Ontology-Based State Representations for Intention Recognition in Human-Robot Collaborative Environments

Craig Schlenoff[a,b,*], Anthony Pietromartire[a], Zeid Kootbally[d], Stephen Balakirsky[a], Sebti Foufou[c,b]

[a] *National Institute of Standards and Technology (NIST), 100 Bureau Drive, Stop 8230, Gaithersburg MD 20899 USA*
[b] *University of Burgundy, LE2i Lab, Dijon, France*
[c] *Computer Science and Engineering Department Qatar University, Doha Qatar*
[d] *University of Maryland, Department of Mechanical Engineering, College Park MD 20742 USA*

Abstract

In this paper, we describe a novel approach for representing state information for the purpose of intention recognition in cooperative human-robot environments. States are represented by a combination of spatial relationships in a Cartesian frame along with cardinal direction information. This approach is applied to a manufacturing kitting operation, where humans and robots are working together to develop kits. Based upon a set of predefined high-level states relationships that must be true for future actions to occur, a robot can use the detailed state information described in this paper to infer the probability of subsequent actions occurring. This would allow the robot to better help the human with the task or, at a minimum, better stay out of his or her way.

Keywords: Intention recognition, state-based representation, ontologies, human robot safety, RCC8

*Corresponding author: 301-975-3456
Email addresses: craig.schlenoff@nist.gov (Craig Schlenoff), pietromartire.anthony@nist.gov (Anthony Pietromartire), zeid.kootbally@nist.gov (Zeid Kootbally), stephen.balakirsky@nist.gov (Stephen Balakirsky), sfoufou@qu.edu.qa (Sebti Foufou)

1. Introduction

Humans and robots working safely and seamlessly together in a cooperative environment is one of the future goals of the robotics community. When humans and robots can work together in the same space, a whole class of tasks becomes amenable to automation, ranging from collaborative assembly, to parts and material handling and delivery. Keeping humans safe requires the ability of the robot to monitor the work area, infer human intention, and be aware of potential dangers soon enough to avoid them. Robots are under development throughout the world that will revolutionize manufacturing by allowing humans and robots to operate in close proximity while performing a variety of tasks [1].

Proposed standards exist for robot-human safety, but these standards focus on robots adjusting their speed based on the separation distance between the human and the robot [2]. In essence, as the robot gets closer to a detected human, the robot gradually decreases its speed to ensure that if a collision between the human and robot occurs, minimal damage will be caused. The approaches focus on where the human is at a given point in time. It does not focus on where they are anticipated to be at points in the future.

A key enabler for human-robot safety in cooperative environments involves the field of intention recognition, in which the robot attempts to understand the intention of an agent (the human) by recognizing some or all of their actions [3] to help predict the human's future actions. Knowing these future actions will allow a robot to plan in such a way as to either help the human perform his/her activities or, at a minimum, not put itself in a position to cause an unsafe situation.

In this paper, we present an approach to representing state information in an ontology for the purpose of ontology-based intention recognition. This is an extension to the conference paper [4] presented at the Ubiquitous Computing Conference, Workshop on Ubiquitous Robotics. In this context, a state is defined as a set of properties of one or more objects in an area of interest that consist of specific recognizable configuration(s) and or characteristic(s). A state is composed of one to many state relationships, which is a specific relation between two objects (e.g., Object 1 is on top of Object 2). This approach to intention recognition is different than many ontology-based intention recognition approaches in the literature (as described in the section 2) as they primarily focus on activity (as opposed to state) recognition and then use a form of abduction to provide explanations for observations. We

infer detailed state relationships using observations based on Region Connection Calculus 8 (RCC8) [5] and other cardinal relationships and then combine these observations to infer the overall state relationships that are true at a given time. The advantages of using state representation over activity representation to perform intention recognition is documented in [6]. In this paper, we describe the state of the art in state vs. activity recognition and show that, although still a very tough problem, the results of state recognition algorithms tend to be much more accurate than those doing activity recognition. Because the intention recognition approaches are only as good as the input that is provided to them, the higher accuracy in state recognition would imply a great accuracy in the intention recognition algorithms that employ them. Once a sequence of state relationships has been identified, we use probabilistic procedures to associate those states with likely overall intentions to determine the next possible action (and associated state) that is likely to occur. This paper focuses on the way that states are represented, reasoned over, and updated in the ontology. The way that the states are combined to represent intentions is only lightly described in this paper. The reader is referred to [6] for more information about this topic. The value that this journal article provides is a general approach for the recognition and representation of states in the environment for the purpose of intention recognition as well as the application of this approach to the manufacturing kitting domain.

We start by providing an overview of intention recognition efforts in the literature as well as various approaches for ontology-based state representation. The approach to state recognition is then presented. Because the ontology is meant to be a more permanent form of knowledge representation and does not need to be updated with every new state detected, we explain the logic employed to determine when the ontology should be updated. We then present a manufacturing kitting scenario along with a corresponding kitting ontology and we apply the state representation approach to this domain. Lastly, we describe the results of an implementation of these state-based approaches in a simulation environment.

2. Intention Recognition and State Representation Related Work

Intention recognition traditionally involves recognizing the intent of an agent by analyzing some of, or all of, the actions that the agent performs. Many of the recognition efforts in the literature are composed of at least

three components: (1) identification and representation of a set of intentions that are relevant to the domain of interest, (2) representation of a set of actions that are expected to be performed in the domain of interest and the association of these actions with the intentions, (3) recognition of a sequence of observed actions executed by the agent and matching them to the actions in the representation [3].

There have been many techniques in the literature applied to intention recognition that follow the three steps listed above, including an ontology-based approach [7], multiple probabilistic frameworks such as Hidden Markov Models [8] and Dynamic Bayesian Networks [9], utility-based intention recognition [10], and graph-based intention recognition [11]. In this paper, we focus on ontology/logic-based approaches.

Once observations of actions have been made, different approaches exist to match those observations to an overall intention or goal. For example, in [12], the authors use existentially quantified observations (not fully grounded observations) to match actions to plan libraries. Mulder can handle situations when they see an action occur (e.g., opening a door), without seeing or knowing who performed that action. Other approaches have focused on building plans with frequency information, to represent how often an activity occurs [13]. The rationale behind these approaches is that there are some activities that occur very frequently and are often not relevant to the recognition process (e.g., a person cleaning his/her hands). These frequently-occurring activities can be mostly ignored, and only activities that are less commonly performed can be considered. In [14], the authors combine probabilities and situation calculus-like formalization of actions. In particular, Demolombe not only define the actions and sequences of actions that constitute an intention, they also state which activities cannot occur for the intention to be valid. For example, if the intention was to drive a car, the activity may be to open the door, get into the car, turn on the engine, release the emergency brake, and take the car out of park. Demolombe may also include that an activity cannot be to turn the car off after it is turned on and before the car is taken out of park.

All of these approaches have focused on the activity being performed as the primary basis for observation and the building block for intention recognition. However, as noted in [3], activity recognition is a very hard problem and one that is far from being solved. There has been limited success in using Radio Frequency Identification (RFID) readers and tags attached to objects of interest to track their movement with the goal of

associating their movement with known activities. For example, in [15], the authors describe the process of making tea as a three step process involving using a kettle, getting a box of tea bags, and adding some combination of milk, sugar or lemon. Each of these activities is identified by a user wearing a special set of gloves that can read RFID tags on objects of interest. However, this additional hardware can be inhibiting and unnatural. Recognizing and representing states as opposed to actions can help to address some of the issues involved in activity recognition (e.g., the location of the tea bag box and the milk carton with respect to the tea cup) which will be the focus of the rest of this paper.

