Assembly Planning in Cluttered Environments through Heterogeneous Reasoning

Daniel Beßler[1], Mihai Pomarlan[1], Aliakbar Akbari[2], Muhayyuddin[2], Mohammed Diab[2], Jan Rosell[2], John Bateman[1], Michael Beetz[1] *

[1]Universität Bremen, Bremen, Germany
[2]Universitat Politècnica de Catalunya, Barcelona, Spain

Abstract. Assembly recipes can elegantly be represented in description logic theories. With such a recipe, the robot can figure out the next assembly step through logical inference. However, before performing an action, the robot needs to ensure various spatial constraints are met, such as that the parts to be put together are reachable, non occluded, etc. Such inferences are very complicated to support in logic theories, but specialized algorithms exist that efficiently compute qualitative spatial relations such as whether an object is reachable. In this work, we combine a logic-based planner for assembly tasks with geometric reasoning capabilities to enable robots to perform their tasks under spatial constraints. The geometric reasoner is integrated into the logic-based reasoning through decision procedures attached to symbols in the ontology.

1 Introduction

Robotic tasks are usually described at a high level of abstraction. Such representations are compact, natural for humans for describing the goals of a task, and at least in principle applicable to variations of the task. An abstract "pick part" action is more generally useful than a more concrete "pick part from position x", as long as the robot can locate the target part and reach it.

Robotics manipulation problems, however, may involve many task constraints related to the geometry of the environment and the robot, constraints which are difficult to represent at a higher level of abstraction. Such constraints are, for example, that there is either no direct collision-free motion path or feasible configuration to grasp an object because of the placement of some other, occluding object. Recently, much research has been centred on solving manipulation problems using geometric reasoning, but there is still a lack of incorporating the geometric information inside higher abstraction levels.

In this paper, we look at the task of robotic assembly planning, which we approach, at the higher abstract level, in a knowledge-enabled way. We use an

* This work was partially funded by Deutsche Forschungsgemeinschaft (DFG) through the Collaborative Research Center 1320, EASE, and by the Spanish Government through the project DPI2016-80077-R. Aliakbar Akbari is supported by the Spanish Government through the grant FPI 2015.

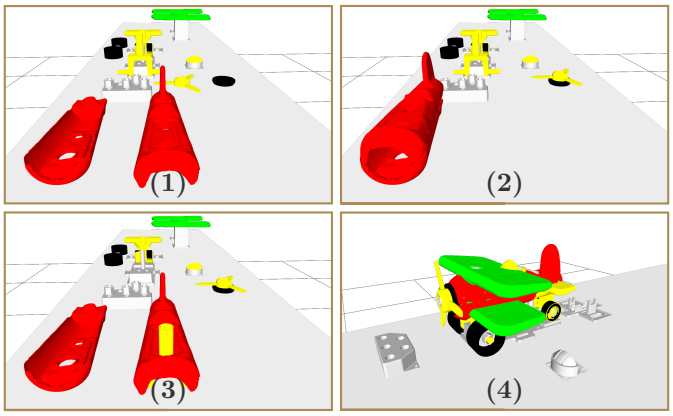

Fig. 1: Different initial workspace configurations of a toy plane assembly (1-3), and the completed plane assembly (4).

assembly planner based on formal specifications of products to be created, parts they are to be created from, and mechanical connections to be formed between them. At this level we represent what affordances a part must provide, in order for it to be able to enter a particular connection or be grasped in a certain way, as well as model that certain grasps and connections block certain affordances. The planning itself proceeds by comparing individuals in the knowledge base with their terminological model, finding inconsistencies, and producing action items to resolve these. For example, if the asserted type of an entity is "Car", the robot can infer that, to be a car, this entity must have some wheels attached, and if this is not the case, the planner will create action items to add them.

In our previous work, the various geometrically motivated constraints pertaining, for example, what grasps are available on a part depending on what mechanical connections it has to other parts, were modelled symbolically. We added axioms to the knowledge base that assert that a connection of a given type will block certain affordances, thus preventing the part to enter certain other connections and grasps. We also assumed that the workspace of the robot would be sufficiently uncluttered so that abstract actions like "pick part" will succeed. In this paper, we go beyond these limitations and ground geometrically-meaningful symbolic relations through geometric reasoning that can perform collision and reachability checking, and sampling of good placements.

