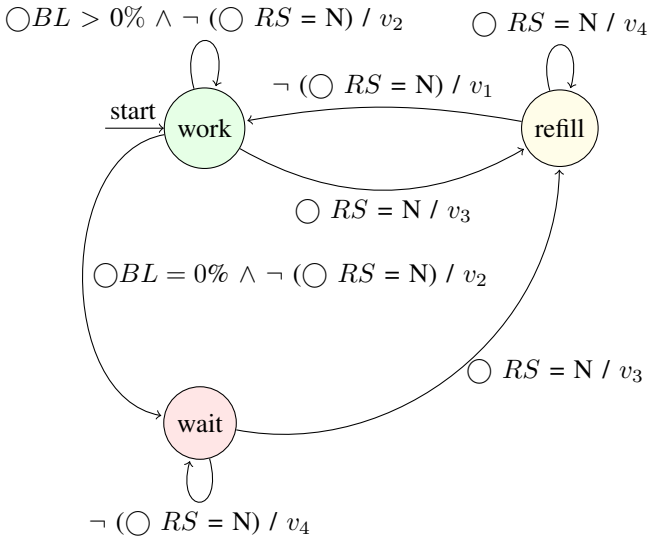

Fig. 2. Human Model with three states. Transitions are triggered by guards, g, and there are corresponding outputs, y. In this transition system, edges are labeled in a g / y format. In this system, the outputs are described by v_2, v_3, and v_4.

in the Human Workstation is equivalent to the robot state, RS, being equal to N. When the robot arrives at the Human Workstation, BL grows by δ. The human then returns to the working state when the robot departs from the workstation. As seen in Fig. 2, there are guards based on BL and robot behaviors which determine allowable state transitions. The BL variable is tracked, and grows and shrinks according to v_1, v_2, v_3, and v_4. The formula v_3 captures BL growth when the robot arrives at the Human Workstation:

$$\bigcirc v_3 \triangleq BL + \delta, \tag{4}$$

and v_4 captures when BL does not change between time steps:

$$\bigcirc v_4 \triangleq BL. \tag{5}$$

Fig. 2 demonstrates the possible non-determinism in how BL changes with each time step, and planning for this uncertainty is discussed further in Sec. III-B.

2) Robot Model: The robot moves in a 1D grid. There are N+1 grid spaces, with the grid spaces labeled as 0 through N. Space 0 corresponds to the Inventory Station, and (3) is valid in this location when the robot drops off completed work. As discussed in the previous section, space N is the Human Workstation. The robot's actions are Go_{Sj}, which indicate that the robot is currently moving to position j ∈ 0,1,...,N in the grid. The robot is free to move within this grid, except for when there is an obstacle present in a grid space blocking a path. We define the atomic proposition, "an obstacle is present in State j," as $\mathcal{O}_j$ $\forall j$ = 1,2,...,N-1. The transition system is shown in Fig. 3.

3) Full HRI System Model: A transition system is then formulated to capture that the robot drops off deliverables and picks up completed work at the Human Workstation, and drops off completed work at the Inventory Station. The robot also operates with an awareness of the human's

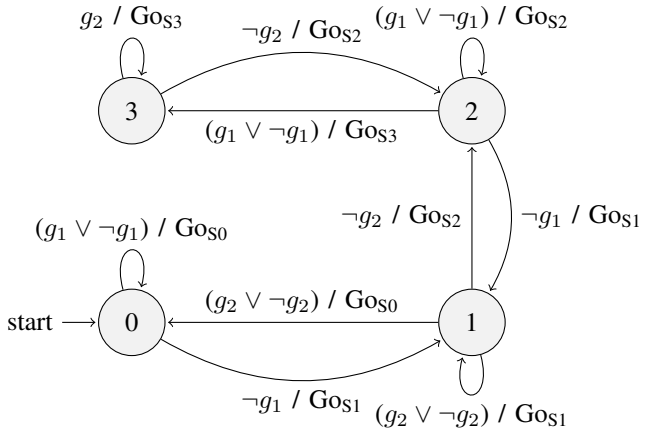

Fig. 3. Robot Model with N = 3. State 0 is the Inventory Station and State 3 is the Human Workstation. Each edge of the TS is labeled with a guard and an action. The presence of an obstruction in one of the robot's adjacent positions can restrict the robot's next action. The guards express whether or not there is an obstacle blocking the robot from proceeding to a neighboring state, and we define $g_1 \triangleq \mathcal{O}_1$ and $g_2 \triangleq \mathcal{O}_2$

