diagrams, and (4) how to perform automatic model verification and code generation of state diagrams for cooperating robots.

The organization of the paper is as follows. In Section 2, we describe a sample indoor environment referred to as environmental resources. The environmental triggers described in Section 3 can be used for specifying robot behavior. In Section 4, we discuss higher-level state diagrams with probabilistic transitions for robot navigation. In Section 5 we describe the underlying probabilistic model. In Section 6, the model is used to process probabilistic queries. In Section 7, we describe lower-level probabilistic state diagrams and their conversion to higher-level diagrams. In Section 8 we describe probabilistic state diagrams for cooperating robots.

## 2 ENVIRONMENTAL RESOURCES

From the point of view of the robot programmer it is important to determine the type of an indoor environment. In general, an indoor environment can be known or unknown. In this paper we will concentrate on describing robot behavior in a known environment. The known environment is typically described by a topological map identifying all landmarks and accessible places for robot occupancy and movement. One of the representations of a topological map can be a graph [1, 2] showing all accessible places in the form of nodes and the ways to get to these places in the form of graph paths.

Any topological map in the form of a graph can be also interpreted as a graph of environmental resources. It means that each node of the graph can be also interpreted as a resource and when the robot position is associated with this node we claim that the robot acquired the resource. The link between two nodes can be interpreted as a constraint in acquiring resources by a robot. Such an interpretation of a topological graph allows us to apply known resource allocation algorithms for the description of multiple robot behavior.

Let us assume, for our case study, the indoor environment of a corridor in the "I" shape that has two doors leading to two rooms. The topological graph interpreted also as an environmental resource graph can be constructed as shown in Figure 1.

The following nodes corresponding to environmental resources can be identified: (a) the beginning of corridor called "Corridor Beginning", (b) in front of the first door called "Door 1 Front", (c) between walls in the middle of the corridor called in short form "Corridor Middle 1 ", (c) in front of the second door called "Door 2 Front", (d) between walls in short form "Corridor Middle 2", (e) in front of the third door called "Door 3 Front" (f)inside first room called in short form "Room 1", (f) inside second room called in short form "Room 2", and (f) inside third room called in short form "Room 3". The link between two nodes e.g. "Corridor Beginning" and "Door 1 Front" can be interpreted as follows: if the robot is assigned a resource "Corridor Beginning" it should first acquire resource "Door 1 Front" before releasing resource "Corridor Beginning".

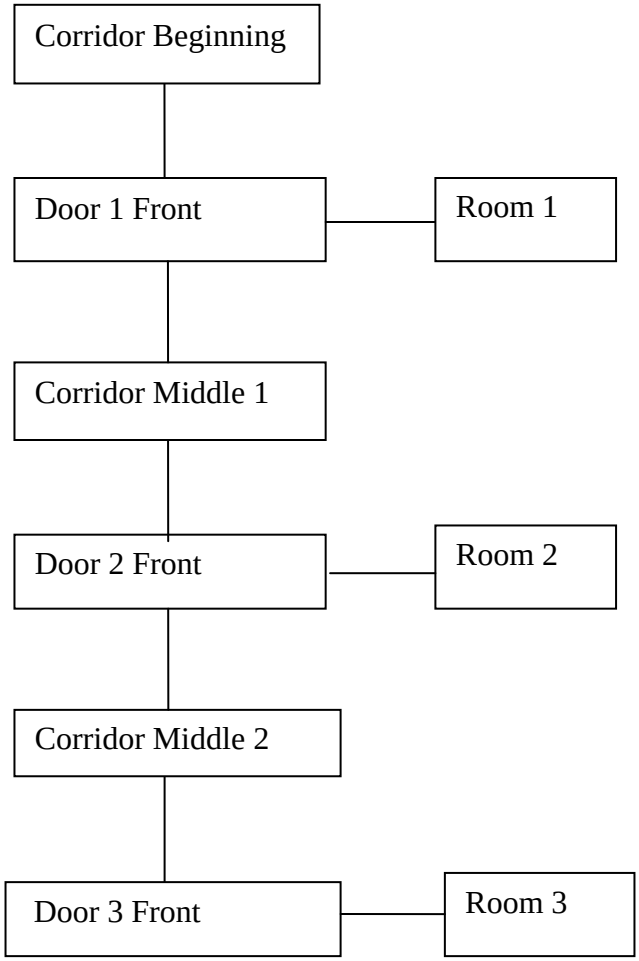


**Figure 1.** An Environmental Resource Graph.

## 3 ENVIRONMENTAL TRIGGERS

Different topological places i.e. different resources would usually generate different values for the robot's sensors. The sensor signal processing algorithms i.e. algorithms describing a translation of robot sensor signals into a high level signals that can be used to directly identify the environment. We will refer to these high level signals as the environmental triggers.

In our previous papers [10], we extensively studied the environmental triggers to identify wall of the corridor, end of the corridor, doors, door opening etc. Let us summarize the previous results. Various computer vision techniques can be used to create such environmental triggers. The Hough transform is widely used in computer vision for detecting line segments and regular geometric features such as line segments and circles in images. More specifically, Progressive Probabilistic Hough Transform [11] can be used for detecting naturally occurring lines in indoor images of corridors and hallways [10]. The histogram based difference methods can be used for discriminating between corridor features and for recognition of major landmark objects

[2] such as doors and wall surfaces between them.

Using Hough transform we can implement the environmental triggers to allow the robot to direct itself to the middle of the corridor. The OrientationPointToLeft and Orientation-PointToRight triggers are used to identify the misalignment.

Histogram based difference measurement can be used during indoor robot navigation to solve the significant problem of recognition of major landmark objects [2] such as doors and wall surfaces between them. When the robot moves beside a door and the edge of the camera detects the door pattern then the environmental trigger DoorPatternDetected becomes True otherwise this condition is False. Similarly, when the robot moves beside a wall and the edge of the camera detects wall pattern then the environmental trigger WallPatternDetected becomes True.

The other triggers e.g. Middle-OfDoorDetected can be derived based on previous triggers. When the trigger WallPatternDetected changes from False to True a robot odometer is initialized and the trigger MiddleOfDoorDetected becomes False. When the value of that odometer is equal or greater than half of the width of the door, then the trigger MiddleOfDoorDetected becomes True.

In addition, the triggers MiddleDoor-OrientationPointToLeft and Middle-DoorOrientationPointToRight are computed based on the hybrid analysis combining Hough Transform with histogram analysis. More specifically, the algorithm selects vertical lines separating surfaces based on histogram based differences and computes the orientation point as a point in the middle of these lines.

We also used simple environmental triggers such as Obstacle whose value was generated by a simple infrared sensor.

In this paper, we extended previous studies to include situations when the robot might not be able to recognize the landmarks or can recognize them with a certain probability. There are two cases here. The first case is when the existence of the landmark is immaterial for the robot behavior. For example, if the robot does not need to enter the door it might be immaterial if it cannot recognize the door. In such a case the reduction of the resource graph can be performed by removing unnecessary nodes.