The contributions of this paper are the following ones:

- a framework for assembly planning that allows reasoning about relations that are grounded on demand in results of geometric reasoning procedures, and the definition of procedures that abstract results of the geometric reasoner into symbols of the knowledge base; and
- extensions of the planner that allow switching between different planning strategies with different goal configurations, and the declaration of action pre-conditions and planning strategies for assembly tasks in cluttered scenes.

2 Related Work

Several projects have pursued ontological modelling in robotics. The IEEE-RAS work group ORA [16] aims to create a standard for knowledge representation in robotics. The ORA core ontology has been extended with ontologies for specific industrial tasks [7], such as kitting: the robot places a set of parts on a tray so these may be carried elsewhere. To the best of our knowledge, assembly tasks have not yet been represented in ORA ontologies. Other robotic ontologies are the Affordance Ontology [23] and the open-source KnowRob ontology [22], the latter of which we use.

Knowledge-enabled approaches have been used for some industrial processes: kitting [3, 4, 14] and assembly (the EU ROSETTA project [8, 12, 18]). Logic descriptions have also been used to define a problem for general purpose planners [4, 9]. In previously cited papers, knowledge modelling for assembly is either in the form of abstract concepts about sequences of tasks (as in [8]), or about geometric features of atomic parts (as in [13]). The approach we use in this paper builds on our previous work [5], where we generate assembly operations from OWL specifications directly (without using PDDL solvers), and the knowledge modelling includes concepts such as affordances, grasps, mechanical connections, and how grasps and mechanical connections influence which affordances are available. Generating assembly operations from OWL specifications is faster than planning approaches and amenable to frequent customization of assembled products. We improve on our previous work by integrating geometric reasoning about the action execution into the knowledge-based planner.

Different types of geometric reasoning have been considered in manipulation planning. [11] has investigated dynamic interactions between rigid bodies. A general manipulation planning approach using several Probabilistic Roadmaps (PRM) has been developed by [17] that considers multiple possible grasps (usable for re-grasping objects) and stable placements of movable objects. The manipulation problem of Navigation Among Movable Obstacles (NAMO) has been addressed by the work in [20] and [19] using a backward search from the goal in order to move objects out of the way between two robot configurations. The work in [1,2] have extended this work with ontological knowledge by integrating task and motion planning.

3 Assembly Activities in Cluttered Workspaces

Assembly tasks often have a fixed recipe that, if followed correctly, would control an agent such that available parts are transformed into an assembled product. These recipes can elegantly be represented using description logics [5]. But inferring the sequence of assembly actions is not sufficient for robots because actions may not be performable in the current situation. This is, for example, the case when the robot cannot reach an object because it is occluded. A notion of space, on the other hand, is very complicated in a logic formalism, but specialized methods exist that efficiently compute qualitative spatial relations such as whether objects are occluding each other.

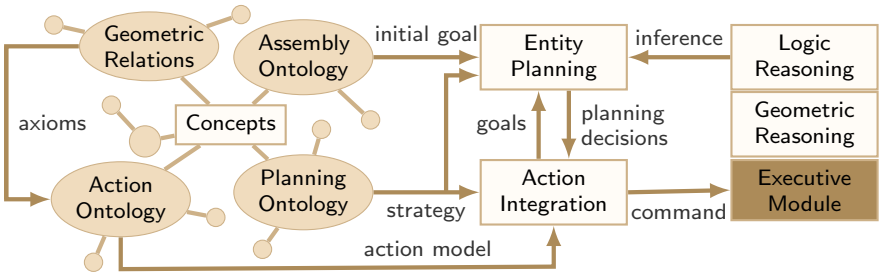

Fig. 2: The architecture of our heterogeneous reasoning system.

Our solution is depicted in Figure 2. We build upon an existing planner and extend it with a notion of action, and geometric reasoning capabilities. Actions are represented in terms of the action ontology which also defines action pre-conditions. Pre-conditions are ensured by running the planner for the action entity. This is used to ensure that the robot can reach an object, or else tries to put away occluding objects. To this end we integrate a geometric reasoner with the knowledge base. The interfaces of the geometric reasoner are hooked into the logic-based reasoning through procedural attachments in the knowledge base.