State representation is documented in the literature, although it has not been used (to the authors knowledge) for ontology-based intention recognition. An important aspect of an object's state is its spatial relationships to other objects. In [16], an overview is given that describes the way that spatial information is represented in various upper ontologies including the Descriptive Ontology for Linguistics and Cognitive Engineering (DOLCE), Cyc, the Standard Upper Merged Ontology (SUMO), and Basic Formal Ontology (BFO). The conclusion of this work is the identification of a list of high-level requirements that were necessary for any spatial ontology, including:

- A selection of an appropriate granular partition of the world that picks out the entity that we wish to locate with respect to other entities.
- A selection of an appropriate space region formalization that brings out or makes available relevant spatial relationships.
- A selection of an appropriate partition over the space region (e.g., RCC8, qualitative distance, cardinal direction).
- The identification of the location of the entity with respect to the selected space region description.

Bateman ended up using a variation of the DOLCE ontology, but there is no mention in the literature about the detailed spatial relations that were developed as part of this effort.

Region Connected Calculus (RCC8) [17] is a well-known and cited approach for representing the relationship between two regions in Euclidean

space or in a topological space. There are eight possible relations, including disconnected, externally connected, partially overlapping, etc. However, RCC8 only addresses these relationships in two-dimensional space. There have been other approaches that have tried to extend this into a region connected calculus in three-dimensional space while addressing occlusions [18]. There have also been other approaches to develop calculi for spatial relations. FlipFlop calculus [19] describes the position of one point (the referent) in a plane with respect to two other points (the origin and the relatum). Single Cross Calculus (SCC) [20] is a ternary calculus that describes the direction of a point (C - the referent) with respect to a second point (B - the relatum) as seen from a third point (A - the origin) in a plane. Double Cross Calculus (DCC) [20] extends SCC by allowing one to also determine the relative location of point A with respect to point B (in addition to point B with respect to point A as in SCC). Coarse-grained Dipole Relation Algebra [21] describes the orientation relation between two dipoles (an oriented line segment as determined by a start and end point). Oriented Point Relation Algebra (OPRA) [22] relates two oriented points (a point in a plane with an additional direction parameter) and describes their relative orientation towards each other. All of these approaches, apart from RCC8, focus on points and lines as opposed to regions. Also, despite the large variety of qualitative spatial calculi, the amount of applications employing qualitative spatial reasoning techniques is comparatively small [23].

Throughout the rest of this paper, we will describe an approach for ontology-based state representation within the context of a typical manufacturing scenario.

3. Representing States and Spatial Relations in the Ontology

In this section, we describe an approach that uses RCC8 to model state relationships based on the relative position of objects in the environment. However, we will extend RCC8, which was initially developed for a two-dimensional space, to a three-dimensions space by applying it along all three planes (x-y, x-z, y-z). The frame of reference (see Figure 1) will be with respect to the fixed object (e.g., a worktable), with the z-dimension pointing straight upwards and the tabletop extending in the x- and y- dimension (with detailed orientation specific to the application). Each of the high-level state relationships (to be discusses later in the paper) will have a set of logical rules that associate these RCC8 relations to them. These RCC8 relations should

easily allow a sensor system to characterize the corresponding state relations. To avoid any confusion and to clarify several doubts, It is important to note that in the manufacturing kitting domain, not all objects can be represented as convex regions, as is required by the RCC8 formalism.

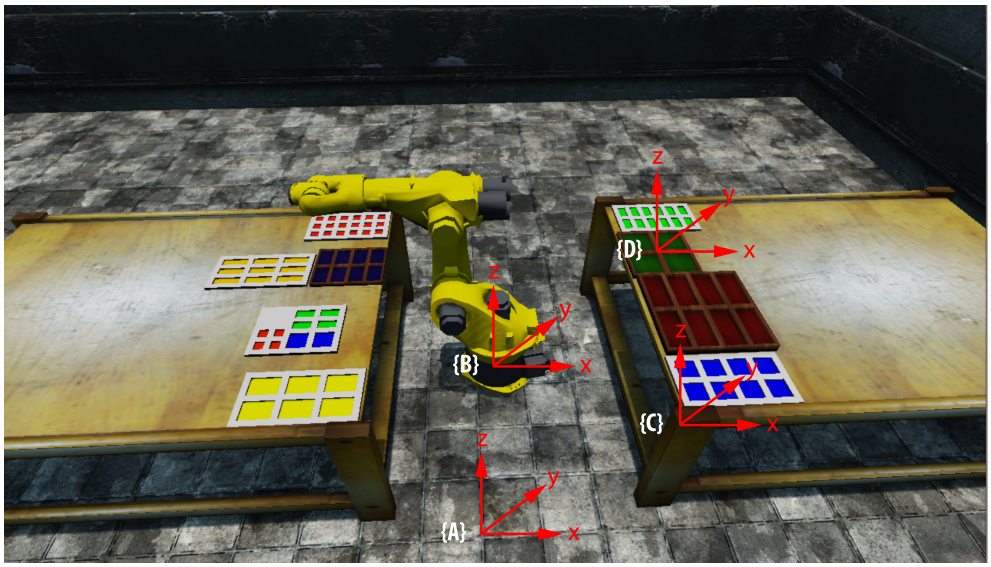


Figure 1: A Sample of frames of reference for a kitting workstation. {A} is a frame of reference of the workstation. {B} is a frame of reference of the robotic arm in relation with {A}. {C} is a frame of reference of the worktable in relation with {A}. {D} is a frame of reference of a parts tray in relation with {C}.

3.1. RCC8 Approach

As mentioned earlier, RCC8 abstractly describes regions in Euclidean or topological space by their relations to each other. RCC8 consists of eight basic relations that are possible between any two regions:

- Disconnected (DC)
- Externally Connected (EC)
- Tangential Proper Part (TPP)
- Non-Tangential Proper Part (NTPP)

- Partially Overlapping (PO)
- Equal (EQ)
- Tangential Proper Part Inverse (TPPi)
- Non-Tangential Proper Part Inverse (NTPPi)

These are shown pictorially in Figure 2.

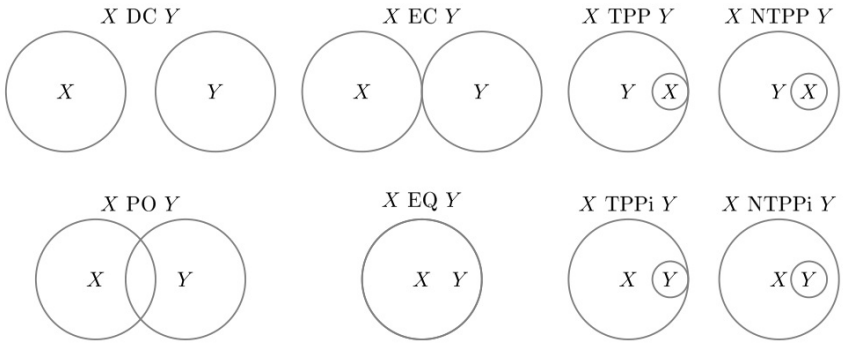


Figure 2: RCC8 Relations (credit: http://en.wikipedia.org/wiki/RCC8).