The second case is when the existence of the landmark is crucial for robot behavior. The proposed techniques in this paper namely, probabilistic state diagrams can be used to model such behavior.

## 4 STATE DIAGRAMS WITH PROBABILISTIC TRANSITIONS FOR ROBOT NAVIGATION

The deterministic state diagrams are well described in literature [7, 8]. Generally, a deterministic state diagram, in addition to states, has transitions consisting of triggers that cause the transition of the robot from one state to another, and actions, that are invoked during a transition. Triggers are expressed by Boolean conditions evaluated continuously to respond to changes in the environment.

To specify state diagrams we use the notation based on Universal Modeling Language (UML) [11] where a state is indicated by a box and a transition is indicated by an arrow with a label. The first part of the transition label (before

the slash) specifies the trigger and the part after the slash specifies the action to be invoked during the transition [11]. The syntax of probabilistic specifications is described in the literature [6] as an additional third component specifying the probability of the entire transition.

State diagrams that are explicitly location dependent can be convenient to specify robot behavior for several reasons. Firstly, the diagram can be constructed by relatively simple transformation of environmental resource diagram. Second, probabilistic components can be added relatively easily. Thirdly, the behavior of cooperating robots can be described by concurrent state diagrams and all well-established techniques for concurrent program analysis with limited resources can be used i.e. deadlock detection or deadlock avoidance algorithms. The analysis of concurrency can be done automatically and the robot program can be directly generated from state diagram model.

Based on environmental graph and corresponding environmental triggers we can rapidly specify the various location dependent state diagram. For example, let us consider Behavior 1 of a robot: start from the corridor beginning, then follow the corridor until encountering the middle of the first door, then randomly either (a) turn itself towards the door, and then enter the room and stop, or (b) continue to follow the corridor until encountering the middle of the second door, then turn itself towards this door, and then enter the room and stop.

In order to model such behavior a two-level model is used. The higher, more abstract level can be obtained by transforming the environmental graph i.e. converting non-directional to directional edges and providing the necessary triggers, actions and probabilities. On the higher level the model describes only how the need for use of environmental resources is changing without specifying how each resource is used.

The second level model describes the robot behavior for each environmental resource. The models can be refined into an "implementable" model. By implementable model we mean the model specific enough so that it can be used to generate the program for the robot.

Another reason for using multi-level model is to create better abstractions for probabilistic processing.

The higher level model, in our case study, is represented by the state diagram shown in Figure 2. It specifies the robot Behavior 1 in some detail dividing it into several phases:

- Phase 1. Initially the robot is in the "In Corridor Beginning" state. In this state, the forward command is executed until it recognizes the door. This built-in command engages the motors to move the robot forward.
- Phase 2. When robot is in the "In Door 1 Front" state it continues moving forward until it reaches the door opening. After detecting the door it either enters Room 1 or continues going forward. The decision is randomly selected, each option with the probability 0.5.
- Phase 3. In the state "In Room 1" it continuously moves forward until encountering the end of Room 1.
- Phase 4. In the "In Corridor Middle 1" state the robot moves forward until encountering another door.
- Phase 5. When robot stays in the "In Door 2 Front" state it continues moving forward until it reaches the

door opening. After detecting the door it enters Room 2.

- Phase 6. In the state "In Room 2" it continuously moves forward until encountering the end of Room 2.
- Phase 7. In the "In Corridor Middle 2" state the robot moves forward until encountering another door.
- Phase 8. When robot stays in the "In Door 3 Front" state it continues moving forward until it reaches the door opening. After detecting the door it enters the Room 3.
- Phase 9. In the state "In Room 3" it continuously moves forward until encountering the end of Room 3.

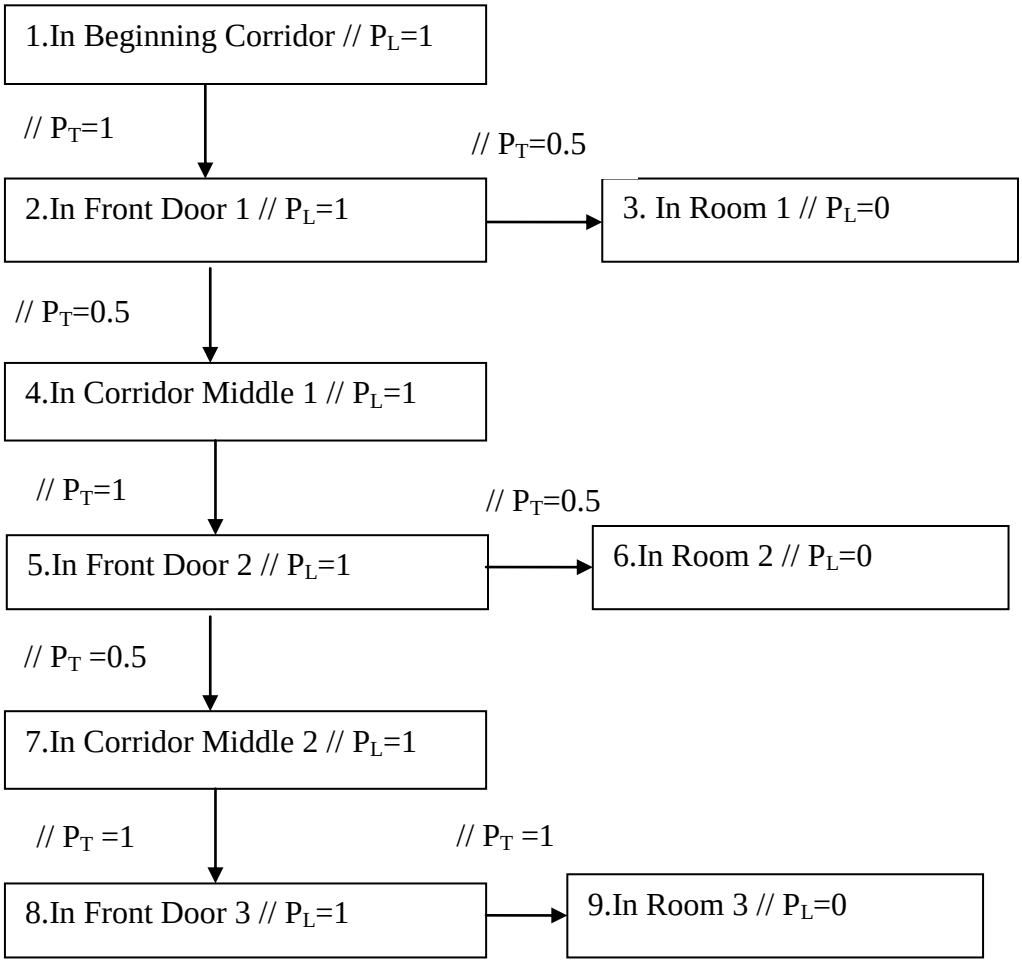


**Figure 2.** Higher-level State Diagram to describe Robot Behavior 1.

Creation of a high-level state diagram can be accomplished by using our graphical interface for cooperative design of multi-level state diagrams (GISAC2). The interface supports the following functions:

- Creation of the environmental graph by selecting menu operators: create-Resource and createResourceLink.
- Converting the environmental graph into high-level state diagram by selecting menu operators delete-Resource, duplicateResource, convert-ResourceLink, and addLabel (to add triggers, actions and probabilities).