The planner [5] is an extension of the KnowRob knowledge base [1] [22]. KnowRob is a Prolog-based system with Web Ontology Language (OWL) support. OWL semantics is implemented with the closed world assumption through the use of negation as failure in Prolog rules. We use this to identify what information is missing or false about an individual according to OWL entailment rules. Another useful aspect of KnowRob is that symbols can be grounded through invoking computational procedures such as geometric reasoner.

The geometric reasoner is a module of the Kautham Project [2] [15]. It is a C++ based tool for motion planning that enables to plan under geometric and kinodynamic constraints. It uses the Open Motion Planning Library (OMPL) [21] as a core set of sampling-based planning algorithms. In this work, the RRT-Connect motion planner [10] is used. For the computation of inverse kinematics, the approach developed by [24] is used.

4 Knowledge Representation for Assembly Activities

In our approach, the planner runs within the *perception-action* loop of a robot. The knowledge base maintains a belief state, and entities in it may be referred to in planning tasks. In previous work, we have defined an ontology to describe assemblages, and meta knowledge to control the planner [5]. In the following sections, we will briefly review our previous work in assembly modelling and present further additions to it that we implemented for this paper. The interplay between the different ontologies used in our system is depicted in Figure 3.

[1] http://knowrob.org
[2] https://sir.upc.edu/projects/kautham/

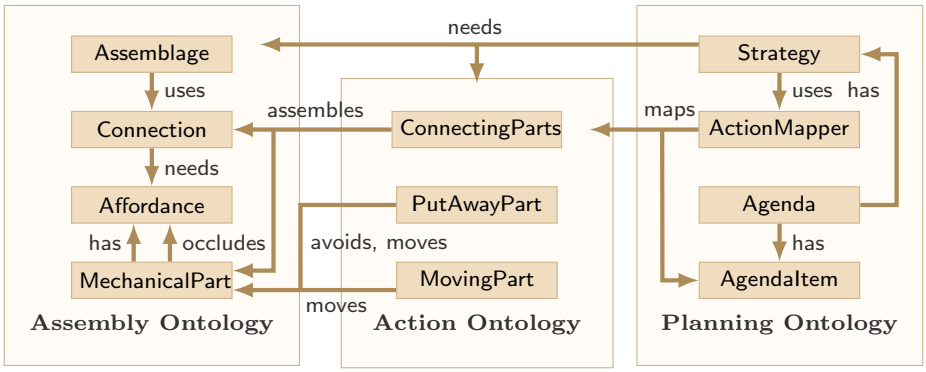

Fig. 3: The interplay of ontologies in our system.

4.1 Assembly Ontology

The upper level of the assembly ontology defines general concepts such as *MechanicalPart* and *AssemblyConnection*. Underneath this level there are part ontologies that describe properties of parts such as what connections they may have, and what ways they can be grasped. Finally, assemblage ontologies describe what parts and connections may form an assemblage. This layered organization allows part ontologies to be reused for different assemblies. Also important is the *Affordance* concept. Mechanical parts provide affordances, which are required (and possibly blocked) by grasps and connections. Apart from these very abstract concepts, some common types of affordances and connections are also defined (e.g. screwing, sliding, and snapping connections).

To these, we have added a new relation *occludesAffordance* with domain *AtomicPart* and range *Affordance*. A triple "P occludesAffordance A" means the atomic part P is placed in such a way that it prevents the robot from moving one of its end effectors to the affordance A (belonging to some other part P'). Parts can be said to be graspable if they have at least one non-occluded grasping affordance. The motivation for the addition of this property is that it helps representing spatial constraints in the workspace, a consideration we did not address in our previous work.

Also, in our previous work, the belief state of the robot was stored entirely in the knowledge base. It includes object poses, if and how an object is grasped, mechanical connections between objects, etc. Consistency is easier to maintain for a centralized belief state, but components of the robot control system need to be tightly integrated with the knowledge base for this to work. In our previous work, we could enforce this as both perception and executive components of our system were developed in our research group. For our work here, however, we need to integrate KnowRob with a motion planner that stores its own representation of the robot workspace, and uses its own naming convention for the objects. We therefore add a data property *planningSceneIndex* to help relate KnowRob object identifiers with Kautham planning scene objects.

4.2 Action Ontology

At some point during the planning process, the robot has to move its body to perform an action. In previous work, we used action data structures which were passed to the plan executive. The plan executive had to take care that pre-conditions were met, which sub-actions to perform, etc. In this work, explicit action representations are used to ensure that pre-conditions are met before performing an action. The action ontology includes relations to describe objects involved, sub-actions, etc. Here, we focus on the representation of pre-conditions.