work backlog, which will allow for controller synthesis that considers BL. To combine and synchronize the human and robot systems, the two separate human and robot models were used to create states which represent both the human and robot at each time step. The transition system, shown in Fig. 4, is expressed as $TS_{HRI} = (S_1, Act_1, \rightarrow, I_1, AP_1, L_1)$ where:

- S_1 = $\{j_{work}, j_{wait}, N_{refill}\}$ $\forall$ j = 0,1,...,N-1
- Act_1 = { Go_{Sj} } $\forall$ j = 0,1,...,N
- I_1 = $\{0_{work}\}$
- $L_1(0_{work})$ = $L_1(0_{wait})$ = Robot is at Inventory Station.
- $L_1(N_{refill})$ = Robot is at Human Workstation.

B. Reactive Synthesis

It is necessary to incorporate robustness to uncertainty in our approach in order for the robot to pick up completed work, drop off deliverables, and reason about the human's BL in a real environment. We consider a two-player game in which the robot's actions are controllable. Obstacle interference, success of dropping off completed work, and the backlog reduction rate act as the uncontrollable environment. The robot and the environment take turns executing actions, and we seek to automatically synthesize a robot controller strategy that allows a system specification to be realizable despite any antagonistic actions executed by the environment. To meet the system requirements while handling external disturbances, we formulate this scenario as a reactive synthesis problem in which plant actions are controllable and environment actions are uncontrollable. We seek to find a strategy that will uphold a specification no matter how the environment selects its actions for all time [20].

To automatically synthesize a controller, we must formalize the specifications that describe the possible environment behaviors. It is possible in a work environment that people may pass through states S_d = $\{j_{work}, j_{wait}\}$ $\forall j : 0 < j < N$ and obstruct the robot from proceeding into an adjacent position in the workspace. We impose as a safety