## 5 PROBABILISTIC STATE DIAGRAM MODEL

There are several issues, however, that need to be discussed related to the syntax and the semantics of probabilistic specifications. Generally, in probabilistic state diagrams there can be a third component in any transition specifying the probability of that transition represented as a $P_T(S_i, S_j)$. In this section, we define the model and provide the notation we will use.

| | |
|---|---|
| S | Finite set of states, $\{S_i\}$ |
| T | Set of state transitions, which are pairs $(S_i, S_j)$. |
| $P_L(S_i)$ | Probability of (ever) leaving state $S_i$ given that it is the current state. |
| $P_T(S_i, S_j)$ | Probability of the next state being $S_j$ given that current state is $S_i$ and that the robot leaves $S_i$. |
| $S_{START}$ | The initial state at the beginning of the analysis. |

**Table 1.** Probabilistic state diagram model components.

A probabilistic state diagram is given as $(S, T, P_L, P_T, S_{START})$ where the components are defined in Table 1.

**Constraints.** These model parameters are subject to consistency constraints. First, all of the probabilities must be in the interval [0, 1]. Second, $P_T(S_i, S_j) = 0$ when $(S_i, S_j)$ is not in T. Third, for any $S_i$, the sum of $P_T(S_i, S_j)$ taken over all $S_j$ must be equal to 1.

**Example.** A diagrammatic example was given in Fig. 2. There, we have the components for the model as specified in Table 2.

Note that the probability of staying forever in State i ($S_i$) given that it occurred is 1 minus the probability of leaving, or $1 - P_L(S_i)$.

| | |
|---|---|
| S | {1, 2, 3, …, 9} |
| T | {(1,2), (2,3), (2,4), (4,5), (5,6), (5,7), (7,8), (8,9)} |
| $P_L(S_i)$ | $P_L(S_i) = 1$ for $S_i$ in {1, 2, 4, 5, 7, 8} and $P_L(S_i) = 0$ for $S_i$ in {3, 6, 9} |
| $P_T(S_i, S_j)$ | $P_T(S_i, S_j)$ for $(S_i, S_j)$ in T is 0.5 if $S_i$ in {2, 5}, and 1 otherwise |
| $S_{START}$ | 1 |

**Table 2.** Probabilistic state diagram model components for the example in Figure 2.

**Model Assumption.** The model assumes that the processing time within the state is unknown, but the transition time is viewed as instantaneous.

**Observations.** We have two main observations:

1. *Ubiquitous Termination.* A state that can be a final state for a robot can be viewed as a leaf in a tree diagram or a terminal state. A state with $P_L(S_i)=0$

always ends a state sequence (or path). However, any state $S_i$ with $P_L(S_i) < 1$ can be a final state. The model therefore allows, for some parameterization, for any state to be a terminal state. We call this property ubiquitous termination.

2. *Asynchronous.* Our model allows for state transitions to occur in continuous time. Unlike many models of state transitions that depend on a consistent universal synchronous clock tick, this model enables asynchronous simulations.

Based on the above definitions of probabilistic state diagrams we can compute interesting properties e.g. probability of ever reaching the state $S_i$. We will extend our notation and represent such probability as $P_R(S_i)$. The processing of such and other properties will be described in the next section.

## 6. QUERY PROCESSING FOR PROBABILISTIC STATE DIAGRAMS DESCRIBING ROBOT NAVIGATION

There can be many queries about robot behavior that can be answered based on probabilistic state diagrams. The queries can be classified into temporal and non-temporal queries. This paper is restricted to discussion of non-temporal queries; temporal queries are a topic of ongoing research. Among the non-temporal queries there are two computed most often:

**Query A.** What is the probability of ever reaching the state $S_i$? It can be denoted as $P_R(Si)$.

**Query B.** What is the probability of ending in the state $S_i$? It can be denoted as $P_E(S_i)$.

Let us describe processing these queries for different types of probabilistic state diagrams.

### 6.1. Query Processing for Probabilistic State Diagrams - Trees

Let us start the discussion of Probabilistic State Diagrams based on an example diagram in Figure 2. This diagram belongs to a general class of tree state diagrams.

First we introduce the notion of a path prefix probability. Next, we provide an algorithm to compute the probabilities. Finally, we show how this algorithm can be used to process queries A and B.

**Definition.** A *path* in a probabilistic diagram model is a sequence of states in the probabilistic state diagram where each consecutive pair of states is a transition in T.

**Definition.** The *path probability* of a path s is the probability of the state sequence in s as determined by the model parameters.

**Inductive Path Probability Algorithm for Trees.** Order the states in S such that all of the states that can lead into $S_i$ are listed before $S_i$. This is possible because we are assuming that the diagram is a tree. We can refer to these ordered states as:

$$S_{k1} = S_{START}, S_{k2}, \ldots, S_i, S_j, \ldots, S_{kn}.$$

For simplicity we assume that $S_{START}$ is the root node of the tree.

Let us first consider the special type of tree with all non-leaf nodes $S_i$ having $P_L(S_i)=1$. We will refer to such a tree as a transient tree.

Now, for each state $S_j$ in order, set $P_R(S_j)$ according to:

(a) $P_R(S_j) = 1$ if $S_j = S_{START}$
(b) $P_R(S_j) = P_R(S_i)\, P_T(S_i, S_j)$

This formula inductively computes the path prefix probabilities of reaching each state. The correctness of the formula follows from the uniqueness of the path from the root to any state and from the decomposition of the event into (1) reaching the parent, and (2) choosing the particular child.

Note that the probability of staying forever in State i denoted by $P_E(S_i)$ is the probability of reaching the state $P_R(S_i)$ times 1 - $P_L(S_i)$:

$$P_E(S_i) = P_R(S_i)\,(1 - P_L(S_i)).$$

The computation can then be used to process Query A and Query B for transient trees. We can apply the algorithm to each of the 9 states in our example diagram in Figure 2 as shown in Table 3.

| State | Rule | Computation for $P_R$ | $P_R$ | $P_E$ |
|---|---|---|---|---|
| 1 | (a) | 1 | 1 | 0 |
| 2 | (b) | 1 · 1 | 1 | 0 |
| 3 | (b) | 1 · 0.5 | 0.5 | 0.5 |
| 4 | (b) | 1 · 0.5 | 0.5 | 0 |
| 5 | (b) | 0.5 · 1 | 0.5 | 0 |
| 6 | (b) | 0.5 · 0.5 | 0.25 | 0.25 |
| 7 | (b) | 0.5 · 0.5 | 0.25 | 0 |
| 8 | (b) | 0.25 · 1 | 0.25 | 0 |
| 9 | (b) | 0.2 · 1 | 0.25 | 0.25 |

**Table 3.** Probabilistic state diagram model components for the robot Behavior 1.

Note that the ending probabilities sum to the value of one. The reaching probabilities will not generally sum to one because most paths will traverse multiple states. Consider the computation for state 5. State 5 can be reached only from state 4. The probability of reaching state 5 is the probability of having been in state 4, and then selecting state 5. These two parts have probabilities 0.5, and 1. Their product is computed in the table as the probability of reaching state 5.