RCC8 was created to model the relationships between two regions in two dimensions. In many domains, these relations need to be modeled in all three dimensions. As such, every pair of objects has a RCC8 relation in all three dimensions. To address this, we are prepending an x-, y- or z- before each of the RCC8 relations. For example, to represent the RCC8 relations in the x-dimension, the nomenclature would be:

- `x-DC`
- `x-EC`
- `x-TPP`
- `x-NTPP`
- `x-PO`

- `x-EQ`
- `x-TPPi`
- `x-NTPPi`

Similar nomenclature would be used in the y- and z- dimensions. The combination of all 24 RCC relations starts to describe the spatial relations between any two objects in the scene. However, more information is needed to represent the cardinal direction between any two objects. For example, to state that a worktable is empty (worktable-empty(worktable)), one needs to state that there is nothing on top of it. If we assume that the vertical dimension is the z-dimension, then saying that:

$$\texttt{z-EC}(\textit{worktable}, \textit{obj1}) \tag{1}$$

(which intuitively means *obj1* is externally connected to the *worktable* in the z-dimension) is not sufficient because *obj1* could be either on top of or below the *worktable*. In other words, we need to represent directionality. We do this using the following predicates:

$$\texttt{greater-x}(A, B) \tag{2}$$

$$\texttt{smaller-x}(A, B) \tag{3}$$

$$\texttt{greater-y}(A, B) \tag{4}$$

$$\texttt{smaller-y}(A, B) \tag{5}$$

$$\texttt{greater-z}(A, B) \tag{6}$$

$$\texttt{smaller-z}(A, B) \tag{7}$$

which intuitively means, in Equation 2, that the center of gravity of object A is greater than (in the x-dimension in the defined frame of reference) the center of gravity of object B.

3.2. Defining More Complex Relations

There are undoubtedly many other relationships that may be needed in the future to describe a scene of interest. These could include absolute locations and orientations of objects (x, y, z, roll, pitch, yaw) and relative distance (closer, farther, etc.). However, these spatial relations are sufficient for describing the manufacturing kitting example later in this paper.

From these RCC8 spatial relations, we can define more complex spatial relations such as the ones below:

- **Contained-In** - an object is enclosed in a second object from all sides
- **Not-Contained-In** - an object is not enclosed in a second object from all sides
- **Partially-In** - an object is fully inside of a second object in two dimensions and partially in the third dimension
- **In-Contact-With** - touching at least one side and not contained within (i.e., touching outer edges)
- **On-Top-Of** - the z component of the center of gravity of an object is greater than that of a second object and the two objects are overlapping in the x and y dimensions
- **Under** - the z component of the center of gravity of an object is less than that of a second object and the two objects are overlapping in the x and y dimensions

And from these, we can define composite spatial relationships such as:

- **Under-And-In-Contact-With** - an object is both under and in contact with a second object
- **Partially-In-And-In-Contact-With** - an object is fully inside of a second object in two dimensions and partially in the third dimension and is touching in at least one dimension

Below,we formalize these spatial relationships by defining them using the RCC8 state representation. In natural language, Equation 8 below states that object 1 (*obj1*) is contained in object 2 (*obj2*) if *obj1* is tangentially or non-tangentially a proper part of *obj2* in the x, y, and z-dimension. One can logically envision this by drawing two convex figures, and the first convex hull is completely inside of the second convex hull in all three dimensions, with it touching or not touching the second convex hull in all of the three

dimensions.

$$\textbf{Contained-In}(obj1, obj2) \rightarrow \tag{8}$$
$$(\texttt{x-TPP}(obj1, obj2) \vee \texttt{x-NTPP}(obj1, obj2)) \wedge$$
$$(\texttt{y-TPP}(obj1, obj2) \vee \texttt{y-NTPP}(obj1, obj2)) \wedge$$
$$(\texttt{z-TPP}(obj1, obj2) \vee \texttt{z-NTPP}(obj1, obj2))$$

$$\textbf{Not-Contained-In}(obj1, obj2) \rightarrow \tag{9}$$
$$\neg\textbf{Contained-In}(obj1, obj2)$$

$$\textbf{Partially-In}(obj1, obj2) \rightarrow \tag{10}$$
$$\textbf{Not-Contained-In}(obj1, obj2) \wedge$$
$$((\texttt{x-TPP}(obj1, obj2) \vee \texttt{x-NTPP}(obj1, obj2)) \wedge$$
$$(\texttt{y-TPP}(obj1, obj2) \vee \texttt{y-NTPP}(obj1, obj2)) \wedge$$
$$\texttt{z-PO}(obj1, obj2)) \vee$$
$$((\texttt{x-TPP}(obj1, obj2) \vee \texttt{x-NTPP}(obj1, obj2) \wedge$$
$$(\texttt{z-TPP}(obj1, obj2) \vee \texttt{z-NTPP}(obj1, obj2))) \vee$$
$$\texttt{y-PO}(obj1, obj2)) \vee$$
$$((\texttt{y-TPP}(obj1, obj2) \vee \texttt{y-NTPP}(obj1, obj2)) \wedge$$
$$(\texttt{z-TPP}(obj1, obj2) \vee \texttt{z-NTPP}(obj1, obj2)) \wedge$$
$$\texttt{y-PO}(obj1, obj2))$$

$$\textbf{In-Contact-With}(obj1, obj2) \rightarrow \tag{11}$$
$$\texttt{x-EC}(obj1, obj2) \vee \texttt{y-EC}(obj1, obj2) \vee \texttt{z-EC}(obj1, obj2)$$

$$\textbf{On-Top-Of}(obj1, obj2) \rightarrow \tag{12}$$
$$\texttt{greater-z}(obj1, obj2) \wedge ((\texttt{x-EQ}(obj1, obj2) \vee \texttt{x-NTPP}(obj1, obj2) \vee$$
$$\texttt{x-TPP}(obj1, obj2) \vee \texttt{x-PO}(obj1, obj2) \vee \texttt{x-NTPPi}(obj1, obj2) \vee$$
$$\texttt{x-TPPi}(obj1, obj2)) \wedge ((\texttt{y-EQ}(obj1, obj2) \vee \texttt{y-NTPP}(obj1, obj2) \vee$$
$$\texttt{y-TPP}(obj1, obj2) \vee \texttt{y-PO}(obj1, obj2) \vee \texttt{y-NTPPi}(obj1, obj2) \vee$$
$$\texttt{y-TPPi}(obj1, obj2))$$

$$\textbf{Under}(obj1, obj2) \rightarrow \tag{13}$$
$$\texttt{smaller-z}(obj1, obj2) \wedge ((\texttt{x-EQ}(obj1, obj2) \vee \texttt{x-NTPP}(obj1, obj2) \vee$$
$$\texttt{x-TPP}(obj1, obj2) \vee \texttt{x-PO}(obj1, obj2) \vee \texttt{x-NTPPi}(obj1, obj2) \vee$$
$$\texttt{x-TPPi}(obj1, obj2)) \wedge ((\texttt{y-EQ}(obj1, obj2) \vee \texttt{y-NTPP}(obj1, obj2) \vee$$
$$\texttt{y-TPP}(obj1, obj2) \vee \texttt{y-PO}(obj1, obj2) \vee \texttt{y-NTPPi}(obj1, obj2) \vee$$
$$\texttt{y-TPPi}(obj1, obj2))$$

$$\textbf{Under-And-In-Contact-With}(obj1, obj2) \rightarrow \tag{14}$$
$$\textbf{Under}(obj1, obj2) \wedge \textbf{In-Contact-With}(obj1, obj2)$$

$$\textbf{Partially-In-And-In-Contact-With}(obj1, obj2) \rightarrow \tag{15}$$
$$\textbf{Partially-In}(obj1, obj2) \wedge \textbf{In-Contact-With}(obj1, obj2)$$

These spatial relationships will be used later in the paper to define two manufacturing kitting intentions.