Our modelling of action pre-conditions is based on the *preActors* relation which is used to assert axioms about entities involved in the action that must hold before performing it. The upper ontology also defines more specific cases of this relation such as *objectActedOn* that denotes objects that are manipulated, or *toolUsed* that denotes tools which are operated by the robot.

ConnectingParts The most essential action the robot has to perform during an assembly task is to connect parts with each other. At least one of the parts must be held by the robot and moved in a way that establishes the connection. Performing the action is not directly possible when the part to be moved cannot be grasped. This is the case when a part blocks a required affordance, for example, due to being in the wrong holder, blocked by another part, etc.

First, we define the relations $assemblesPart \sqsubseteq objectActedOn$, and $fixedPart$ and $mobilePart \sqsubseteq assemblesPart$. These denote *MechanicalPart*'s involved in *ConnectingParts* actions, and distinguish between mobile and static parts. We further define the relation *assemblesConnection* that denotes the *AssemblyConnection* the action tries to establish. The *assemblesPart* relation is defined as property chain $assemblesConnection \circ hasAtomicPart$, where *hasAtomicPart* denotes the parts linked in an *AssemblyConnection*. This ensures that *assemblesPart* only denotes parts that are required by the connection. Using these relations we assert following axioms for the *ConnectingParts* action:

$$\leq 1 assemblesConnection.Thing \wedge \geq 1 assemblesConnection.Thing \quad (1)$$

$$\geq 2 assemblesPart.Thing \quad (2)$$

$$\leq 2 mobilePart.Thing \wedge \geq 1 mobilePart.Thing \quad (3)$$

These axioms define that (1) an action is performed for exactly one assembly connection; (2) at least two parts are involved; and (3) at max two parts are mobile, and at least one mobile part is involved.

Another pre-condition is the graspability of mobile parts. Parts may relate to *GraspingAffordance*'s that describe how the robot should position its gripper, how much force to apply, etc. to grasp the part. We assert the following axioms that ensure each mobile part offers at least one unblocked *GraspingAffordance*:

$$FreeAffordance \equiv (\leq 0 blocksAffordance^{-}.AssemblyConnection) \quad (4)$$

$$\forall mobilePart.(\exists hasAffordance.(GraspingAffordance \wedge FreeAffordance) \quad (5)$$

Next, we define a property *partConnectedTo* that relates a part to parts it is connected to. It is sub-property of the property chain $hasAtomicPart^{-} \circ hasAtomicPart$. Also, we assert that this relation is transitive such that it holds for parts which are indirectly linked with each other. This is used to assert that fixed parts must be attached to some fixture:

$$\forall fixedPart.(\exists partConnectedTo.Fixture) \tag{6}$$

Also, parts must be in the correct fixture for the intended connection. To ensure this, we assert that required affordances must be unblocked:

$$\forall assemblesConnection.(\forall usesAffordance.FreeAffordance) \tag{7}$$

Finally, we define $partOccludedBy \equiv hasAffordance \circ occludesAffordance^{-}$ which relates parts to parts occluding them, and assert that parts cannot be occluded by other parts when the robot intends to put them together:

$$\forall assemblesPart.(\leq 0 partOccludedBy.MechanicalPart) \tag{8}$$

MovingPart and PutAwayPart The above statements assert axioms that must be ensured by the planner. These refer to entities in the world and may require certain actions to be performed to destroy or create relations between them. In this work, we focus on ensuring valid spatial arrangement in the scene.

First, the robot should break non permanent connections in case one of the required affordances is blocked. We define this action as $MovingPart \sqsubseteq PuttingSomethingSomewhere$. The only pre-actor is the part itself. It is linked to the action via the relation $movesPart \sqsubseteq objectActedOn$. We assert that the part must have an unblocked grasping affordance (analogues to axiom (5)).

Further, parts that occlude required parts for an assembly step must be put away. We define this action as $PutAwayPart \sqsubseteq PuttingSomethingSomewhere$. This action needs exactly one *movesPart*, and additionally refers to the parts that should be "avoided", which means that the target position should not lie between the robot and avoided parts: $\exists avoidsPart.MechanicalPart$, where *avoidsPart* is another sub-property of *preActors*. Describing possible target positions in detail would be extremely difficult in a logical formalism, and is not considered in the scope of this work.