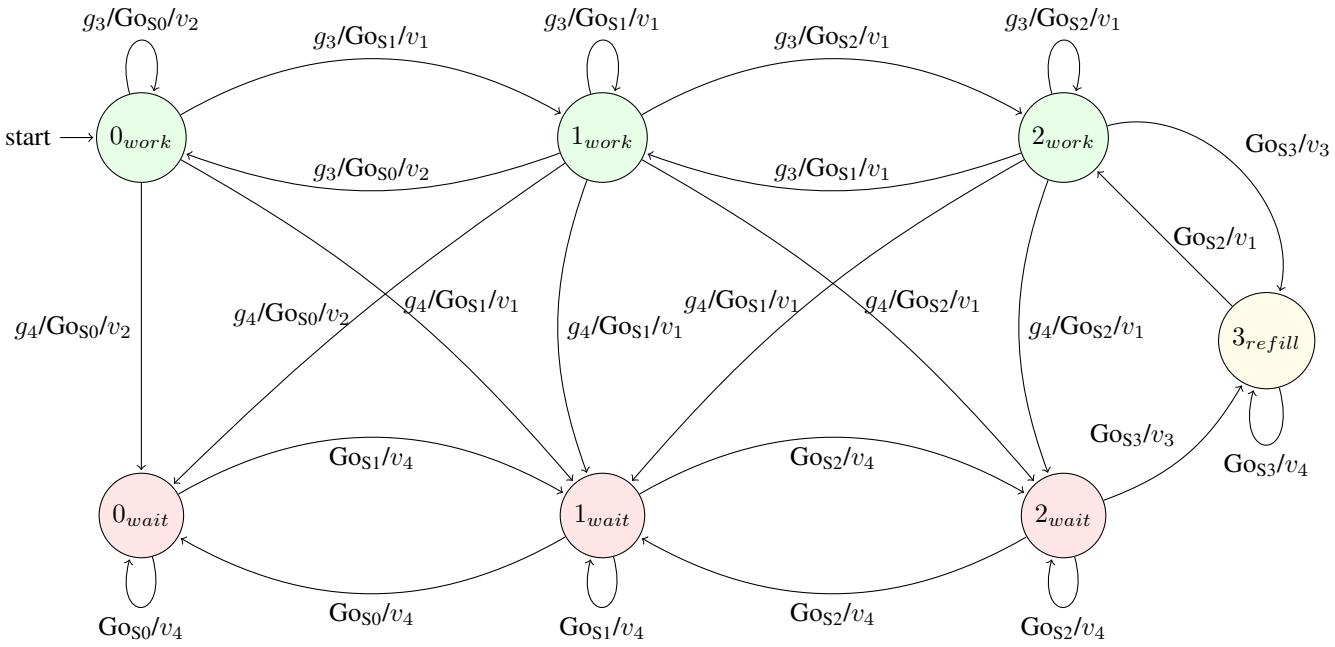

Fig. 4. HRI Work Delivery and Pickup Transition System with N = 3. The edges of the transition system are labeled first by guards (g_3, g_4), then by actions, and lastly by pertinent output variables (v_1, v_2, v_3, v_4). The guards which determine whether the human is working or waiting are expressed by $g_3 \triangleq \bigcirc BL > 0\%$ and $g_4 \triangleq \bigcirc BL = 0\%$. As illustrated in Fig. 3, the robot cannot move to a neighboring state if it contains an obstacle, but we do not show this guard due to space limitations.

specification on the robot that it will not proceed toward a position that contains an obstacle. We also assume that if the human sees the robot moving to a workspace position, she will not intentionally move to block the robot's desired workspace position:

$$\Phi_{d1} \triangleq \bigcirc RS = j \rightarrow \bigcirc \neg \mathcal{O}_j \quad \forall j = 1,2,..\text{N-1}. \tag{6}$$

In our implementation, if there is a human occupying one of the robot's neighboring positions, the robot will ask her not to block the workspace. Thus, we assume that the person will move out of the way by the next time step:

$$\Phi_{d2} \triangleq \mathcal{O}_j \rightarrow \bigcirc \neg \mathcal{O}_j \quad \forall j = 1,2,..\text{N-1}. \tag{7}$$

We also account for the possibility that the robot may not successfully drop off completed work in the Inventory Station on its first try. At all robot states, the robot either is or is not manipulating completed work ("hand full" $\triangleq HF$ is true or false). When $RS = 0$, if the robot is currently manipulating completed work, it will be successful ($\mathcal{S}$) or unsuccessful ($\neg\mathcal{S}$) at dropping off the completed work in that time step. In order to prevent the environment from interfering indefinitely with a successful dropoff, we assume that no more than two consecutive tries are required to be successful. The associated specifications are written as follows:

$$\Phi_{d3} \triangleq (\neg(RS = 0) \wedge \bigcirc RS = 0 \wedge HF) \rightarrow \bigcirc (HF \wedge \text{tries} = 1 \wedge (\neg\mathcal{S} \vee \mathcal{S})), \tag{8}$$

$$\Phi_{d4} \triangleq (RS = 0 \wedge HF \wedge \neg\mathcal{S} \wedge \text{tries} = 1) \rightarrow \bigcirc (HF \wedge \text{tries} = 2 \wedge \mathcal{S}), \text{ and} \tag{9}$$

$$\Phi_{d5} \triangleq (RS = 0 \wedge HF \wedge \mathcal{S}) \rightarrow \bigcirc(\neg HF). \tag{10}$$

As discussed in Sec. III-A.1, when the robot is not dropping off deliverables, there is uncertainty in how much BL will decrease as the robot transition between states in the real environment. This uncertainty will impact the value of BL at the next time step when the human is working. To express the specifications for BL reduction, we define a formula that describes the situations in which the robot will attempt to drop off completed work:

$$g_4 \triangleq \left[\{\neg(RS = 0) \wedge \bigcirc RS = 0\} \vee (\text{tries} = 1 \wedge \neg\mathcal{S})\right] \wedge HF. \tag{11}$$