3.3. How States are Represented in the Ontology

The spatial relations above are represented as subclasses of the RelativeLocation class which is a subtype of the PhysicalLocation class which is a subtype of DataThing class in the ontology (to be discussed in more detail later in the paper). DataThings are abstract, non-tangible things that are classes in the ontology. There are three types of spatial relations, each described below:

- **RCC8_Relations** - These are the 24 RCC8 relations and the six cardinality direction operators described earlier in this section. These classes are not any further defined but can be instantiated as occurrences of them are found in the environment.

- **Intermediate_State_Relations** - These are intermediate level state relations that can be inferred from the combination of RCC8 and cardinal direction relations. The examples above such as **Under** and **In-Contact-With** are examples of state relations. The logical expression based on the RCC8 and cardinal direction relations (as shown in Equations 8–15) which are evaluated to determine that truth-value of the state relation are represented within the Equivalent Classes. The

information is exported from the ontology during run-time and converted into code that is evaluated as new perception data is presented to the system.

- **Predicates** - These are domain-specific states that are of interest current intention (or set of intentions) being evaluated. For example, in the manufacturing example to be discussed later in the paper, one state of interest is that the worktable is empty. This is true if the worktable is not **Under-And-In-Contact-With** any object. The truth-value of predicates can be determined through the logical combination of state relations. As with state relations, this is captured using the equivalent classes in the ontology.

3.4. Reading and Writing From the Ontology

The purpose of representing state information is to try to determine the intention that is being performed by a human in the environment. It is impractical for the ontology to be updated every time a new state relation or predicate is identified as being true because of the overhead involved in updating the ontology (which is often represented as flat files) and the frequency in which state relations and predicates can change in a highly dynamic environment. This information is maintained in the code that is performing the intention recognition and is only written to the ontology when certain conditions are met. This section describes those conditions.

First, it is important to describe what the state information will be used for. Once intentions in the environment are recognized, it is envisioned that a robot will determine what actions it should take to assist the human in performing those actions. This could be anything from staying away from the human's projected next actions (that are consistent with the intention) to proactively taking steps to help the human perform their intention. In the latter case, the robot would not only need to know the intention of the human; it would also need to know the state of the environment to determine what the best action would be for it to perform. For example, if it was determined that the human's intention was to build an industrial kit that required two more Part A's, and there was only one more Part A left in the part bin, the robot may go to retrieve additional Part A's to help the human accomplish his/her intention.

Only the objects and state relations that are considered "of interest" are updated in the ontology. Every intention has a set of objects that participate

in it. For example, Parts A, B, C, D, a kit tray, a robot gripper, a set of parts bin and a table may be the objects that participate in a kitting operation. Each of the objects is part of state relations and predicates that are important and relevant to the intention(s) being evaluated. For example, in the kitting intention, the intention system may be specifically looking for cases in which Part A is either in the parts bin or in the kit tray. Therefore, Part A is an object of interest. When updating the ontology, as described below, all spatial relations in which Part A is involved, either those that are expected or those that are not (e.g., Part A is on the floor) are considered state relations of interest. Though there could be many objects and relations of interest, only a very few of them will change during any individual state, thus the system performance will not be negatively affected even with a large number of objects.

State information is updated in the ontology when one of three conditions occur:

- **A new intention becomes the "most likely" intention** - Although the details of the intention recognition system is outside the scope of the paper (more information can be found in [6]), it is important to understand that the system's output consists of a list of possible intentions and a probability associated with each. As new perception data becomes available and new state relations and predicates evaluate to true, the intention recognition algorithms are re-run and new probabilities are determined. As a new intention becomes most likely, the states relations that are associated with that intention are updated in the ontology so that robot can have the latest snapshot of the state of the world to make informed decisions of the actions it should take to help with the intention. It is anticipated that early in the process, the most likely intention will change often, but as the intention proceeds, the results will stabilize and the state information will not need to be updated as often.

- **An intention completed** - When an intention is complete, the state information is updated in the ontology to reflect the final state of the intention. This information can also be used as the initial state to recognize subsequent intentions.

- **A failure is determined** - There are cases when a failure occurs during the intention recognition process. This occurs when an object

of interest occupies a state that was not anticipated in any intention described in the ontology. For example, a part may drop out of a person's hand and fall on the floor. In this case, the part would have a relation of On-Top-Of with the floor, which is not a state relation which was defined in the ontology. Similarly, a part could be put on the table temporarily while another activity is being performed by the human. This would be an unexpected state that was not pre-determined. In these cases, not only would the probability of an intention be decreased, but the current state relations and predicates would be updated in the ontology to allow for replanning or to allow the robotic system to explore options to rectify the situations to further the most likely intention.

4. The Manufacturing Kitting Domain and Ontology

Although we expect the approaches described in this paper to be generic, we are initially applying them to a specific manufacturing domain to show their feasibility. In this domain, we focus on manufacturing kitting operations as described in [24].

4.1. Manufacturing Kitting Domain Description

Figure 3: Example kit (courtesy of http://littlemachineshop.com/).

Kitting is the process in which several different, but related items are placed into a container and supplied together as a single unit (kit) as shown in Figure 3. Kitting is often performed prior to final assembly in industrial

assembly of manufactured products so all of the necessary parts are gathered in one location. Manufacturers utilize kitting due to its ability to provide cost savings including saving manufacturing or assembly space, reducing assembly workers' walking and searching times, and increasing line flexibility and balance.

In batch kitting, the kit's component parts may be staged in containers positioned in the workstation or may arrive on a conveyor. Component parts may be fixtured, for example, placed in compartments on trays, or may be in random orientations, for example placed in a large bin. In addition to the kit's component parts, the workstation usually contains a storage area for empty kit boxes as well as completed kits.

Kitting has not yet been automated in many industries where automation may be feasible. Consequently, the cost of building kits is higher than it could be [24]. We are addressing this problem by building models of the knowledge that will be required to operate an automated kitting workstation in an agile manufacturing environment. For our automated kitting workstation, we assume that a robot performs a series of pick-and-place operations in order to construct the kit. These operations include:

1. Pick up empty kit and place on work table.
2. Pick up multiple component parts and place in kit.
3. Pick up completed kit and place in full kit storage area.

Each of these actions may be a compound action that includes other actions such as end-of-arm tool changes, path planning, and obstacle avoidance. Finished kits are moved to the assembly floor where components are picked from the kit for use in the assembly procedure. The kits are normally designed to facilitate component picking in the correct sequence for assembly. Component orientation may be constrained by the kit design in order to ease the pick-to-assembly process. Empty kits are returned to the kit building area for reuse.

4.2. Manufacturing Kitting Ontology Description

An industrial kitting ontology has been developed which will serve as the basis for the Industrial Robotics Ontology as part of the IEEE Robotics and Automation Society's (RAS) Ontologies for Robotics and Automation (ORA) Standard Working Group[1].The kitting workstation model was defined

[1]http://lissi.fr/ora/doku.php

Table 1: Kitting object ontology overview.

