structures and use the annotation of plans for monitoring and failure handling.

$Loc(obj, loc)$	The location of an object
$Loc(Robot, loc)$	The location of the robot
$ObjectVisible(obj)$	The object is visible to the robot
$ObjectInHand(obj)$	The object is carried by the robot
$ObjectPlacedAt(obj, loc)$	The object has been placed at location.
$TaskStatus(task, status)$	The status of a task.

TABLE III

OCCASION STATEMENTS.

Asserting Intentions. In order to infer the intentions of a plan we have to consider the interpretation stack more carefully. Achieving a state *s* has been an intention if the routine *Achieve(s)* that was on the interpretation stack during the execution. The robot pursued the goal *Achieve(s)* in the interval between the start and the end of the corresponding task. The purpose of achieving *s* can be computed by contemplating the supertasks of *Achieve(s)*. Thus, if we want to answer if there has been a task that navigated the robot in order to grasp an object, we state:

$$\begin{aligned} &Task(tsk) \ \wedge \ TaskGoal(tsk, Achieve(Loc(Robot, loc))) \\ &\wedge \ Task(super) \ \wedge \ Subtask^{+}(super, task) \\ &\wedge \ TaskGoal(super, Achieve(ObjectInHand(obj))) \end{aligned}$$

VI. EVALUATION

In this paper we have explained a reasoning mechanism that equips a robot control program with additional functionality and information that can be used to explain past action scenarios but that is also available at runtime. The power of our approach lies in the number of different questions that can be answered, i.e. the number of Prolog clauses that can be defined over the predicates introduced above. In the following, we illustrate the expressiveness by couple of examples.

As a basic query, we might ask the robot if it grasped any green object, by this statement:

$$\begin{aligned} &Task(tsk) \ \wedge \ TaskGoal(tsk, Achieve(ObjectInHand(obj))) \\ &\wedge \ TaskEnd(tsk, t) \\ &\wedge \ DesignatorPropertyAt(obj, Color(Green), t) \end{aligned}$$

Please note that this does not imply that it was specified in the plan that the robot should grasp a green object. The reason is that before grasping it, the robot searches for the object and asserts additional properties if the perception routines provide the corresponding information. That explains also why we need a time parameter in the predicate *DesignatorPropertyAt* since the description matching a specific object may change during the course of action. For instance, a cup can be filled at the beginning but empty later on.

Other queries we can answer include whether the robot manipulated the same object twice. For instance, if the robot is to achieve that one cup is on the table and another cup is on the counter, we can detect the failure that it first placed a cup on the table and then put the same cup on the counter. Obviously, this is unwanted behavior that cannot be detected by program exceptions in a general way. With our first-order representation, this can be expressed by asserting an error if an occasion that has been achieved by a top level task does not hold at the end:

$$\begin{aligned} &UnachievedGoalError(o) \ \Leftrightarrow \\ &\quad TopLevel(tt) \ \wedge \ Task(tsk) \wedge \ Subtask(tt, tsk) \\ &\quad \wedge \ TaskGoal(tsk, o) \ \wedge \ TaskEnd(tt, t) \\ &\quad \wedge \ \neg Holds(o, t) \end{aligned}$$

The predicate *TopLevel* is defined to unify the task that is not the subtask of any other task. Please not that we use *SubTask* instead of $SubTask^{+}$ to denote only direct subtasks of the toplevel. Recording the execution log has no negative influence on the performance of plan execution. That means although all information that is necessary to completely reconstruct plan execution is saved, execution time is not increased noticeable. Answering complex queries that involve data structures that change frequently, such as the location of the robot, take less than 5 seconds. By careful implementation of the predicates it is even possible to do light weight inferences within fast control loops. More time consuming inferences can be made at runtime without loss of time if they are parallelized, for instance if they are executed while the robot is navigating to a location.

VII. CONCLUSIONS

In this paper, we presented an extension to CRAM that implements high performance and accurate reasoning mechanisms. These allow to answer complex questions about plan execution, the intention of the robot, the reason for failures and the belief state of the robot. This is achieved by creating an extensive execution trace and mechanisms to query it through a first-order representation. Since this system can not only be used offline but also during plan execution, i.e. within control routines, it enables deep and complex failure handling mechanisms that are based on descriptions of failures in the first-order representation.

Acknowledgements: The research reported in this paper is supported by the cluster of excellence COTESYS (Cognition for Technical Systems, *www.cotesys.org*).

REFERENCES

[1] M. Beetz, L. Mösenlechner, and M. Tenorth, "CRAM – A Cognitive Robot Abstract Machine for Everyday Manipulation in Human Environments," in *IEEE/RSJ International Conference on Intelligent RObots and Systems.*, 2010, accepted for publication.

[2] R. Simmons, D. Goldberg, A. Goode, M. Montemerlo, N. Roy, B. Sellner, C. Urmson, M. Bugajska, M. Coblenz, M. Macmahon, D. Perzanowski, I. Horswill, R. Zubek, D. Kortenkamp, B. Wolfe, T. Milam, and B. Maxwell, "Grace: An autonomous robot for the aaai robot challenge," 2002.

[3] L. Sterling and E. Shapiro, *The Art of Prolog: Advanced Programming Techniques*. MIT Press, 1994.

[4] M. Beetz, "Structured Reactive Controllers," *Journal of Autonomous Agents and Multi-Agent Systems. Special Issue: Best Papers of the International Conference on Autonomous Agents '99*, vol. 4, pp. 25–55, March/June 2001.

[5] M. Beetz and D. McDermott, "Declarative goals in reactive plans," in *First International Conference on AI Planning Systems*, J. Hendler, Ed., Morgan Kaufmann, 1992, pp. 3–12.

[6] P. Doherty, J. Gustafsson, L. Karlsson, and J. Kvarnstrom, "Temporal action logics (tal): Language specification and tutorial," 1998.

[7] M. Tenorth and M. Beetz, "KnowRob — Knowledge Processing for Autonomous Personal Robots," in *IEEE/RSJ International Conference on Intelligent RObots and Systems.*, 2009.

[8] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Ng, "Ros: an open-source robot operating system," in *In IEEE International Conference on Robotics and Automation (ICRA 2009)*, 2009.