We now distinguish between the robot moving within the workspace and performing the dropoff behavior. For workspace motions, we define

$$\Phi_{d6} \triangleq \neg(\bigcirc RS = \text{N}) \wedge \neg g_4 \wedge \neg\text{wait} \rightarrow \bigcirc v_1, \tag{12}$$

where v_1 is as defined in (2). During dropoff at the Inventory Station the following specification holds:

$$\Phi_{d7} \triangleq g_4 \wedge \neg\text{wait} \rightarrow \bigcirc v_2, \tag{13}$$

where v_2 is as defined in (3). We desire that the robot always eventually drops off completed work at the Inventory Station. Unless BL decreases, there would be no guarantee that the robot would always eventually have completed work to pick up from the human and drop off at the Inverntory Station. We add the assumption that if BL stays constant in two consecutive time steps, then BL will decrease in the next time step:

$$\Phi_{d8} \triangleq \neg(\bigcirc RS = \text{N}) \wedge v_4 \wedge \neg\text{wait} \rightarrow \bigcirc \left(\bigvee_{k=1}^{5} (BL - k\gamma)\right), \tag{14}$$

where v_4 is as defined in (5). We synchronize the real robot's motion with the BL model so that this assumption is valid in our implementation.

C. Controller Synthesis

The reactive synthesis problem was implemented using Slugs [21] with N = 3. (We consider four states for our proof-of-concept hardware implementation in Sec. IV, but the underlying techniques of our approach can handle a much larger number of states.) By formulating the problem as a two-player game, Slugs can construct a reactive robot controller that upholds our specification of interest within TS_{HRI}.

In the simulation, the robot starts at State 0, $\gamma = 3.3\%$, and $\delta = 50\%$. In order to strike a balance between state space fineness and computational efficiency, BL is represented in Slugs as 0,1,2,...,30, which corresponds to 0%,3.3%,6.7%,...,100%. Slugs was used to synthesize a controller that always satisfies the specification that the human's backlog never reaches 0% and never exceeds 87%. In this manner, the human always has work to complete, but the robot does not seek to stress her. It is also desired that the robot will return to the Inventory Station infinitely often to drop off completed work. Since BL can vary from 0 to 30 in Slugs, we express the system specification as:

$$\Phi_1 = (\square BL \leq 26) \wedge (\square BL > 0) \wedge \square\lozenge(RS = 0 \wedge HF) \tag{15}$$

We provide as an initial condition that BL_{init} is between 30% and 86.7% ($9 \leq BL_{init} \leq 26$), as we found that outside of this BL_{init} range, (15) is not realizable. We synthesize a strategy using a quad-core Intel Core i7 processor and 12GB of RAM. Slugs computes in less than five seconds that the specification is realizable, and devises a high-level controller that guarantees that the robot will react properly to its environment while upholding the system specification. The controller is in the form of a decision tree, with nodes that capture all possible combinations of robot and environment behaviors, and the possible transitions from each node. Based on the robot's present state and the environment's behavior, the decision tree determines the appropriate next robot action. We now have a policy that can be leveraged for online decision-making.

IV. Experiments

A. Toyota Human Support Robot

The Toyota Human Support Robot (HSR), which comprises an omni-directional base and a 5-DOF single manipulator, was adopted as the hardware platform for experimentation. The HSR uses two different computers: The main PC is for primary perception, navigation, and manipulation tasks. An Alienware laptop (Intel Core i7-7820HK, GTX 1080) is used for running OpenPose[1], a real-time convolution neural network based algorithm used for human detection. All robot sub-programs communicate with each other via the Robot Operating System (ROS) interface. An overview of the HSR skills used for controller implementation is provided below.