SolidObject *PrimaryLocation SecondaryLocation*
BoxyObject *Length Width Height*
WorkTable
EndEffector *Description Weight Id LoadWeight*
GripperEffector
VacuumEffector *CupDiameter Length*
VacuumEffectorMultiCup *ArrayNumber ArrayRadius*
VacuumEffectorSingleCup
EndEffectorChangingStation *EndEffectorHolders*
EndEffectorHolder *EndEffector*
Kit *Tray DesignRef Parts Finished?*
KittingWorkstation *WorkTable Robot ChangingStation AngleUnit LengthUnit WeightUnit KitDesigns OtherObstacles Skus*
KitTray *SkuRef Serialnumber*
LargeBoxWithEmptyKitTrays *LargeContainer Trays*
LargeBoxWithKits *LargeContainer Kits KitDesignRef Capacity*
LargeContainer *SkuRef SerialNumber*
Part *SkuRef SerialNumber*
PartsBin *PartQuantity PartSkuRef SkuRef SerialNumber*
PartsTray *SkuRef SerialNumber*
PartsTrayWithParts *PartTray*
Robot *Description MaximumLoadWeight EndEffector WorkVolume*

DataThing
BoxVolume *MaximumPoint MinimumPoint*
KitDesign *KitTraySkuRef PartRefAndPoses*
PartRefAndPose *SkuRef Point XAxis ZAxis*
PhysicalLocation *RefObject*
PoseLocation *Point XAxis ZAxis*
PoseLocationIn
PoseLocationOn
PoseOnlyLocation
RelativeLocation *Description*
RelativeLocationIn
RelativeLocationOn
Point *X Y Z*
ShapeDesign *Description*
BoxyShape *Length Width Height HasTop*
StockKeepingUnit *Description Shape Weight EndEffectorRefs*
Vector *I J K*

in OWL's functional-style syntax.

The model has two top-level classes, SolidObject and DataThing, from which all other classes are derived. SolidObject models solid objects, things made of matter. DataThing models data. Subclasses of SolidObject and DataThing are defined as shown in Table 1. The level of indentation indicates subclassing. For example, WorkTable is derived from BoxyObject, and BoxyObject is derived from SolidObject. Items in italics following classes are names of class attributes. Derived types inherit the attributes of the parent. Each attribute has a specific type not shown in the listing below. If an attribute type has derived types, any of the derived types may be used.

Each SolidObject has a native coordinate system conceptually fixed to the object. The native coordinate system of a BoxyObject, for example, has its origin at the middle of the bottom of the object, its Z axis perpendicular to the bottom, and the X axis parallel to the longer horizontal edges of the object. In addition to objects, the ontology also represents activ-

Table 2: Preconditions and effects for the action *take-kittray*

preconditions	*effects*
robot-empty(*robot*)	¬robot-empty(*robot*)
lbwekt-not-empty(*lbwekt*)	kittray-loc-robot(*kittray*,*robot*)
robot-with-endeffector(*robot*,*eeff*)	robot-holds-kittray(*robot*,*kittray*)
kittray-loc-lbwekt(*kittray*,*lbwekt*)	¬kittray-loc-lbwekt(*kittray*,*lbwekt*)
endeffector-loc-robot(*eeff*,*robot*)	
worktable-empty(*worktable*)	
endeffector-type-kittray(*eeff*,*kittray*)	

ities. In the manufacturing kitting ontology, both the activities and the pre- and post-conditions of those activities need to be represented. Preconditions and post-conditions (effects) are a combination of predicates. An example of action is *take-kittray* (take kit tray) which is defined as *take-kittray*(*robot*,*kittray*,*lbwekt*,*eeff*,*worktable*). In natural language, the *take-kittray* action involves a robot (*robot*) equipped with an end effector (*eeff*) picking up a kit tray (*kittray*) from within a large box with empty kit trays (*lbwekt*). This action is formally defined in the State Variable Representation [25]. Table 2 shows the preconditions and effects (predicates) that are associated with this action.

Each of the predicates in Table 2 is described below:

1. robot-empty(*robot*) - TRUE iff robot (*robot*) is not holding anything
2. lbwekt-not-empty(*lbwekt*) - TRUE iff the large box with empty kit trays (*lbwekt*) is not empty
3. robot-with-endeffector(*robot*,*eeff*) - TRUE iff robot (*robot*) is equipped with the end effector (*eeff*)
4. kittray-loc-lbwekt(*kittray*,*lbwekt*) - TRUE iff the kit tray (*kittray*) is in the large box with empty kit trays (*lbwekt*)
5. endeffector-loc-robot(*eeff*,*robot*) - TRUE iff the end effector (*eeff*) is being held by the robot (*robot*)
6. worktable-empty(*worktable*) - TRUE iff there is nothing on the work table (*worktable*)
7. endeffector-type-kittray(*eeff*,*kittray*) - TRUE iff the end effector (*eeff*) is designed to handle the kit tray (*kittray*)
8. ¬robot-empty(*robot*) - TRUE iff the robot (*robot*) is holding something
9. kittray-loc-robot(*kittray*,*robot*) - TRUE iff the kit tray (*kittray*) is being held by the Robot (*robot*)
10. robot-holds-kittray(*robot*,*kittray*) - TRUE iff the Robot (*robot*) is holding the kit tray (*kittray*)
11. ¬kittray-loc-lbwekt(*kittray*,*lbwekt*) - TRUE iff the kit tray (*kittray*) is not in the large box with empty kit trays (*lbwekt*)

There are many other actions that can be performed during the kitting operation, including putting down a kit tray, picking up and putting down a part, attaching/removing an end effector, etc. Each of these actions has associated preconditions and effects. Some of the predicates described above appear to be redundant, such as number 3 and 5 above. Some are exact opposites, such as numbers 1 and 8 above. These are included in this way because of the requirements of a sister project that looking at automated planning and the goal was to leverage the same predicates between the two projects.

5. Representing Manufacturing States

When modeling the predicates in the preconditions and effects shown in the previous section, the first step is to precisely define the predicates in such a way as to determine if there were similar intermediate spatial relations that

could be leveraged. We can start to formalize the previous definition of the predicates as depicted in the Revised Definition column in Table 3.

Table 3: Revised definitions of spatial relationships

Predicate	**Previous definition**	**Revised definition**
lbwekt-not-empty(*lbwekt*)	TRUE iff the large box with empty kit trays (*lbwekt*) is not empty	There is an object that is **Contained-In** the large box with empty kit trays (*lbwekt*)
robot-with-endeffector(*robot*,*eeff*)	TRUE iff robot (*robot*) is equipped with the end effector (*eeff*)	The end effector (*eeff*) is **In-Contact-With** the robot (*robot*)[a] [a]For the purpose of this effort, we will assume that if the end effector is in contact with the robot arm, then it is correctly positioned and attached
kittray-loc-lbwekt(*kittray*,*lbwekt*)	TRUE iff the kit tray (*kittray*) is in the large box with empty kit trays (*lbwekt*)	The kit tray (*kittray*) is **Contained-In** the large box with empty kit trays (*lbwekt*)
endeffector-loc-robot(*eeff*,*robot*)	TRUE iff the end effector (*eeff*) is being held by the robot (*robot*)	The end effector (*eeff*) is **In-Contact-With** the robot (*robot*)
worktable-empty(*worktable*)	TRUE iff there is nothing on the work table (*worktable*)	There is no object that is **On-Top-Of** and **In-Contact-With** the work table (*worktable*)
endeffector-type-kittray(*eeff*,*kittray*)	TRUE iff the end effector (*eeff*) is designed to handle the kit tray (*kittray*)	The end effector (*eeff*) can handle the kit tray (*kittray*)
kittray-loc-robot(*kittray*,*robot*)	TRUE iff the kit tray (*kittray*) is being held by the Robot (*robot*)	The kit tray (*kittray*) is **In-Contact-With** the robot (*robot*) and there is nothing **Under-And-In-Contact-With** the kit tray (*kittray*)
¬kittray-loc-lbwekt(*kittray*,*lbwekt*)	TRUE iff the kit tray (*kittray*) is not in the large box with empty kit trays (*lbwekt*)	The kit tray (*kittray*) is **Not-Contained-In** the large box with empty kit trays (*lbwekt*)