4.3 Planning Ontology

Our planner is driven by comparing goals, represented in the TBox, with believes, represented in the ABox, and controlled by meta knowledge that we call planning strategy. The planning strategy determines which parts of the ontology are of interest in the current phase, how steps are ordered, and how they are performed in terms of how the knowledge base is to be manipulated. Possible planning decisions are represented in a data structure that we call planning agenda. Planning agendas are ordered sequences of steps that each, when performed, modify the belief state of the robot in some way. The planner succeeds if the belief state is a proper instantiation of the goal description.

Different tasks require different strategies that focus on different parts of the ontology, and that have specialized rules for processing the agenda. The strategy for planning an assemblage, for example, focuses on relations defined in the assembly ontology. Planning to put away parts, on the other hand, is mainly concerned with spatial relations. In previous work, the strategy selection was done externally. Here, we associate strategies to entities that should be planned with them. To this end, we define the relation *needsEntity* that denotes entities that are planned by some strategy. Strategies assert a universal restriction on this relation in order to define what type of entities can be planned with them. For the assemblage planning strategy, for example, we assert the axiom:

$$\forall needsEntity.(Assemblage \vee AssemblyConnection) \tag{9}$$

Planning decisions may not correspond to actions that the robot needs to perform to establish the decisions in its world. Some decisions are purely virtual, or only one missing piece in a set of missing information required to perform an action. The mapping of planning decisions to action entities is performed in a rule-base fashion in our approach. These rules are described using the *AgendaActionMapper* concept, and are linked to the strategy via the relation *usesActionMapper*. Each *AgendaActionMapper* further describes what types of planning decisions should activate it. This is done with agenda item patterns that activate a mapper in case a pattern matches the selected agenda item. These are linked to the *AgendaActionMapper* via the relation *mapsItem*.

Finally, we define the *AgendaActionPerformer* concept which is linked to the strategy via the relation *usesActionPerformer*. *AgendaActionPerformer* provide facilities to perform actions by mapping them to data structures of the plan executive, and invoking an interface for action execution. They are activated based on whether they match a pattern provided for the last agenda item.

5 Reasoning Process using Knowledge and Geometric Information

Our reasoning system is heterogeneous, which means that different reasoning resources and representations are fused into a coherent picture that covers different aspects. In this section, we will describe the two different reasoning methods used by our system: knowledge-based reasoning and geometric reasoning.

5.1 Knowledge-based Reasoning

In this project, knowledge-based reasoning refers primarily to checking whether an individual obeys the restrictions imposed on the classes to which it is claimed to belong, identifying an individual based on its relations to others, and identifying a set of individuals linked by certain properties (as done when identifying which parts have been linked, directly or indirectly, via connections). This is done by querying an RDF triple store to check whether appropriate triples have been asserted to it or can be inferred.

KnowRob, however, allows more underlying mechanisms for its reasoning. In particular, decision procedures, which can be arbitrary programs, can be linked to properties. In that case, querying whether an object property holds between individuals is not a matter of testing whether triples have been asserted. Rather, the decision procedure is called, and its result indicates whether the property holds or not. Such properties are referred to as computables, and they offer a way to bring together different reasoning mechanisms into a unified framework of knowledge representation and reasoning.

For this work, we use computables to interface to the geometric reasoner provided by the Kautham Project. The reasoner is called to infer whether the relation *occludesAffordance* holds between some part and an affordance.

5.2 Geometric Reasoning

The main role of geometric reasoning is to evaluate geometric conditions of symbolic actions. Two main geometric reasoning processes are provided:

Reachability Reasoning A robot can transit to a pose if it has a valid goal configuration. This is inferred by calling an Inverse Kinematic (IK) module and evaluating whether the IK solution is collision-free. The first found collision-free IK solution is returned, and, if any, the associated pose. Failure may occur if either no IK solution exists or if no collision-free IK solution exists.

Spatial Reasoning We use this module to find a placement for an object within a given region. For the desired object, a pose is sampled that lies in the surface region, and is checked for collisions with other objects, and whether there is enough space to place the object. If the sampled pose is feasible, it is returned. Otherwise, another sample will be tried. If all attempted samples are infeasible, the reasoner reports failure, which can be due to a collision with the objects, or because there is not enough space for the object.