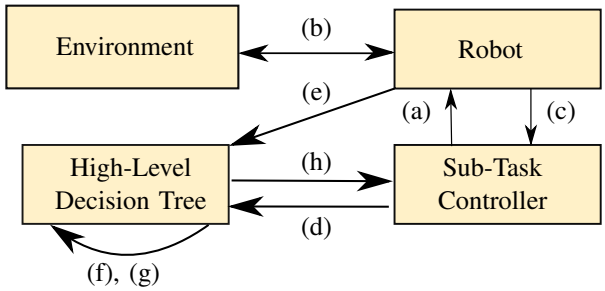

Fig. 5. The communication protocols between the the robot and the sub-task (SMACH) and high-level (Slugs) controllers. The systems communicate by repeating (a)-(h), where: (a) Track and command sequential tasks; (b) Perceive, navigate, and manipulate; (c) Sequential tasks complete; (d) Request next action; (e) Subscribe to ROS environment topics; (f) Check for human at workstation. Update RS and BL when worker is present; (g) Select next action from look-up table. (h) Respond with next action.

1) Perception: A laser range scanner is used on both sides of the mobile base to detect whether there is an obstacle within approximately one meter of the base. To simulate a more realistic working environment, the robot also perceives if there is a human in the workstation or if she has left to take a break. A depth camera for RGB-D video streaming is located on the HSR's head. Recognition of whether or not there is a human in the workstation, based on the RBG-D data, is executed by OpenPose. We created a ROS action so that the HSR turns its head toward the workstation every five seconds to check for worker presence.

2) Navigation: All basic navigation functions, including wheel-joint control and avoiding obstacles, are included in the ROS navigation stack. It is assumed that given a goal position, the robot can safely navigate to this location, while avoiding dynamic obstacles via a re-planning scheme.

3) Manipulation: The robot's manipulator is used to pick up completed work from the human, and to drop off completed work at the Inventory Station. For pickup, the robot moves its end effector near to the human's right hand so that the human can hand over her work. To distinguish between $\mathcal{S}$ and $\neg\mathcal{S}$ during dropoff, as discussed in Sec. III-B, we use a force sensor mounted at the end effector to judge whether or not the object successfully made contact with the counter.

B. Controller Implementation

The decision tree produced by Slugs serves as an online look-up table during robot operation. After transitioning to the next commanded state, the HSR will update its knowledge of the environment (mainly, any obstacles, if a dropoff action was successful, and the human's current BL). We used SMACH[2] to implement a finite-state machine framework that bridges the gap between the high-level action policy from Slugs and the robot's lower-level sequential task executors.

Once the Slugs planner determines the next desired action, the robot's required skills are executed sequentially by the sub-task controller that is associated with a particular SMACH state. In other words, the sub-task control layer is

[1]https://github.com/CMU-Perceptual-Computing-Lab/openpose

[2]http://wiki.ros.org/smach

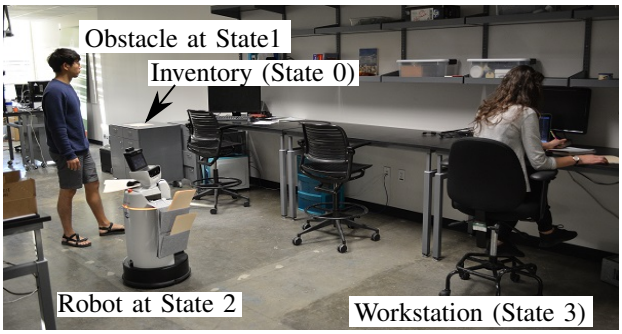

Fig. 6. Experimental setup with four positions in the workspace, corresponding to RS = 0 (Inventory Station), 1, 2, and 3 (Human Workstation). The HSR transports deliverables in its satchel and carries completed work in its manipulator back to the Inventory Station. A human obstructs its path, causing it to remain in State 2 until the next time step, by which time the human will have departed.

responsible for decomposing the desired Slugs action into the sequential, lower-level required skills. For example, when the robot has completed work, the action Go_{S0} first contains navigation (move to counter at State 0), then manipulation (drop off object), and finally navigation (move back from counter) skills. The sequence of behaviors thus requires recognition of when the previous sub-task succeeds. All robot sub-tasks are programmed with the structure of ROS actions in order to be flexible to sub-task execution times. Once a high-level action is completed, or the first dropoff try is unsuccessful, SMACH requests the next desired action from the Slugs planner. The system architecture is shown in Fig. 5.