The case of state diagram with non-transient tree i.e. the tree with non-leaf nodes having the probability $P_L(S_i)$ less than one, requires additional discussion.

This case is often present when the robot gets "stuck" intentionally or unintentionally while doing some task and therefore it is important to model it in the probabilistic state diagram.

Let us assume that the robot from the previous section, while in the state 4, can recognize the second door only with the probability of 0.8 as shown in Figure 3.

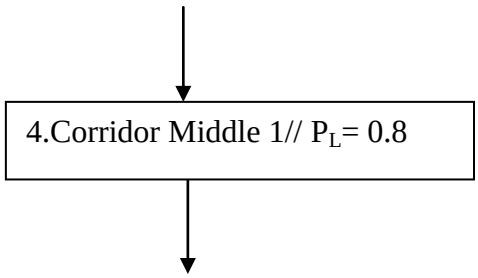


**Figure 3.** Partial Modification of State Diagram in Figure 2 to describe Behavior 2.

For a non-transient state diagram the algorithm is slightly modified. Now, for each state $S_j$ in order, set $P_R(S_j)$ according to:

(a) $P_R(S_j) = 1$ if $S_j = S_{START}$
(b) $P_R(S_j) = P_R(S_i)\, P_L(S_i)\, P_T(S_i, S_j)$

This formula again inductively computes the path prefix probabilities of reaching

each state. The correctness of the formula follows from the uniqueness of the path from the root to any state and from the decomposition of the event into (1) reaching the parent, (2) leaving the parent, and (3) choosing the particular child.

The formula to compute probability of staying forever in State i denoted by $P_E(S_i)$ is the same:

$$P_E(S_i) = P_R(S_i)\,(1 - P_L(S_i)).$$

The computation can then be used to process Query A and Query B for robot Behavior 2 as shown in Table 4.

| State | Rule | Computation | $P_R$ | $P_E$ |
|---|---|---|---|---|
| 1 | (a) | 1 | 1 | 0 |
| 2 | (b) | 1 · 1 · 1 | 1 | 0 |
| 3 | (b) | 1 · 1 · 0.5 | 0.5 | 0.5 |
| 4 | (b) | 1 · 1 · 0.5 | 0.5 | 0.1 |
| 5 | (b) | 0.5 · 0.8 · 1 | 0.4 | 0 |
| 6 | (b) | 0.4 · 1 · 0.5 | 0.2 | 0.2 |
| 7 | (b) | 0.4 · 1 · 0.5 | 0.2 | 0 |
| 8 | (b) | 0.2 · 1 · 1 | 0.2 | 0 |
| 9 | (b) | 0.2 · 1 · 1 | 0.2 | 0.2 |

**Table 4.** Probabilistic state diagram model components for the robot Behavior 2.

Consider the computation for state 5. State 5 can be reached only from state 4. The probability of reaching state 5 is the probability of having been in state 4, of then leaving state 4, and then selecting state 5. These three parts have probabilities 0.5, 0.8, and 1. Their product is computed in the table as the probability of reaching state 5.

### 6.2. Query Processing for Probabilistic State Diagrams - DAGs

Let us extend the discussion of Probabilistic State Diagrams using an initial example modified as shown in Figure 4. This resulting diagram belongs to a general class of Directed Acyclic Graph (DAG) state diagrams.

Query A cannot be answered for DAGs by our tree algorithm. The problem is that there is no longer a unique path to each state. However, Query A can be answered either by an appropriate DAG traversal with probability computations at each node, just as before. In particular, we can update the inductive probability rules.

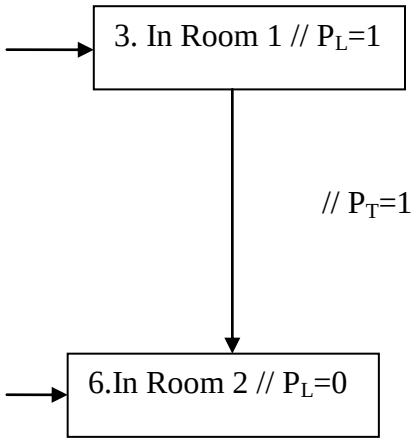


**Figure 4.** Modified state diagram fragment from Figure 2 to describe Behavior 3.

**Inductive Path Probability Algorithm for DAGs.** Order the states in S such that all of the states that can lead into $S_i$ are listed before $S_i$. This is possible because we are assuming that the diagram is a DAG.

Now, for each state $S_j$ in order, set $P_R(S_j)$ according to:

(a) $P_R(S_j) = 1$ if $S_j = S_{START}$
(b) $P_R(S_j) = 0$ if $S_j$ has no states leading into it but $S_j$ is not $S_{START}$,
(c) $P_R(S_j) = \Sigma_{Si} P_R(S_i) P_L(S_i) P_T(S_i, S_j)$ where $S_i$ is taken over all states having $(S_i, S_j)$ as a transition.

This formula inductively computes the path prefix probabilities of reaching each state. The correctness of the formula follows from using the law of total probabilities to combine probabilities from all paths from the root to any state. As in the tree algorithm, we decompose the event into (1) reaching the parent, (2) leaving the parent, and (3) choosing the particular child.

However, now we sum over all predecessors to obtain the total probability.

| State | Rule | Computation | $P_R$ | $P_E$ |
|---|---|---|---|---|
| 1 | (a) | 1 | 1 | 0 |
| 2 | (c) | 1 · 1 · 1 | 1 | 0 |
| 3 | (c) | 1 · 1 · 0.5 | 0.5 | **0** |
| 4 | (c) | 1 · 1 · 0.5 | 0.5 | 0.1 |
| 5 | (c) | 0.5 · 0.8 · 1 | 0.4 | 0 |
| 6 | (c) | **0.4 · 1 · 0.5 + 0.5 · 1 · 1** | **0.7** | **0.7** |
| 7 | (c) | 0.4 · 1 · 0.5 | 0.2 | 0 |
| 8 | (c) | 0.2 · 1 · 1 | 0.2 | 0 |
| 9 | (c) | 0.2 · 1 · 1 | 0.2 | 0.2 |

**Table 5. Probabilistic state diagram model components for the robot Behavior 3.**

This algorithm can then be used to process Query A and B. For example, in Figure 4, we compute probability of reaching the state 6 as the sum:

$$P_R(S_6) = P_R(S_3)\, P_L(S_3)\, P_T(S_3, S_6) + P_R(S_5)\, P_L(S_5)\, P_T(S_5, S_6)$$
$$P_R(S_6) = 0.2+0.5 = 0.7.$$

We can in the same way obtain the probabilities of three final states with assigned probability of ever reaching these states: $P(S_4) = 0.1$, $P(S_6) = 0.7$, $P(S_9) = 0.2$. Obviously, the sum of these probabilities should be equal to 1. In full detail, we have described this in Table 5, with changes indicated in bold.

