Proc. of IEEE International Conference on Robotics and Automation, Barcelona, Spain, May 2005

ASyMTRe: Automated Synthesis of Multi-Robot Task Solutions through Software Reconfiguration

Fang Tang and Lynne E. Parker
Distributed Intelligence Laboratory, Department of Computer Science,
University of Tennessee, Knoxville, TN 37996-3450 Email: {ftang|parker}@cs.utk.edu

***Abstract*— This paper describes a methodology for automatically synthesizing task solutions for heterogeneous multi-robot teams. In contrast to prior approaches that require a manual predefinition of *how* the robot team will accomplish its task (while perhaps automating who performs which task), our approach automates both the *how* and the *who* to generate task solution approaches that were not explicitly defined by the designer *a priori*. The advantages of this new approach are that it: (1) enables the robot team to synthesize new task solutions that use fundamentally different combinations of robot behaviors for different team compositions, and (2) provides a general mechanism for sharing sensory information across networked robots, so that more capable robots can assist less capable robots in accomplishing their objectives. Our approach, which we call ASyMTRe (Automated Synthesis of Multi-robot Task solutions through software Reconfiguration, pronounced "Asymmetry"), is based on mapping environmental, perceptual, and motor control schemas to the required flow of information through the multi-robot system, automatically reconfiguring the connections of schemas within and across robots to synthesize valid and efficient multi-robot behaviors for accomplishing the team objectives. We validate this approach by presenting the results of applying our methodology to two different teaming scenarios: *altruistic* cooperation involving multi-robot transportation, and *coalescent* cooperation involving multi-robot box pushing.**

***Index Terms*— Multi-robot teams, behavior synthesis**

I. Introduction

When dealing with heterogeneous multi-robot teams, two issues are particularly challenging: (1) determining how to share sensor and perceptual resources across heterogeneous team members, and (2) determining the appropriate teaming behaviors to accomplish a task when the definition of *how* to solve a task is dependent on the available collection of robots and their sensory, perceptual, and motor resources. In typical approaches to multi-robot teaming, the task tree describing the decomposition of the team task into subtasks is defined by the human designer in advance of the robot team performance. The robots then choose from one of the alternative task decomposition trees, followed by the use of an automated approach for task allocation to determine the dynamic mapping of subtasks to robots. In these typical approaches, the predefined task decomposition tree defines the available multi-robot task solutions in advance of the mission (i.e., the *how*). This paper describes a new methodology for automating the synthesis of multi-robot task solutions in a general way that addresses both of the above challenges.

Our approach, which we call ASyMTRe (Automated Synthesis of Multi-robot Task solutions through software Reconfiguration, pronounced "Asymmetry"), automates the task solution generation process by providing the ability for heterogeneous robots to collaborate to find new solutions to tasks through various combinations of sensing, effectors, and behaviors that may be distributed across multiple robots. ASyMTRe enables a close, dynamic cooperation amongst heterogeneous team members to accomplish tasks that might be impossible for a single type of robot to achieve, even if it were duplicated multiple times.

Our ASyMTRe automated task synthesis approach is inspired by the concept of information invariants [5], which showed the equivalences between different combinations of sensing, communication, and action based upon information content. ASyMTRe allows the robots to reason about how to solve an application based upon the fundamental information needed to accomplish the task. The information needed to activate a certain behavior remains the same regardless of the way that the robot may obtain or generate it. Robots can collaborate to define different task strategies in terms of the required flow of information in the system.

The basic building blocks of our approach are collections of environmental sensors, perceptual schemas [9], motor schemas [1], and a simple new component we introduce, called *communication schemas*. These schemas are assumed to be preprogrammed into the robots at design time, and represent fundamental individual capabilities of the robots. Perceptual schemas process input from environmental sensors to provide information to motor schemas, which then generate an output control vector corresponding to the way the robot should move in response to the perceived stimuli. Communication schemas transfer information between various schemas. ASyMTRe automatically determines the proper connections between the sensors and the schemas – across multiple robots – to ensure that the team-level goals are achieved.

The rest of this paper is organized as follows. Section II introduces the formal definition of the problem with motivating examples. Section III explains our approach. We present the details of the experiments that are used to validate our approach in Section IV. Our results are shown and discussed in Section V. We end the paper with a review of related work and a discussion of future work and conclusions.