C. Results

Through experimentation, we sought to verify that our automatically-synthesized, high-level controller properly reacts to its environment while maintaining (15). Fig. 6 shows the layout of our experimental setup and the positions of States 0 through 3. Referring to the BL model in (1), we take $t_d = 10s$.

To test the robustness of the controller, we allowed the HSR to operate autonomously for 30 minutes. During this time, the robot reacted to obstacles, interacted with the worker at the workstation, and returned to the Inventory Station several times to drop off completed work, as shown in Fig 7. While we did not account for the human taking a break in our reactive synthesis problem, we incorporated this consideration for our experiments. If the HSR does not sense a human at the workstation, the HSR waits until she reappears, and then proceeds with its actions according to the Slugs planner.

It is also of interest to investigate the high-level controller behavior for differing BL_{init} values. Fig. 8 shows the results with BL_{init} = 30% ($BL_{init} = \frac{9}{30}$ in Slugs).[3] We also highlight whether the robot is manipulating completed work, and the implementation of the work dropoff logic at State 0. Fig. 9 shows the robot's behavior with BL_{init} = 86.7%

[3]This experiment is shown in the included video, which is also available at https://youtu.be/My6WIiZZCsM.

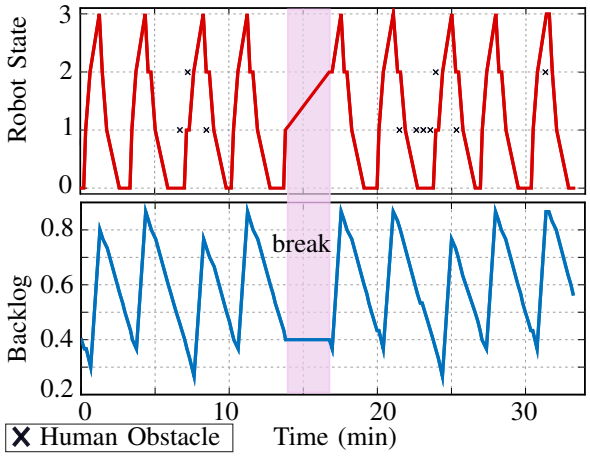

Fig. 7. Robustness test with BL_{init} = 40%. Human obstacles that appear in States 1 and 2 and depart by the next time step are marked by black x's. The worker moves out of the workspace for three minutes, which is highlighted in pink. The robot waits for the human to return at $RS = 2$, but does not update RS and the environment variables in the Slugs planner until the human returns to work.

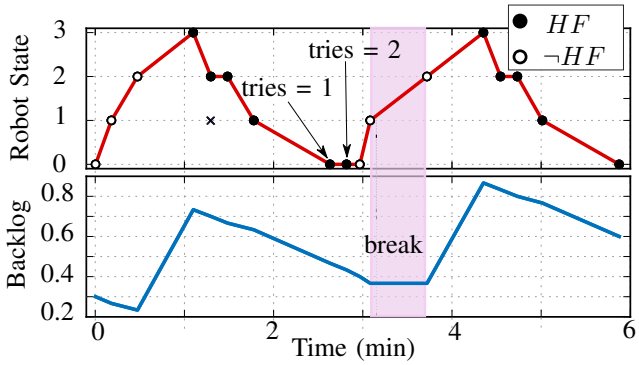

Fig. 8. Robot behavior with BL_{init} = 30%. When $RS = 1$ and HF at 1.75 minutes, the SMACH sub-task controller tracks the amount of time the robot takes to travel from State 1 to State 0 and execute the dropoff sub-tasks, during which time, tries = 1. If this time exceeds 35 seconds, $\neg\mathcal{S}$ is communicated to the high-level controller. The Slugs planner updates the tries value to be equal to 2, and commands that the robot continue to execute its dropoff sub-tasks in the next time step. After the second try, the robot successfully drops off the work, and the robot's manipulator is empty again.