### 6.3. Query Processing for Probabilistic State Diagrams - Graphs

Let us extend the discussion of Probabilistic State Diagrams using an example of a diagram in Figure 5 describing robot Behavior 4. This diagram belongs to a general class of structured state diagrams with one loop. We will refer to a structured state diagram when loops can be nested inside other loops. Let us start with the example of a single loop as shown in Figure 5.

In order to answer Query A, some repetitive loop (of the graph) traversal with probability computing can be attempted or analytical transformation performed.

Let us first distinguish two types of Query A. The first type of query A is the query asking for a probability of ever being in the state i where the state i is outside the loop. In this situation we create a higher level state diagram with the new state Loop 1 that is a loop abstraction. In the higher level diagram instead of asking for the state i inside the loop we can ask for probabilities of leaving the abstract state called Loop 1 and transitions to the states outside the loop as shown in Figure 6.

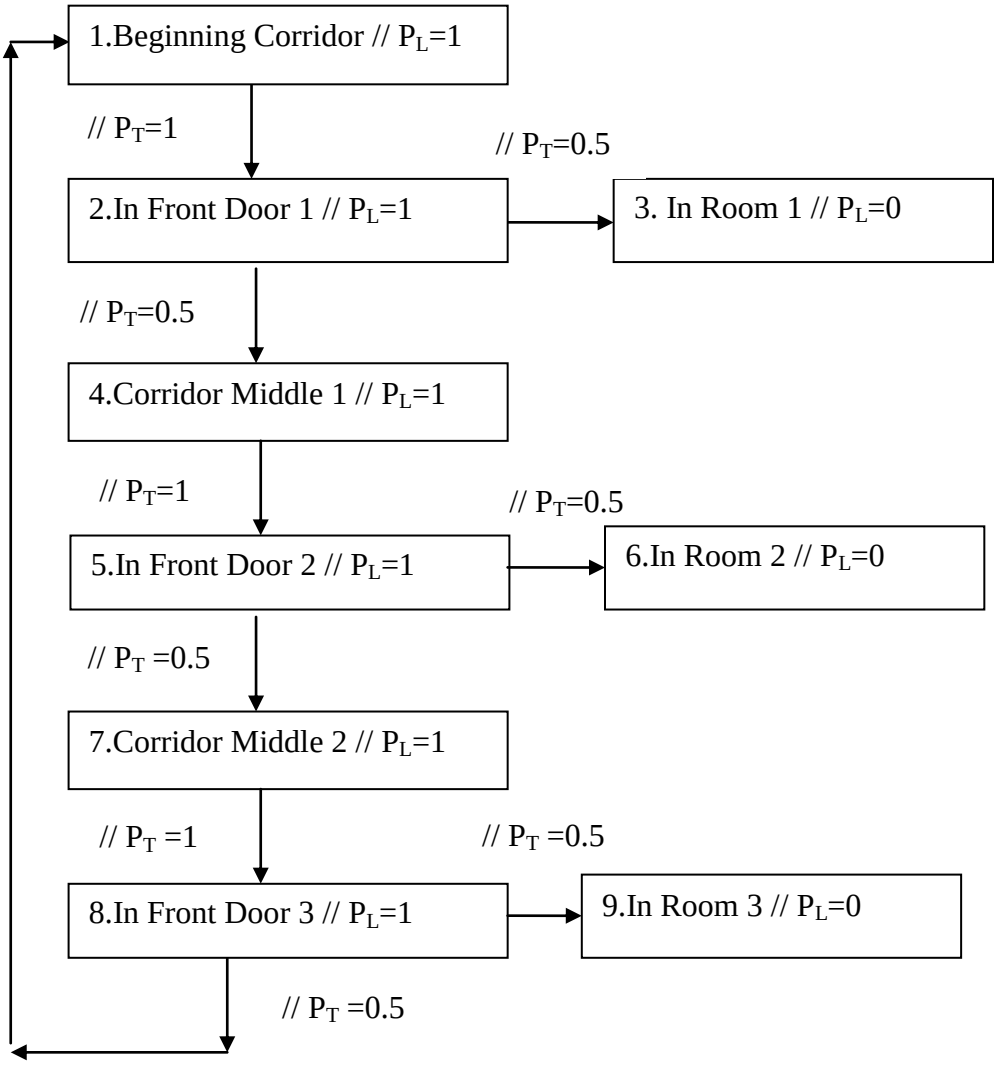


**Figure 5.** State Diagram to describe Behavior 4.

Let us look first at the probability of leaving the Loop after one cycle (we can imagine breaking the cycle by copying state 1 to 1' but it is not necessary) from our example. Such case can be described by the previous formulas for the non-loop computations. The probability for transition in the first cycle into State 3 is 0.5, to State 6 is 0.25 and to state 9 is 0.125. Getting back into state 1 is 0.125 according to our computations rules described in the previous sections. Let us denote an abstract transition from state 1 back to itself as a transition $P_T(S_1,S_1)$.

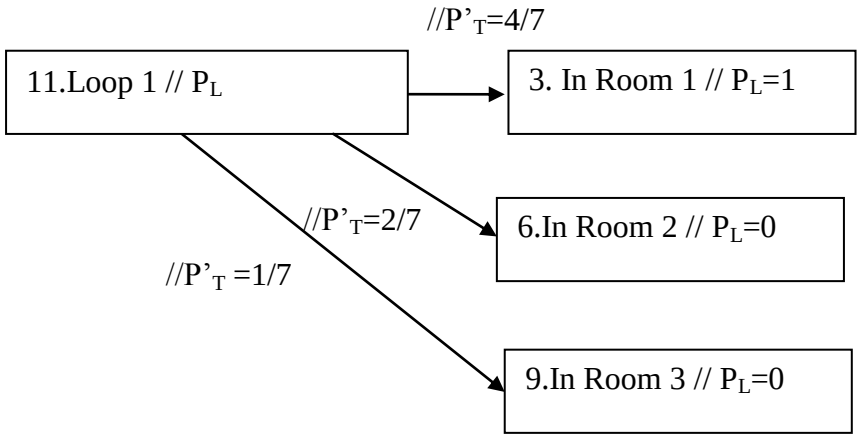


**Figure 6.** Higher-level State Diagram to describe Behavior 4.

If we repeat this process infinitely (compute limit) we will obtain the correct probabilities staying in the Loop 1 state and transitions to other states. In the case of a loop represented by state 11 (in the higher-level diagram), and an escape state 3, we have:

$$P_R(S_3) = P_T(S_1,S_3) + P_T(S_1, S_1)\, P_T(S_1,S_3) + P_T(S_1, S_1)^2\, P_T(S_1,S_3) + \ldots.$$

Which, being a geometric sum, can be rewritten to the formula

$$P_R(S_3) = \Sigma_{k=1 \text{ to infinity}}\, P_T(S_1, S_1)^k\, P_T(S_1,S_3)$$

$$P_R(S_3) = P_T(S_1,S_3) / (1 - P_T(S_1, S_1))$$

In our case the values for abstract transitions (indicated by a prime) in the higher level diagram are as follows:

$$P'_T(S_{11}, S_3) = 0.5 / (1 - 0.125) = 4/7$$
$$P'_T(S_{11}, S_6) = 0.25 / (1 - 0.125) = 2/7$$
$$P'_T(S_{11}, S_9) = 0.125 / (1 - 0.125) = 1/7$$

The structured state diagrams with many nested loops can be computed similarly to no loop graph. Starting from the most inner loop, we convert it to an abstract Loop 1 state. Then we proceed with the second loop, Loop 2, etc. We end up with the hierarchy of state diagrams where the numbers of levels in the hierarchy correspond to the number of nested loops.