6 OWL Assembly Planning using the Reasoning Process

We extend the planner for computable relations, and also for being able to generate sub-plans in case some pre-conditions of actions the robot needs to perform are not met. We will explain the changes we made for this paper below.

6.1 Selection of Planning Strategies

The planner is driven by finding differences between a designated goal state and the belief state of a robotic agent. The goal is the classification of an entity as a particular assemblage concept. The initial goal state is part of the meta knowledge supplied to the planner (i.e., knowledge that controls the planning process). Strategies further declare meta knowledge about prioritization of actions, and also allow ignoring certain relations entirely during a particular planning phase.

Strategies are useful because it is often hard to formalize a complete planning domain in a coherent way. One way to approach such problems is decomposition: Planning problems may be separated into different phases that have different planning goals, and that have a low degree of interrelations.

Planning in our approach means to transform an entity in the belief state of the robot with local model violations into one that is in accordance with its model. In our approach, each of the planned entities may use its own planning strategy. The strategy for a planning task is selected based on universal restrictions of the *needsEntity* relation. The selection procedure iterates over all known strategies and checks for each whether the planned entity is a consistent value for the *needsEntity* relation. Only the first matching strategy is selected.

Activating a strategy while another is active pauses the former until the sub-plan finished. In case the sub-plan fails, the parent plan also fails if no other way to achieve the sub-plan goal is known. The meta-knowledge controlling the planner ensures to some extent that the planner does not end up in a bad state where it loops between sequences of decisions that revert each other. In case this happens, the planner will detect the loop and fail.

6.2 Integration with Task Executive

Assembly action commands can be generated whenever an assemblage is entirely specified. This is the case if the assemblage is a proper instance of all its asserted types according to OWL entailment rules including the connection it must establish and the sub-assemblies it must link. Further action commands are generated if a part of interest cannot be grasped because another part is occluding it. To this end, we have extended the planning loop such that it uses a notion of actions, and can reason about which action the robot should perform to establish planning decisions in the belief state.

In each step of the planning loop, the agenda item with top priority is selected for processing. Each item has an associated axiom in the knowledge base that is unsatisfied according to what the robot believes. First, the planner infers a possible domain for the decision. That is, for example, which part it should use to specify a connection. This step is followed by the projection step in which the planner manipulates the knowledge base by asserting or retracting facts about entities. Finally, the item is either deleted if completed, or re-added in case the axiom remains unsatisfied. Also, new items are added to the agenda for all the entities that were linked to the planned entity during the projection step.

We extend this process by the notion of *AgendaActionMapper* and *AgendaActionPerformer* which are used for generating action entities and passing them to an action executive respectively. Their implementation in the knowledge base is very similar. They both restrict relations to describe for which entities they should be activated, and may specify agenda item patterns used for their activation. Matching a pattern means in our framework that the processed agenda item is an instance of the pattern according to OWL entailment rules. Finally, both define hooks to procedures that should be invoked to either generate an action description, or to perform it.

The mapping procedure is invoked after the planner inferred the domain for the currently processed agenda item. The generated action entities must not necessarily satisfy all their pre-conditions. Instead, the planner is called recursively while restricting the planning context to *preActor* axioms. This creates a specific *preActor*-agenda that contains only items corresponding to unsatisfied pre-conditions of the action. The items in the *preActor*-agenda may again be associated to actions that need to be performed to establish the pre-conditions in the belief state, and for which individual planning strategies and agendas are used. Finally, the action entity is passed to the selected action performer. In case the action failed, the agenda item is added to the end of the agenda such that the robot tries again later on, and the planner fails in case it detected a loop.

6.3 Planning with Computable Relations

Computable relations are inferred on demand using decision procedures, and as such are not asserted to the triple store. They often depend on other properties, such as the object locations, and require that the robot performs some action that will change its believes, such as putting the object to some other location.

The planner needs to project its decisions into the belief state for non-computable relations. This step is skipped entirely for computable relations: Only the action handler is called to generate actions that influence the computation. In case the robot was not able to change its believes such that the action pre-conditions are fulfilled, the agenda item is put back at the end of agenda.

In addition, we switched to the computable based reasoning interface offered by KnowRob. The difference is that it considers computed and asserted triples.