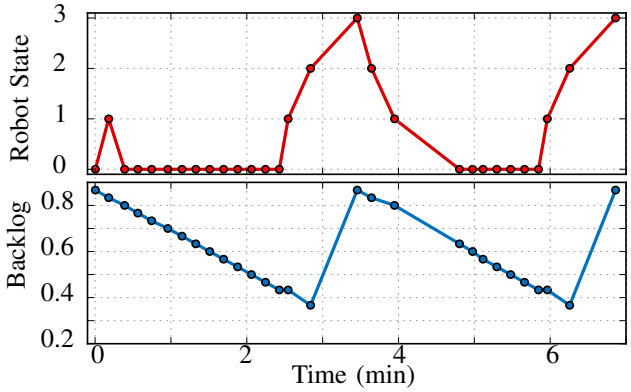

Fig. 9. Robot behavior with BL_{init} = 86.7%. The circular markers indicate the times at which the sub-task controller calls the high-level decision tree to request the next robot action. Excluding its initial movement to and from State 1, the robot elects to wait at State 0 until it moves to deliver work, even when there are no obstacles present.

($BL_{init} = \frac{26}{30}$ in Slugs). We note that the robot first moves to State 1, moves back to State 0, and remains at this position until it proceeds to the workstation to provide deliverables. The initial movement to and from State 1 upholds a winning strategy in the two-player game, but provides no useful output. Additional specifications could be imposed in the reactive synthesis problem to incorporate robot energy efficiency as a consideration to eliminate unnecessary robot movements.

In all three experiments, the robot successfully maintains human BL needs and can autonomously reason about and react to its environment. In the situations considered, the BL value reaches 86.7%, but does not exceed 87%, as desired. It is interesting to note that the controller strategies drive the robot to wait in State 0 until it moves to deliver new work. The robot does not wait in States 1 or 2 for extended periods of time. This behavior is sufficient to satisfy system specifications, but optimizing where the robot waits could be needed in other work environments. For the experimental setup considered, we have verified that our proposed controller offers robustness and flexibility for a real-time, human-centered application.

V. Discussion

Human factors are often neglected in formal methods when devising robot controllers, at the expense of having no formal guarantees that the robot can actually realize a human-centered goal. In this paper, we leverage reactive synthesis to devise a controller that supports a human's and a robot's collaborative goal. By formulating a work delivery scenario as a two-player game, we automatically synthesize a controller that considers a human's work needs and supports robustness to system disturbances. Experimental validation on the HSR hardware demonstrates that the high-level controller strategy enables the robot to operate autonomously and robustly, while considering the activities and productivity of the human worker.

There are several interesting areas remaining for future work. The human backlog model in this case study is a toy model used for proof-of-concept. In the future, we are interested to incorporate verified, dynamic human models based on psychology and cognitive theory, to consider how factors such as fatigue, training, motivation, and stress impact backlog. We are also interested to study how we may extend our work to produce not only feasible motions, but also optimal motions. For now, our results support the feasibility of designing robots that are formally guaranteed to reduce the human share of work activities in the execution of everyday tasks.

References

[1] J. Sutherland and B. Bennett, "The seven deadly wastes of logistics: applying toyota production system principles to create logistics value," *White paper*, vol. 701, pp. 40–50, 2007.

[2] T. Kruse, A. K. Pandey, R. Alami, and A. Kirsch, "Human-aware robot navigation: A survey," *Robotics and Autonomous Systems*, vol. 61, no. 12, pp. 1726–1743, 2013.

[3] S. Maniatopoulos, P. Schillinger, V. Pong, D. C. Conner, and H. Kress-Gazit, "Reactive high-level behavior synthesis for an atlas humanoid robot," in *Robotics and Automation (ICRA), 2016 IEEE International Conference on*. IEEE, 2016, pp. 4192–4199.