## 7. COMPUTING PROBABILISTIC PROPERTIES OF HIGHER-LEVEL DIAGRAMS FROM LOWER-LEVEL STATE DIAGRAMS.

The lower-level diagrams describe all details of behavior within the environmental resource and define the probabilities of a higher-level diagram i.e. probability for leaving the resource and probabilities for different paths for leaving the resource (assuming that it is left). Let us describe these diagrams in an order.

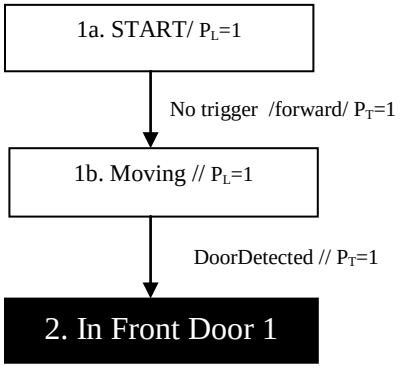


**Figure 7.** Lower-level State Diagram to describe phase 1 or robot Behaviour 1.

The lower-level diagram associated with the environmental resource "Beginning Corridor" is shown in Figure 7. The state diagram includes typical robot actions such as forward, and environmental triggers such as doorDetected. Initially the robot is in the "START" state and with no trigger (or an external trigger), the transition to the "Moving" state takes place. During the time of the transition, the forward command is executed. This built-in command engages the motors to move the robot forward. Once the trigger doorDetected becomes true the transition to the exit state (modeled as a black rectangle) takes place that is linked to the state "In Front Door 1" in the higher level diagram.

Now, the probabilities $P_L$ and $P_T$ can be computed for the State 1 in the higher level diagram shown in Figure 2, using the previously defined formulas. For the higher-level State 1 in Figure 2, $P_L(S_1)$ = 1 and $P_T(S_1,S_2)$ = 1.

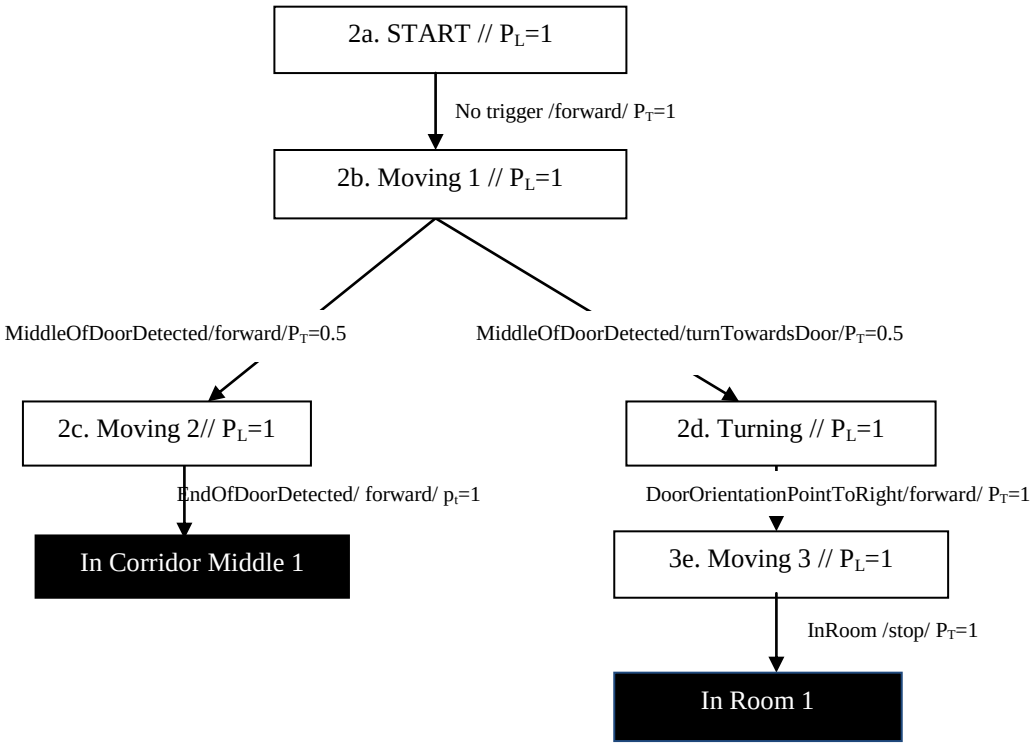


**Figure 8.** Lower-level State Diagram to describe phase 2 of robot Behavior 1.

The lower-level diagram associated with the environmental resource "In Front Door 2" is shown in Figure 8. Initially the robot is in the "START" state and with no trigger; the transition to the "Moving 1" state takes place. The robot stays in the "Moving 1" state (and continues moving forward) until it detects the middle of the door, and the trigger condition MiddleOfDoorDetected becomes True. Detecting the middle of the door triggers the probabilistic transitions to either the "Turning" or "Moving 2" states. Each transition can have a different action i.e. the transition to "Turning" state is associated with action to turn left whereas the transition to "Moving 2" is associated with action to move forward. While in the "Turning" state detecting the room triggers the transitions to "In Room 1" state. While in the "Moving 2" state detecting the room triggers the transitions to the "In Corridor Middle 1" state.

Now, again, the probabilities $P_L$ and $P_T$ can be computed for the State 2 in the higher level diagram from Figure 2, using the previously defined formulas. For the higher-level State 2 in Figure 2, $P_L(S_2) = 1$, $P_T(S_2,S_3) = 0.5$ and $P_T(S_2,S_4) = 0.5$.

Another lower-level diagram can be defined by association with the environmental resource "Room 1". Initially the robot is in the "START" state and with no trigger, the transition to the "Moving" state takes place the robot moves forward to Room 1 until encountering the obstacle (the end of Room 1).

In the similar process a lower-level diagram can associated with the environmental resource "Corridor Middle 1".

Initially the robot is in the "START" state and with no trigger, the transition to the "Moving" state takes place the robot moves forward until encountering the door.

The lower-level can also be associated with the environmental resource "Front Door 2". Initially the robot is in the "START" state and with no triggers, the transition to the "Moving 1" state takes place. The robot stays in the "Moving 1" state (and continues moving forward) until it detects the middle of the door, and the trigger condition MiddleOfDoorDetected becomes True. Detecting the middle of the door triggers the transition to the "Turning" state. While in the "Turning" state detecting the room triggers the transitions to "In Room 2" state. The remaining lower-level state diagrams can be defined similarly.