7 Evaluation

We characterize the performance of our work along following dimensions: Variances of spatial configurations our system can handle, and what types of queries can be answered. The planning domain for evaluation is a toy plane assembly targeted at 4 year old children with 21 parts. The plane is part of the YCB Object and Model Set [6]. It uses slide in connections for the parts, and bolts for fixing the parts afterwards. The robot we use is a YuMi. It is simulated in a kinematics simulator and visualized in RViz.

7.1 Simulation

We test our system with different initial spatial configurations, depicted in Figure 1. The first scene has no occlusions. In the second, the upper part of the plane body is occluding the lower part, and the propeller is occluding the motor grill. Finally, in the third, the chassis is not connected to the holder, and occluded by the upper part of the plane body. We disabled collision checking between the airplane parts to avoid spurious collisions being found at the goal configurations (the connections fit snugly). Geometric reasoning about occlusions allows the

robot knowing when it needs to move parts out of the way and change the initial action sequence provided by the OWL planner.

7.2 Querying

In this work, we have extended the robot's reasoning capabilities regarding to geometric relations it can infer, what pre-conditions an action has, and which actions it has to perform to establish planning decisions in its belief state.

The geometric reasoner is integrated through computable geometric relations. The robot can reason about them by asking questions such as *"what are the occluded parts required in a connection?"*:

```
?- holds( needsAffordance(Connection,Affordance) ),
   holds( hasAffordance(Occluded,Affordance) ),
   holds( partOccludedBy(Occluded,OccludingPart) ).
Occluded='PlaneBottomWing1', OccludingPart='PlaneUpperBody1'.
```

The robot can also reason about what action pre-conditions are not fulfilled, and what it can do to fix this. This is done by creating a planning agenda for the action entity that only considers pre-condition axioms of the action:

```
?- entity(Act, [an,action, [type, 'ConnectingParts'],
      [assemblesConnection, Connection]]),
   agenda_create(Act, Agenda),
   agenda_next_item(Agenda,Item).
Item = "detach_PlaneBottomWing1_partOccludedBy_PlaneUpperBody1"
```

Finally, the robot can reason about what action it should perform that establishes a planning decision in its belief state. It can, for example, ask what action it should perform to dissolve the *partOccludedBy* relation between parts:

```
?- holds( usesActionMapper(Strategy,Mapper) ),
   property_range(Mapper,mapsItem,Pattern),
   individual_of(Item,Pattern),
   call(Mapper,Item,Action).
Action = [an,action, [type, 'PutAwayPart'],
      [movesPart, 'PlaneUpperBody1'], ...].
```

8 Conclusion

In this work, we have described how geometric reasoning procedures may be incorporated into logic-based assembly activity planning to account for spatial constraints in the planning process. The ontology used by the logic-based planner serves as an interface to the information provided by the geometric reasoner. Geometric information is computed through decision procedures which are attached to relation symbols in the ontology. Such relations are referred to in action descriptions to make assertions about what should hold for parts involved in the action before performing it. The planner, driven by finding asserted relations that do not hold in the current situation, can also be used for planning how the situation can be changed such that the preconditions become fulfilled. We have demonstrated that this planning framework enables the robot to handle workspace configurations with occlusions between parts, to reason about them, and to plan sub-activities required to achieve its goals.