The predicates robot-empty, ¬robot-empty, and robot-holds-kittray depend on the type of end effector that is being used to define the predicate. These predicates are not included in Table 3 but instead discussed and elaborated below. We will assume there are two types of end effectors: a vacuum end effector and a parallel gripper end effector. The vacuum end effector picks objects up by positioning itself on top of the object and uses air to create a vacuum to adhere to the object. The parallel gripper end effector picks objects up by squeezing them from both sides. Because the vacuum end effector would not reasonably be used to pick up the kit tray, the vacuum-holds-kittray(*robot*,*kittray*) state is not included below. In the case of the vacuum end effector, the relevant predicates would be:

- vacuum-robot-empty(*robot*) - there is no object **Under-And-In-Contact-With** the robot (*robot*) vacuum effector
- ¬vacuum-robot-empty(*robot*) - there is an object **Under-And-In-Contact-With** the robot (*robot*) vacuum effector

In the case of the parallel gripper end effector, the predicates would be:

- gripper-robot-empty(*robot*) - there is no object **Partially-In-And-In-Contact-With** the robot (*robot*) gripper
- ¬gripper-robot-empty(*robot*) - there is an object **Partially-In-And-In-Contact-With** the robot (*robot*) gripper
- gripper-holds-kittray(*robot*,*kittray*) - the kit tray (*kittray*) is **Partially-In-And-In-Contact-With** the robot (*robot*) gripper [2]

As mentioned earlier, some of the predicates described above are exact opposites. These are included in this way because of the requirements of a sister project that looking at automated planning and the goal was to leverage the same predicates between the two projects. There is also one predicate that does not rely on spatial relations. The definition of endeffector-type-kittray(*eeff*,*kittray*) states that a specific end effector must be able to be used on a kit tray. This information is included in the ontology class to describe the kit tray and therefore is out of scope of this document.

[2]For this work, we assume that if the kit tray in partially in and in contact with the gripper, it is being held by the gripper.

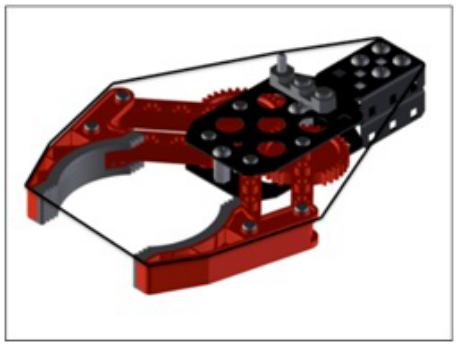


Figure 4: Convex hull around robot gripper.

As stated earlier, in the manufacturing kitting domain, not all objects can be represented as convex regions, as is required by the RCC8 formalism. For example, the robot gripper in Figure 4 is not convex and thus does not neatly fit into the RCC8 approach. To address this, we develop a convex hull along each relevant plane (as shown in Figure 4) around objects of this sort and use that convex hull to represent the region of the object in that plane.

Based on the manufacturing kitting ontology and the spatial relations, we can formally define the 11 manufacturing kitting predicates:

$$\begin{aligned}&\mathsf{lbwekt\text{-}not\text{-}empty}(\mathit{lbwekt}) \rightarrow \\ &\mathsf{SolidObject}(\mathit{obj1}) \wedge \mathbf{Contained\text{-}In}(\mathit{obj1}, \mathit{lbwekt})\end{aligned} \tag{16}$$

$$\begin{aligned}&\mathsf{robot\text{-}with\text{-}endeffector}(\mathit{robot}, \mathit{eeff}) \rightarrow \\ &\mathbf{In\text{-}Contact\text{-}With}(\mathit{robot}, \mathit{eeff})\end{aligned} \tag{17}$$

$$\begin{aligned}&\mathsf{kittray\text{-}loc\text{-}lbwekt}(\mathit{kittray}, \mathit{lbwekt}) \rightarrow \\ &\mathbf{Contained\text{-}In}(\mathit{kittray}, \mathit{lbwekt})\end{aligned} \tag{18}$$

$$\begin{aligned}&\mathsf{worktable\text{-}empty}(\mathit{worktable}) \rightarrow \\ &\mathsf{SolidObject}(\mathit{obj1}) \wedge \neg\mathbf{On\text{-}Top\text{-}Of}(\mathit{obj1}, \mathit{worktable}) \wedge \\ &\neg\mathbf{In\text{-}Contact\text{-}With}(\mathit{obj1}, \mathit{worktable})\end{aligned} \tag{19}$$

$$\text{gripper-holds-kittray}(robot, kittray) \rightarrow \tag{20}$$
$$\text{GripperEffector}(eeff) \wedge \text{robot-with-endeffector}(robot, eeff) \wedge$$
$$\textbf{Partially-In-And-In-Contact-With}(kittray, eeff)$$

$$\neg\text{kittray-loc-lbwekt}(kittray, lbwekt) \rightarrow \tag{21}$$
$$\neg\textbf{Contained-In}(kittray, lbwekt)$$

$$\text{vacuum-robot-empty}(robot) \rightarrow \tag{22}$$
$$\text{SolidObject}(obj1) \wedge \text{SolidObject}(obj2) \wedge \text{VacuumEffector}(eeff) \wedge$$
$$\text{robot-with-endeffector}(robot, eeff) \wedge$$
$$\neg(\textbf{Under-And-In-Contact-With}(obj1, eeff) \wedge$$
$$\neg(\textbf{Under-And-In-Contact-With}(obj2, obj1))$$

$$\neg\text{vacuum-robot-empty}(robot) \rightarrow \tag{23}$$
$$\text{SolidObject}(obj1) \wedge \text{SolidObject}(obj2) \wedge \text{VacuumEffector}(eeff) \wedge$$
$$\text{robot-with-endeffector}(robot, eeff) \wedge$$
$$\textbf{Under-And-In-Contact-With}(obj1, eeff) \wedge$$
$$\neg\textbf{Under-And-With-Contact-With}(obj2, obj1)$$

$$\text{gripper-robot-empty}(robot) \rightarrow \tag{24}$$
$$\text{SolidObject}(obj1) \wedge \text{GripperEffector}(eeff) \wedge$$
$$\text{robot-with-endeffector}(robot, eeff) \wedge$$
$$\neg\textbf{Partially-In-And-In-Contact-With}(obj1, eeff)$$

$$\neg\text{gripper-robot-empty}(robot) \rightarrow \tag{25}$$
$$\text{SolidObject}(obj1) \wedge \text{GripperEffector}(eeff) \wedge$$
$$\text{robot-with-endeffector}(robot, eeff) \wedge$$
$$\textbf{Partially-In-And-In-Contact-With}(obj1, eeff)$$

$$\text{kittray-loc-robot}(kittray, robot) \rightarrow \tag{26}$$
$$\text{gripper-holds-kittray}(robot, kittray)$$

The formal definitions of these predicates will allow their existence to be recognized in a manufacturing environment, which in turn can be used by a state-based intention recognition system. The presence of predicates in certain predefined orders can help a robot recognize the intention of a human in the environment, which would allow the robot to better assist the human in performing upcoming activities.

6. Experimentation and Results

This section describes the experiments that were performed to validate the state recognition and representation approaches described in this paper. Though these states will be used as input to an intention recognition system, this part of the experiment only focused on the state recognition and representations aspects.