Creation of a lower-level state diagram can be accomplished using our graphical interface for design of multi-level state diagrams (GISAC 2). The interface supports the creation of the lower-level state diagram by selecting traditional operators as described in [9].

There are several additional features of GISAC 2 that can be listed: (a) For a more complex robot behavior, when the robot "visits" the same environmental resource several times, the graph node for the environmental resource can be duplicated. (b) There are many levels of abstractions that allow handling large state diagrams. (c) The state diagram fragment reusability is available and convenient to replicate a robot behavior. (d) The models can be converted to a code and executed. In our experiments we generated the code in the Python language for the Scribbler robot [4].

## 8. QUERY PROCESSING FOR COOPERATION OF ROBOTS

In this section we discuss the use of state diagrams to describe and ensure the proper cooperation between autonomous robots. There are many theoretical and practical solutions that can be taken into consideration.

The model checking provides a most general methodology [13, 14, 15, 16, 17] that can be used not only for deadlock avoidance or detection but also for detection and verification of wide variety of process cooperation characteristics. Typically the model checking is based on finite-state methods [14] that can be applied directly to our state diagrams. Therefore, it can be of important practical use for verifying some robot behaviors. Unfortunately, for the described asynchronous and probabilistic models the traditional model checking method cannot offer a set of ready-to-use algorithms and techniques for the analysis of properties of a multiple robots system.

Our approach is similar to the model checking approach in its initial phases. Our approach uses, however, also other techniques e.g. game theory [18, 19]. Let us describe our approach in detail. First, out of the specifications of the robots' behavior by state diagrams we build a possibly large but finite graph containing all possible (reachable) system states and all possible transitions among them. This graph defines the behavioral model of the system of robots. Each path in the graph represents an allowable execution or a (part of a) behavior of a system. The graph contains all possible executions or behaviors. The property list can be used for the graph correctness specification.

Next the investigation of the probabilistic features of individual diagrams and a combined graph can be pursued. One activity is to search the combined system's graph trying to determine whether the desired probabilistic property holds.

In our approach the exponential growth of state space is still a real threat [17]. We included in our methodology multiple forms of reduction of state space, aimed at removing the states and transitions which are irrelevant for the evaluation of a given formula. We also can apply the techniques of compositional model checking, i.e. where some individual parts of a system (of a more acceptable size) are subject to an exhaustive state space search while the conclusion as to the performance of the whole system is reached by combining the results obtained for the individual parts.

Let us consider again the state diagram for robot behavior as specified in Figure 2. Let us assume that we have two robots. Robot A behavior is exactly as in Figure 2 (Behavior 1). Robot B behavior (Behavior 5) is similar, but it starts from the end of the corridor (In Front Door 3) and the robot moves the opposite way. This is one of the often present cases when the behavior of robots is dependent. The analysis of a combined diagram might be necessary.

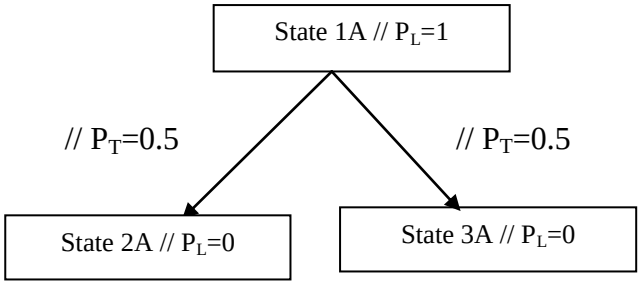


**Figure 9.** A state diagram to describe Behavior 1 of a robot A in an abstract form.

To be specific let us discuss the necessary analytical transformations in case of two robots acting according to Behavior 1 and Behavior 5. First, some form of reduction of state space is necessary to make the combined diagram manageable. One of the forms of the reduction can be based on state abstraction similar to abstractions (lower- and higher-diagrams) defined in previous sections.

Let us assume that we are interested in analyzing any potential collisions between our two robots A and B. We will reduce both diagrams to three state diagrams with uppercase letter A and B identifying each robot label. In the first diagram, let us rename the original state 1 into State 1A, the original state 3 into State 3A. The abstract State 2A combines two original states 6 and 9. Both probabilities of transitions are obviously 0.5 as shown in Figure 9. Similarly, we can create an identical abstract diagram for the robot B.

As discussed before [19], out of the specifications of the 2 robots' behavior by state diagrams we can build a possibly a single finite graph containing all possible (reachable) system states and all possible transitions among them as shown in Figure 10.

The system properties show on the graph may be surprising. If we assume, the synchronized transitions as in [19], then each transition would have probability $P_T$=0.5, which is intuitively accepted.

Since our model is asynchronous we really do not now the time when the transitions occur. We only know that the probability of reaching, say, State 2a/1b is less than 0.5 since we do not know the time of reaching of the appropriate state of the respective robot. If we know duration of time in each state we could compute probabilities more precisely.

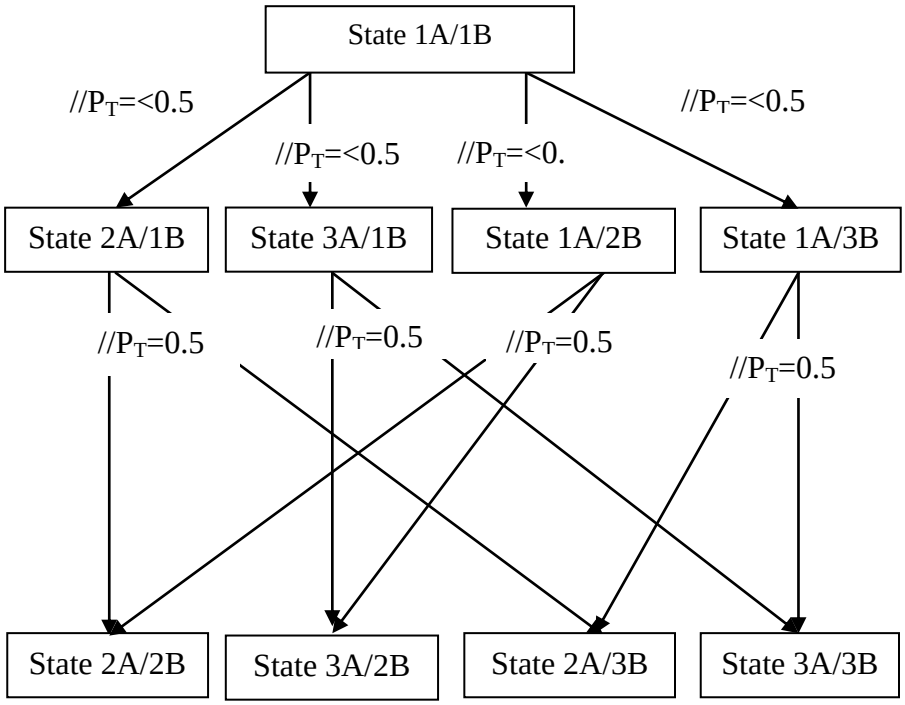


**Figure 10.** A single graph for a complete system of cooperating robots.

Still many properties can be checked against such a graph. For example, the probability of entering Room 1 by robot A and Room 3 by robot B can be computed based on analytical mode. The result is 0.5*0.5=0.25 since this is a final event and therefore it is time independent.