II. The Problem

The problem we address in this paper is the automation of task solution synthesis that enables a collection of heteroge-

neous robots to reorganize into subteams as needed depending upon the requirements of the application tasks and the sensory, perceptual, and effector resources available to the robots. We assume that there is a sufficient mixture of robot capabilities available to solve the problem, although those capabilities may be distributed across multiple robots. An additional assumption is that with a large team of heterogeneous robots, different combinations of robots will be able to solve certain tasks in different ways. We also assume that the robots are team members that share high-level goals and the intent is to cooperate with each other to ensure that the high-level goals are achieved.

A. Motivating Example

To motivate the problem, consider a simple multi-robot transportation application, in which robot team members are given the task of transporting themselves from a set of starting locations to a set of goal positions (one for each robot). This task requires that each robot be able to localize itself relative to its goal position, and to move in a way that reduces this distance to zero. If every robot has these capabilities, a straightforward approach would be to have each robot navigate to its goal independently, e.g., using laser range scanner-based localization. However, if some robots do not have the sensing capabilities to localize themselves relative to their goals, an alternative solution would be for the more capable robots to guide the less capable robots toward their goals by providing them with relative positioning information, perhaps through camera observations. The less capable robot can use this information obtained through another robot's sensors to allow it to accomplish its task. In fact, this particular solution strategy has been demonstrated in our previous work of mobile sensor net deployment [13], in which a capable robot helped other simpler robots deploy to required positions using visual sensing.

In other team compositions, alternative strategies for achieving the transportation application could be imagined, in which different combinations of sensors could be used to generate the information needed to solve the task. It is important to note, however, that the resulting robot behaviors for accomplishing the task could be dramatically different depending upon the combination of sensors that is selected for solving the task (see [12] for a further discussion of this issue). In one case of the transportation problem, a robot would be directly invoking a go-to-goal behavior, whereas in another case, a robot must maintain another robot within its field of view so as to transmit relative position information to that robot, followed perhaps by an invocation of its own go-to-goal behavior once the less capable robot has reached its own goal. Therefore, it is important that the solution approach also synthesize the correct combination of motor behaviors to achieve the goal in light of the sensory distribution that is present and the configuration of schema connections that are generated. In summary, the problem we address is, given a robot team, automatically synthesize an effective task solution by reconfiguring the schema connections across robots to ensure that all robots have access to all information needed to accomplish their objectives.

B. Formalism of the Problem

We formalize the automated task synthesis problem as follows. Given:

- A collection of *n* robots, denoted $R = \{R_1, R_2, ..., R_n\}$.
- A set of *Information Types*, denoted $F = \{F_1, F_2, ...\}$, representing the types of input and output of a schema.
- *Environmental Sensors*, denoted $ES = \{ES_1, ES_2, ...\}$.
  - The input to ES_i is a specific physical sensor signal.
  - The output from ES_i is denoted as $O^{ES_i} \subset F$. We assume each environmental sensor only has one output, although the output may represent a set of features of the sensory data (e.g., range and intensity). The output of an environmental sensor is connected to the input of a perceptual schema.
- *Perceptual Schemas*, denoted $PS = \{PS_1, PS_2, ...\}$.
  - The inputs to PS_i are denoted $\cup I_k^{PS_i} \subset F$, for k going from 1 to the number of inputs to the perceptual schema. The perceptual schema inputs can come from either the outputs of communication schemas or environmental sensors.
  - The output from PS_i is denoted $O^{PS_i} \subset F$. We assume each perceptual schema only has one output with a set of features (e.g., distance and angle). The output from a perceptual schema can go to the inputs of either communication schemas or motor schemas.
- *Communication Schemas*, denoted $CS = \{CS_1, CS_2, ...\}$.
  - The inputs to CS_i are denoted $\cup I_k^{CS_i} \subset F$, for k going from 1 to the number of inputs to the communication schema. The inputs come from the outputs of perceptual schemas or communication schemas.
  - The output from CS_i is denoted $O^{CS_i} \subset F$. We assume each communication schema only has one output with a set of features. The output can go to the inputs of perceptual schemas, motor schemas or other communication schemas.
- *Motor Schemas*, denoted $MS = \{MS_1, MS_2, ...\}$.
  - The inputs to MS_i are denoted $\cup I_k^{MS_i} \subset F$, for k going from 1 to the number of inputs to the motor schema. These inputs always come from the outputs of perceptual schemas or communication schemas.
  - The output from MS_i is denoted $O^{MS_i} \subset F$. We assume each motor schema has only one output with a set of features (e.g., velocity). The output always goes to the robot effector control process.
- *Connection Rules*, denoted $\exists_{i,j,k} CONNECT(O^{S_i}, I_k^{S_j}) \Leftrightarrow O^{S_i} = I_k^{S_j}$, where S_i and S_j are types of schemas. This notation means that the output of S_i can be connected to one of the inputs of S_j, if and only if they have the same information type.
- *Utility*, denoted $\mu(i) = \sum_j (w \times 1/C_j + (1-w) \times P_j)$, where i represents the *ith* robot in the team, and j