[4] K. He, M. Lahijanian, L. E. Kavraki, and M. Y. Vardi, "Reactive synthesis for finite tasks under resource constraints," in *Intelligent Robots and Systems (IROS), 2017 IEEE/RSJ International Conference on*. IEEE, 2017, pp. 5326–5332.

[5] Y. Zhao, U. Topcu, and L. Sentis, "High-level planner synthesis for whole-body locomotion in unstructured environments," in *Decision and Control (CDC), 2016 IEEE 55th Conference on*. IEEE, 2016, pp. 6557–6564.

[6] Y. Wang, N. T. Dantam, S. Chaudhuri, and L. E. Kavraki, "Task and motion policy synthesis as liveness games," in *ICAPS*, 2016, p. 536.

[7] A. Ramachandran, C.-M. Huang, and B. Scassellati, "Give me a break!: Personalized timing strategies to promote learning in robot-child tutoring," in *Proceedings of the 2017 ACM/IEEE International Conference on Human-Robot Interaction*. ACM, 2017, pp. 146–155.

[8] M. Chen, S. Nikolaidis, H. Soh, D. Hsu, and S. Srinivasa, "Planning with trust for human-robot collaboration," in *Proceedings of the 2018 ACM/IEEE International Conference on Human-Robot Interaction*. ACM, 2018, pp. 307–315.

[9] B. Wu, B. Hu, and H. Lin, "Toward efficient manufacturing systems: A trust based human robot collaboration," in *2017 American Control Conference (ACC)*. IEEE, 2017, pp. 1536–1541.

[10] M. Webster, C. Dixon, M. Fisher, M. Salem, J. Saunders, K. L. Koay, K. Dautenhahn, and J. Saez-Pons, "Toward reliable autonomous robotic assistants through formal verification: a case study," *IEEE Transactions on Human-Machine Systems*, vol. 46, no. 2, pp. 186–196, 2016.

[11] M. Kwiatkowska, "Cognitive reasoning and trust in human-robot interactions," in *International Conference on Theory and Applications of Models of Computation*. Springer, 2017, pp. 3–11.

[12] D. Porfirio, A. Sauppé, A. Albarghouthi, and B. Mutlu, "Authoring and verifying human-robot interactions," in *The 31st Annual ACM Symposium on User Interface Software and Technology*. ACM, 2018, pp. 75–86.

[13] D. Sadigh, S. S. Sastry, S. A. Seshia, and A. Dragan, "Information gathering actions over human internal state," in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2016, pp. 66–73.

[14] M. Rausch, A. Fawaz, K. Keefe, and W. H. Sanders, "Modeling humans: A general agent model for the evaluation of security," in *International Conference on Quantitative Evaluation of Systems*. Springer, 2018, pp. 373–388.

[15] L. Feng, C. Wiltsche, L. Humphrey, and U. Topcu, "Synthesis of human-in-the-loop control protocols for autonomous systems," *IEEE Transactions on Automation Science and Engineering*, vol. 13, no. 2, pp. 450–462, 2016.

[16] C. Baier and J.-P. Katoen, *Principles of model checking*. MIT press, 2008.

[17] S. J. Jorgensen, O. Campbell, T. Llado, D. Kim, J. Ahn, and L. Sentis, "Exploring model predictive control to generate optimal control policies for hri dynamical systems," *arXiv preprint arXiv:1701.03839*, 2017.

[18] J. Heard, C. E. Harriott, and J. A. Adams, "A survey of workload assessment algorithms," *IEEE Transactions on Human-Machine Systems*, no. 99, pp. 1–18, 2018.

[19] W. MacDonald, "The impact of job demands and workload on stress and fatigue," *Australian Psychologist*, vol. 38, no. 2, pp. 102–117, 2003.

[20] N. Piterman, A. Pnueli, and Y. Saar, "Synthesis of reactive (1) designs," in *International Workshop on Verification, Model Checking, and Abstract Interpretation*. Springer, 2006, pp. 364–380.

[21] R. Ehlers and V. Raman, "Slugs: Extensible gr (1) synthesis," in *International Conference on Computer Aided Verification*. Springer, 2016, pp. 333–339.