The typical analysis of the combined graph involves checking the provided parameters e.g. restrictions on using environmental resources. The robot movement collisions possibility can be investigated. Once the needed environment resources are specified for each robot, the potential "bottlenecks" can be identified.

The above discussion shows that there are tasks related to multi-level probabilistic state diagram that can be automated. Our GISAC 2 system supporting probabilistic state diagrams implements five main tasks as shown in Figure 11. First, it uses a graphical interface to specify the resource graph. Second, it allows a user to convert the resource graph into a probabilistic state diagram using a similar graphical framework. Third, it can simulate execution of the probabilistic state diagrams. Fourth, it can analyze combined diagrams e.g. computes combined probabilities if requested. Fifth, it generates the code which can be executed by robots.

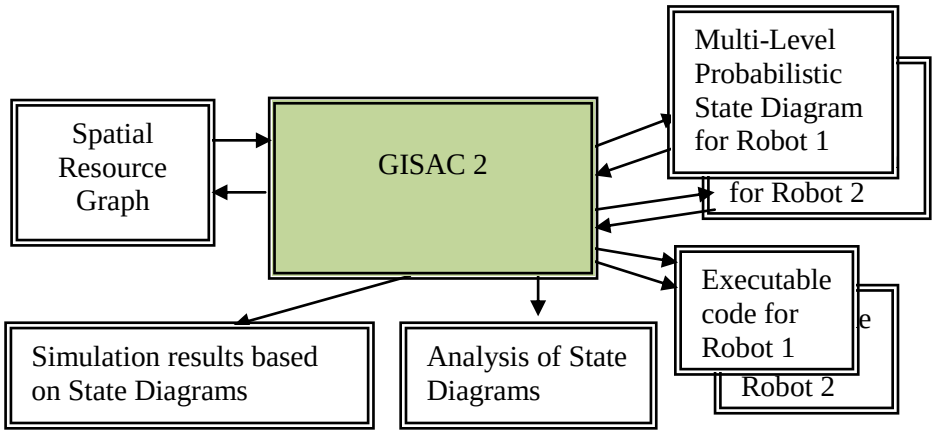


**Figure 11.** Architecture of a system supporting modeling of behavior of cooperating robots**.**

There are many extensions of the system to be included in the future. The most important is to include time modeling. The probability can be also introduced for actions e.g. random speed of the robot.

## 9. SUMMARY

In this paper, we have presented a novel approach for autonomous robot navigation in a known indoor environment using probabilistic state diagram integrated with computer vision techniques. Triggers were developed for state diagram based to detect to changes in the environment and respond according to a given probabilistic specifications.

The techniques were presented to verify and analyze individual probabilistic robots' behaviors, and combine individual robots' behaviors into a single finite graph describing a complete system. It was shown how to use techniques such as state abstraction and transition abstraction to create, verify and combine large probabilistic state diagrams. The probabilistic specifications allowed us to model more advanced cooperation between autonomous robots. We addressed the problem of ensuring the correctness of combined multiple robot behavior.

## 10 ACKNOWLEDGEMENTS

This research was supported in part by an appointment to the Higher Education Research Experiences (HERE) Program at the Oak Ridge National Laboratory (ORNL) for Faculty, sponsored by the U.S. Department of Energy and administered by the Oak Ridge Institute for Science and Education. It was also supported in part by NSF (award id: 0959958) and the Belk Foundation.

## 11 REFERENCES

1. Murphy, R. (2000) *Introduction to AI Robotics*, MIT Press.
2. Desouza, G.N. Kak, A.C. (2002) "Vision for Mobile Robot Navigation: a Survey", Pattern Analysis and Machine Intelligence, IEEE

Transactions on, Volume 24, Issue 2, 237-267.
3. Blank, D. (2006) "Robots Make Computer Science Personal", Communications of the ACM, 49, (12).
4. Kumar, D. (Ed.) (2007) *Learning Computing with Robots (Python)*, 1st Edition, Institute for Personal Robots in Education.
5. Meng, Y. (2008) "Multi-Robot Searching using Game Theory Based Approach", Journal of Advanced Robotics Systems.
6. Czejdo, B. D., Bhattacharya, S., and Czejdo, J. (2010) "Use of Probabilistic State Diagrams for Robot Navigation in an Indoor Environment". Proceedings of the Annual International Conference on Advanced Topics in Artificial Intelligence (ATAI), pp. A-97 to A-102.
7. Harel, D. (1988) "On Visual Formalisms", Communications of the ACM, 31, (5), 514-530.
8. Rumbaugh, J., Blaha, M., Premerlani, W., Eddy, F., Lorensen, W. (1990) *Object-Oriented Modeling and Design*, Prentice Hall, New Jersey.
9. Czejdo, B. D. and Bhattacharya, S., (2009) "Programming Robots with State Diagrams", Journal of Computing Sciences in Colleges, Volume 24 , Issue 5, 19-26.
10. Bhattacharya, S., Czejdo, B. D. & Mobley, S. (2009), "An Integrated Computer Vision and Infrared Sensor Based Approach to Autonomous Robot Navigation in an Indoor Environment". Proceedings of the 7th International Conference on Computing, Communications and Control Technologies.
11. Galamhos, C., Matas, J., Kittler, J. (1999) "Progressive Probabilistic Hough Transform for Line Detection", Conference on Computer Vision and Pattern Recognition, 1999, IEEE Computer Society. Volume 1, 560–566.
12. McMillan, K. L. (1993) *Symbolic Model Checking*, Kluwer Academic Publishers.
13. Peled, D. A. (2001) *Software Reliability Methods*, Springer.
14. Clarke, E.M. & Wing, J. M. (1996) "Formal Methods; State of the Art and Future Directions", ACM Computing Surveys, December, vol. 28, nr 4, 627 – 643.
15. Bryant, R. E. (1995) "Binary Decision Diagrams: Enabling Technologies for Formal Verification", Proc. IEEE/ACM Int. Conf. on Computer-Aided Design, 236-243.
16. Holzmann, G. J. (1997) "The Model Checker SPIN", IEEE Trans. on SE, Vol. 23, No. 5, May, 279-295.
17. Berard, B. (Ed.) (2001) Systems and Software Verification: Model-Checking Techniques and Tools, Springer Verlag.
18. Emery-Montemerlo B., (2005) "Game-Theoretic Control for Robot Teams," doctoral dissertation, tech. report CMU-RI-TR-05-36, Robotics Institute, Carnegie Mellon University, August.
19. Czejdo, B. D., Bhattacharya, S., and Baszun, M. (2011) "Use of Multi-Level State Diagrams for Robot Cooperation in an Indoor Environment". Proceedings of the International Conference on Digital Information Processing and Communications (ICDIPC2011), July, 2011.