represents the *jth* sensori-computational system that the robot needs to use. The utility $\mu(i)$ measures the fitness of the solution on robot R_i. We also have:
  - *Sensori-Computational Systems*, denoted $SCS = \{SCS_1, SCS_2, ...\}$, where SCS_k is a module that computes a function of its inputs and its current pose or position [5]. SCS_k is composed of a specific sensor ES_k and its computational unit.
  - *Success Probabilities*, denoted $P = \{P_1, P_2, ...\}$, where $0 \leq P_k \leq 1$, and $P_k = probability(SCS_k)$.
  - *Sensing Costs*, denoted $C = \{C_1, C_2, ...\}$, where $C_k = cost(ES_k)$, the sensing cost of a sensor ES_k.
  - *Weight w*, which combines probability and cost.
- A task is described as a set of *Motor Schemas* $T = \{MS_1, MS_2, ...\}$ along with some application-specific, user defined parameters, such as the goal position and the pushing direction[1].
- A solution is to organize a team into subteams, such that each subteam can contribute to the goal. There exists a solution if and only if for all $MS_i \in T$, the inputs of MS_i are satisfied, along with all the inputs of the schemas that feed into MS_i. More formally, the connections for subteam t must satisfy the following constraints:
  - $\forall_{MS_i \in T} \forall_{j,k}, \exists CONNECT(O^{S_j}, I_k^{MS_i})$. The inputs of MS_i are satisfied.
  - $\forall_{S_j} \forall_{m,k}, \exists CONNECT(O^{S_m}, I_k^{S_j})$. The inputs of S_j are satisfied. This process continues until there is a series of connections that connect $I_k^{S_q}$ to O^{S_m}.
  - $\forall_{S_q} \forall_{p,k}, \exists CONNECT(O^{S_p}, I_k^{S_q})$. The inputs of S_q are satisfied.
  - $\forall_{S_p} \forall_{n,k}, \exists CONNECT(O^{ES_n}, I_k^{S_p})$. The inputs of S_p are satisfied by some ES(s).
  - Where S_j, S_m or $S_q \in PS \bigcup CS, S_p \in PS$, and $ES_n \in ES$.
  - The utility $\sum_j \mu(j))$ for every R_j in subteam t is maximized.

The problem is to organize the robots into subteams such that robots within each subteam cooperate with each other to contribute to the accomplishment of the team-level task *T* with a maximum utility.

III. The ASyMTRe Approach

A. Information Representation

We represent three types of information in the ASyMTRe system – the information of robot team capabilities, the flow of information required into and out of schemas, and the sensing costs and success probabilities of all sensori-computational systems. Robot team capabilities describe each robot and its sensor resources in the format (`robotID, ES`$_1$`, ES`$_2$`, ...`). The sensing costs and success probabilities are provided to estimate the fitness of a solution among other possible solutions. Most importantly is the flow of information, which is a set of information types *F* that constitutes the input and output of the available schemas. This information flow allows us to distribute the perceptual input or the calculation of effector output across multiple robots. With the introduction of communication schema, information can flow within and across robot team members, which increases the number of possible solutions to the problem.

[1]In future work, we will develop a more general task specification, similar to the formal specification of tasks in [7]

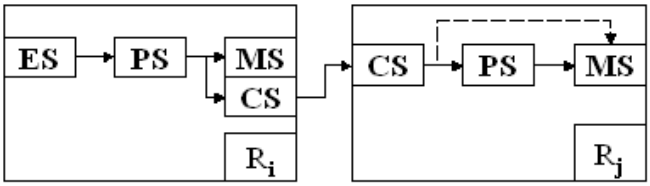

Fig. 1. Possible connections of schemas between two robots.

B. Reasoning Process

Our reasoning process enables robot team members to autonomously connect the inputs and outputs of their available schemas to result in dynamic task solution strategies that are a function of the current team's capabilities. Given the information representation of robot team capabilities and the flow of information, the collaborative reasoning among robots must generate such a mapping that leads to all required connections being made for each robot to accomplish its task.