References

1. Akbari, A., Muhayyuddin, Rosell, J.: Reasoning-based evaluation of manipulation actions for efficient task planning. In: ROBOT2015: Second Iberian Robotics Conference. Springer (2015)
2. Akbari, A., Muhayyudin, Rosell, J.: Task and motion planning using physics-based reasoning. In: IEEE Int. Conf. on Emerging Technologies and Factory Automation (2015)
3. Balakirsky, S.: Ontology based action planning and verification for agile manufacturing. Robotics and Computer-Integrated Manufacturing **33**(Supplement C), 21 – 28 (2015), special Issue on Knowledge Driven Robotics and Manufacturing
4. Balakirsky, S., Kootbally, Z., Kramer, T., Pietromartire, A., Schlenoff, C., Gupta, S.: Knowledge driven robotics for kitting applications. Robot. Auton. Syst. **61**(11), 1205–1214 (Nov 2013)
5. Beßler, D., Pomarlan, M., Beetz, M.: Owl-enabled assembly planning for robotic agents. In: Proceedings of the 2018 International Conference on Autonomous Agents. AAMAS '18 (2018)
6. Çalli, B., Walsman, A., Singh, A., Srinivasa, S., Abbeel, P., Dollar, A.M.: Benchmarking in manipulation research: The YCB object and model set and benchmarking protocols. CoRR **abs/1502.03143** (2015)
7. Fiorini, S.R., Carbonera, J.L., Gonçalves, P., Jorge, V.A., Rey, V.F., Haidegger, T., Abel, M., Redfield, S.A., Balakirsky, S., Ragavan, V., Li, H., Schlenoff, C., Prestes, E.: Extensions to the core ontology for robotics and automation. Robot. Comput.-Integr. Manuf. **33**(C), 3–11 (Jun 2015)
8. J., M., K., N., H., B.: Describing assembly tasks in declarative way. In: IEEE/ICRA Workshop on Semantics (2013)
9. Kootbally, Z., Schlenoff, C., Lawler, C., Kramer, T., Gupta, S.: Towards robust assembly with knowledge representation for the planning domain definition language (pddl). Robot. Comput.-Integr. Manuf. **33**(C), 42–55 (Jun 2015)
10. Kuffner, J.J., LaValle, S.M.: Rrt-connect: An efficient approach to single-query path planning. In: Robotics and Automation, 2000. Proceedings. ICRA'00. IEEE International Conference on. vol. 2, pp. 995–1001. IEEE (2000)
11. Muhayyudin, Akbari, A., Rosell, J.: Ontological physics-based motion planning for manipulation. In: IEEE Int. Conf. on Emerging Technologies and Factory Automation. IEEE (2015)
12. Patel, R., Hedelind, M., Lozan-Villegas, P.: Enabling robots in small-part assembly lines: The "rosetta approach" - an industrial perspective. In: ROBOTIK. VDE-Verlag (2012)
13. Perzylo, A., Somani, N., Profanter, S., Kessler, I., Rickert, M., Knoll, A.: Intuitive instruction of industrial robots: Semantic process descriptions for small lot production. In: IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). pp. 2293–2300 (2016)
14. Polydoros, A.S., Großmann, B., Rovida, F., Nalpantidis, L., Krüger, V.: Accurate and versatile automation of industrial kitting operations with skiros. In: Towards Autonomous Robotic Systems - 17th Annual Conference (TAROS). pp. 255–268 (2016)
15. Rosell, J., Pérez, A., Aliakbar, A., Muhayyuddin, Palomo, L., García, N.: The Kautham Project: A teaching and research tool for robot motion planning. In: IEEE Int. Conf. on Emerging Technologies and Factory Automation (2014)

16. Schlenoff, C., Prestes, E., Madhavan, R., Goncalves, P., Li, H., Balakirsky, S., Kramer, T., Miguelanez, E.: An IEEE standard ontology for robotics and automation. In: Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on. pp. 1337–1342. IEEE (2012)
17. Siméon, T., Laumond, J.P., Cortés, J., Sahbani, A.: Manipulation planning with probabilistic roadmaps. The International Journal of Robotics Research **23**(7-8), 729–746 (2004)
18. Stenmark, M., Malec, J., Nilsson, K., Robertsson, A.: On distributed knowledge bases for robotized small-batch assembly **12**(2), 519–528 (2015)
19. Stilman, M., Kuffner, J.: Planning among movable obstacles with artificial constraints. The International Journal of Robotics Research **27**(11-12), 1295–1307 (2008)
20. Stilman, M., Schamburek, J.U., Kuffner, J., Asfour, T.: Manipulation planning among movable obstacles. In: Robotics and Automation, 2007 IEEE International Conference on. pp. 3327–3332. IEEE (2007)
21. Sucan, I., Moll, M., Kavraki, L.E., et al.: The open motion planning library. Robotics & Automation Magazine, IEEE **19**(4), 72–82 (2012)
22. Tenorth, M., Beetz, M.: KnowRob – A Knowledge Processing Infrastructure for Cognition-enabled Robots. Int. Journal of Robotics Research **32**(5), 566 – 590 (April 2013)
23. Varadarajan, K.M., Vincze, M.: Afrob: The affordance network ontology for robots. In: Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on. pp. 1343–1350. IEEE (2012)
24. Zaplana, I.; Claret, J., Basañez, L.: Kinematic analysis of redundant robotic manipulators: application to kuka lwr 4+ and abb yumi. Revista Iberoamericana de Automtica e Informtica Industrial. In press. (2017)