6.1. Implementation Architecture and Description

Figure 5 shows the implementation architecture that was applied to the experiment. The evaluation was performed using the Unified System for Automation and Robot Simulation (`USARSim`) simulator [26]. `USARSim` is a high-fidelity simulation of robots and environments based on the Unreal Tournament game engine. It is intended as a research tool and is the basis for the RoboCup rescue virtual robot competition [27]. In this experiment, we used `USARSim` to simulate kitting operations. The `USARTruth` tool extracts ground truth data (the coordinates, the rotation, and the dimensions) of all objects in the simulated environment.

The ontology described earlier in this paper and the output from `USARTruth` are used as inputs to the `state recognition algorithms`. The `state recognition algorithms` parse the intermediate state relations and predicates in the ontology and extracts their equivalent classes. This is used to determine the logical formulas to analyze and link to the locations, rotations, and dimensions data retrieved from `USARTruth`. The algorithms first compute the truth-value of the RCC8 relationships in the ontology. The intermediate state relationships are then evaluated based on the truth-value of the RCC8 relationships, and finally the predicates are evaluated based on the truth-value of the intermediate state relationships. This process is run twice per second as new input is received from `USARTruth`. This frequency can be modified as needed.

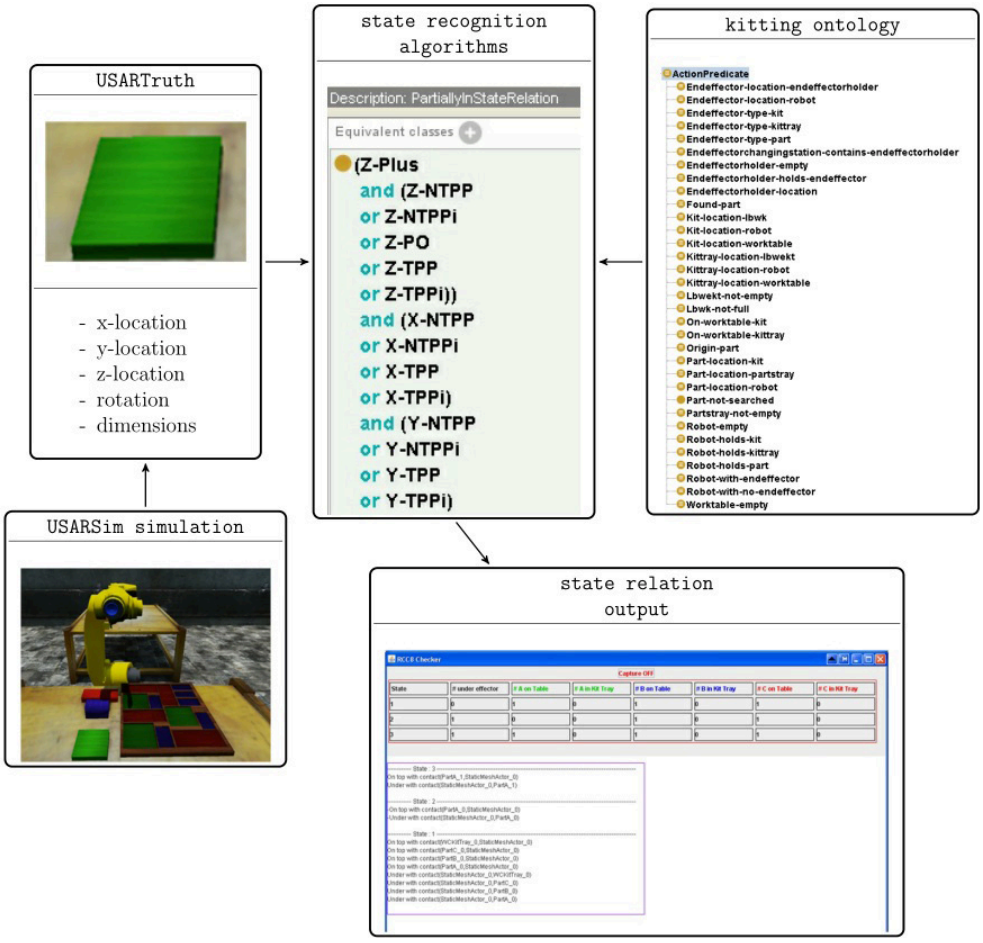


Figure 5: Implementation architecture.

A set of kitting scenarios was developed in `USARSim` to test the algorithms. All were industrial kitting scenarios, which involved different kit configurations, different parts, and different number of parts. One such scenario is shown in Figure 6. In this scenario, we use a robotic arm to represent a human arm performing actions. The kit that is being built contains three part A's (green), six part B's (blue) and four part C's (red). This is shown by the colors on the kit tray. As parts are picked up from the table, a new part of the same type is spawn. When the kit is finished, i.e., when the kit tray contains all the parts, the robot arm returns to its initial position.

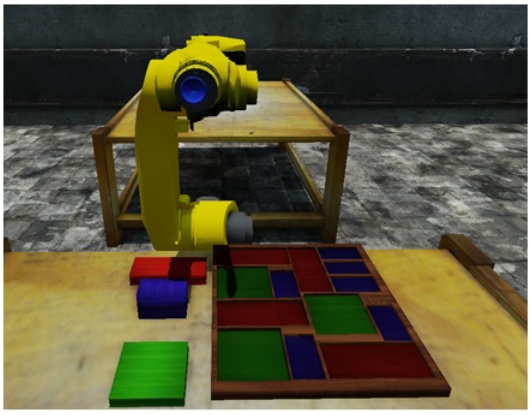


Figure 6: Scenario.

6.2. Output and Results

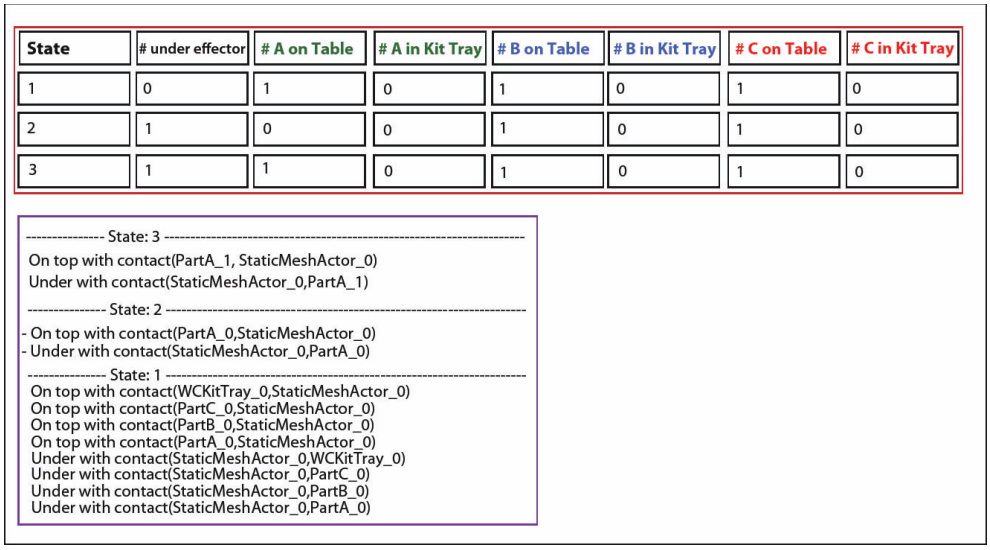
Interestingly, in this industrial kitting scenario, there are relatively few state relationships that are relevant and of interest. They include:

- Part A, B, or C is **On-Top-Of** and **In-Contact-With** the table
- Part A, B, or C is **On-Top-Of** and **In-Contact-With** the kit tray
- Part A, B, C is **Under-And-In-Contact-With** the end effector

For the third bullet above, the spatial relation is **Under-And-In-Contact-With** instead of **Partially-In-And-In-Contact-With** because a vacuum effector is being used, which attaches to the part from the top and "sucks it in". As such, the two are connected when the part is "under and in contact with" the end effector.

Figure 7 shows the output of the `state recognition algorithms` and consists of:

- A table (top part in Figure 7) showing the number of parts held by the robot, the number and types of parts on the Table, and the number and types of Parts in the Kit Tray. Each of these is represented as columns in the table. Each row of the table represents a state.
- A textual description (bottom part in Figure 7) of all the predicates that are true in each state. State 1 displays all predicates that are true



State	# under effector	# A on Table	# A in Kit Tray	# B on Table	# B in Kit Tray	# C on Table	# C in Kit Tray
1	0	1	0	1	0	1	0
2	1	0	0	1	0	1	0
3	1	1	0	1	0	1	0

Figure 7: State recognition algorithms output.

in that state. States 2,...,n show the predicates that have changed from the previous state. If a "-" sign precedes the predicate name, the predicate is no longer true in the current state. The absence of a "-" sign before the predicate name means that this predicate was not in the previous state and is now true in the current state.

In Figure 7, Parts A, B, and C are on the table and there are no parts in the kit tray for State 1. This is characterized by the three `On top with contact` relations (read as "**On-Top-Of**" and "**In-Contact-With**") between the parts and the table. A new state is established when a predicate-of-interest's truth-value changes. In State 2, the effector picked up Part A so Part A is under the effector and no longer on the table. In State 3, a new Part A appears on the table since the previous Part A has been picked up by the robot. This process will continue and new states will be defined until the kit assembly process is complete.

7. Conclusion

In this paper, we described a novel approach for representing state information for the purpose of intention recognition in cooperative human-robot

environments. States were represented by a combination of spatial relationships in a Cartesian frame along with cardinal direction information. This approach was applied to a manufacturing kitting operation, where humans and robots are working together to develop kits.

The next step in this work is to apply the output of the state recognition algorithms to the ultimate goal of intention recognition. Each row in Figure represents the predicates that are true at a given point of time (a state). The time ordering of these perceived predicates can be compared to predefined ordering of predicates that comprise an known intention. Future work will involve developing probabilistic algorithms to perform this comparison, thus allowing a robotic observer to infer what intention is most likely occurring by a human in the environment and then take actions to help advance that intention.

[1] S. Szabo, R. Norcross, W. Shackleford, Safety of Human-Robot Collaboration Systems Project, http://www.nist.gov/el/isd/ps/safhumrobcollsys.cfm, 2011.

[2] J. Chabrol, Robotics 3 (1987) 229–233.

[3] F. Sadri, Handbook of Research on Ambient Intelligence and Smart Environment: Trends and Perspectives, pp. 346–375.

[4] C. Schlenoff, A. Pietromartire, Z. Kootbally, S. Balakirsky, S. Foufou, in: Proceedings of the 2012 ACM Conference on Ubiquitous Computing, UbiComp '12, Pittsburg, PA, USA, pp. 810–817.

[5] D. A. Randell, Z. Cui, A. G. Cohn, in: Proceedings of the 3rd International Conference on Knowledge Representation and Reasoning, KRR '92, San Mateo, CA, USA, pp. 165–176.

[6] C. Schlenoff, A. Pietromartire, Z. Kootbally, S. Balakirsky, T. Kramer, S. Foufou, Engineering Creative Design in Robotics and Mechatronics.

[7] H. Jeon, T. Kim, J. Choi, in: Proceedings of the 2008 International Conference on Multimedia and Ubiquitous Engineering, MUE '08, Busan, Korea, pp. 244–248.

[8] R. Kelley, A. Tavakkoli, C. King, M. Nicolescu, M. Nicolescu, G. Bebi, in: Proceedings of the 3rd ACM/IEEE International Conference on Human Hobot Interaction, HRI '08, Amsterdam, The Netherlands, pp. 367–374.

[9] O. C. Schrempf, U. D. Hanebeck, in: International Conference on Informatics in Control, Automation and Robotics, ICAR'05, Barcelona, Spain, pp. 251–256.

[10] W. Mao, J. Gratch, in: Workshop on Agent Tracking: Modeling Other Agents from Observations, AAMAS 2004, New York, NY, USA.

[11] S.-J. Youn, K.-W. Oh, International Journal of Applied Science, Engineering & Technology 4 (2008).

[12] F. Mulder, F. Voorbraak, Information Fusion 4 (2003) 47–61.

[13] P. A. Jarvis, T. F. Lunt, K. L. Myers, AI Magazine 6 (2005) 73–81.

[14] R. Demolombe, A. M. O. Fernandez, in: Proceedings of the 6th international conference on Computational Logic in Multi-Agent Systems, CLIMA V, Springer-Verlag, London, UK, 2005, pp. 358–372.

[15] M. Philipose, K. P. Fishkin, M. Perkowitz, D. J. Patterson, D. Fox, H. Kautz, D. Hahnel, IEEE Pervasive Computing 3 (2004) 50–57.

[16] J. Bateman, S. Farrar, Spatial Ontology Baseline Version 2.0, University of Bremen, 2006.

[17] F. Wolter, M. Zakharyaschev, in: Proceedings of the 7th Conference on Principles of Knowledge Representation and Reasoning, KR2000, Morgan Kaufmann, 2000, pp. 3–14.

[18] J. Albath, J. Leopold, C. Sabharwal, A. Maglia, in: Proceedings of the 23rd International Conference on Computer Applications in Industry and Engineering, CAINE 2010, Las Vegas, NV, USA.

[19] G. F. Ligozat, in: A. U. Frank, I. Campari (Eds.), Spatial Information Theory A Theoretical Basis for GIS, volume 716, Springer Berlin Heidelberg, 1993, pp. 54–68.

[20] C. Freksa, in: A. U. Frank, I. Campari, U. Formentini (Eds.), Theories and Methods of SpatioTemporal Reasoning in Geographic Space, volume 639 of *Lecture Notes in Computer Science*, Berlin: Springer-Verlag, 1992, pp. 162–178.

[21] C. Schlieder, in: A. U. Frank, W. Kuhn (Eds.), Spatial Information Theory A Theoretical Basis for GIS, volume 988 of *Lecture Notes in Computer Science*, Springer Berlin Heidelberg, 1995, pp. 341–349.

[22] R. Moratz, F. Dylla, J. Frommberger, in: Proceedings of the Workshop on Agents in Real-Time and Dynamic Environments, IJCAI 2005.

[23] J. O. Wallgrün, L. Frommberger, D. Wolter, F. Dylla, C. Freksa, in: Proceedings of the 2006 International Conference on Spatial Cognition V: Reasoning, Action, Interaction, Springer-Verlag, Bremen, Germany, 2007, pp. 39–58.

[24] S. Balakirsky, Z. Kootbally, C. Schlenoff, T. Kramer, S. Gupta, in: Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2012, Vilamoura, Algarve, Portugal.

[25] D. Nau, M. Ghallab, P. Traverso, Automated Planning: Theory & Practice, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2004.

[26] S. Carpin, M. Lewis, J. Wang, S. Balakirsky, , C. Scrapper, in: Proceedings of the 2007 IEEE International Conference on obotics and Automation, ICRA 2007, Vilamoura, Algarve, Portugal, pp. 1400–1405.

[27] H. Kitano, M. Asada, Y. Kuniyoshi, I. Noda, E. Osawa, in: Proceedings of the 1st International Conference on Autonomous Agents, AGENTS'97, Marina del Rey, CA, USA, pp. 340–347.