None of the schemas are connected initially and connections can be made within or across robots depending on the required information flow. A connection within a robot can be built between two schemas if and only if the input information type of one schema is exactly the output information type of another schema, and these two schemas can be implemented on one robot. Figure 1 shows some possible connections between two robots. According to the connection rules, the general task solutions are converted to combinations of various schemas, providing different ways to accomplish the task so that each robot can determine its possible solution.

C. The Configuration Algorithm

The core of the ASyMTRe configuration algorithm shown in Table I is a greedy search algorithm which first handles robots with fewer sensor resources (less capable robots). Each robot is assigned a priority at the beginning of a task according to its sensing capability. Less capable robots have higher priorities to configure their solutions, since these robots will likely have fewer solutions for success. When a robot does not have the required sensor to fulfill a task, the algorithm will find a robot with the least sensing capability and maximal utility to provide the needed information. Therefore, robots with more sensor resources are saved for future configuration, since they likely can be helpful in many different ways.

At present, this approach clearly utilizes a centralized reasoning system. We began with a centralized reasoner to illustrate the capabilities of the software reconfiguration approach that is based on varying the connections of environmental, perceptual, and motor schemas within and across multiple robots.

TABLE I

THE CORE OF THE ASYMTRE CONFIGURATION ALGORITHM

Reason(R, T, U)
(R, T, U): the robot team composition, task, and utility
n: the number of robots in the team
m: the number of configurations to accomplish the task
k: a constant, which specifies the number of iterations

1) Sort the robot team members according to increasing sensing capabilities. $[O(nlog(n))]$
2) Generate a list of potential combinations of schemas of size *m* that can accomplish the task. $[O(1)]$
3) Configure solutions on the robot team. $[O(kmn^2)]$
   - For each robot R_i in the sorted order: $[O(n)]$
     - For each combination j to accomplish the task $[O(m)]$
       - If R_i can accomplish the task by itself, assign solution j to R_i. $[O(1)]$
       - Else check the other $n-1$ robots to see if one can provide the needed information. $[O(n)]$
       - If the estimated utility of R_i executing solution j is greater than the utility of its previous solution, update the solution strategy on R_i. $[O(1)]$
4) Continue the above process until: $[O(kmn^2)]$
   - All the robots in the team can accomplish the task.
   - Or, after k number of trials.
5) If a solution exists, report the solution; otherwise, report "Failure".

* A complete version of the ASyMTRe algorithm can be found in [16].

The level of redundancy and the maintenance of a centralized database would be the main concern of the centralized system. Thus, our ongoing work focuses on distributing the reasoning capability across multiple robots.

An important feature of the algorithm is its relatively low complexity bound. The computational complexity of the algorithm is $O(kmn^2)$, where n is the number of robots, m is the number of potential solutions to accomplish the task, and k is the number of iterations. Since *k* and *m* are relatively small, the complexity is primarily determined by n. In our experiments, we have varied the value of n from 2 to 100, and the running time is always less than one second on desktop computers. With this performance, we can duplicate the reasoning system on every robot to increase the robustness, and can also easily replan the solution when required. A more formal analysis of this approach is given in [16], showing the soundness, completeness, and optimality of ASyMTRe.

IV. EXPERIMENTS

We have claimed that the ASyMTRe approach can enable the robot team to find a task solution automatically based on the flow of information in the system. To validate this approach, we designed two experiments: multi-robot transportation and box pushing. These two experiments present the characteristics of the robot cooperation task to which the reasoning system is applicable: *altruistic* cooperation and *coalescent* cooperation. In altruistic cooperation, robots cooperate only when a less capable robot needs help from a more capable robot (helper) to accomplish the task. The helping behavior does no good to the helper, but it is beneficial to the entire team. In coalescent cooperation, robots cooperate when the task requires them to work together to achieve the goal. It is beneficial to the robot itself and to the entire robot team. In the following sections, we describe the two experiments in detail, as well as the results of these studies.

A. Multi-Robot Transportation

1) Task description: In this application, a team of robots must navigate from their starting positions to a set of goal positions (one per robot) defined in a global coordinate reference frame. Assume that all the robots are programmed with the motor schema *go-to-goal*, which moves the robot from its current position to a goal position, defined in a global coordinate reference frame. To successfully use this motor schema, a robot must be able to perceive its own current position relative to its goal. If every robot team member can localize itself, obviously the solution is to have every robot navigate independently. However, on some teams, there might be robots that do not have the sensing and behavior capabilities to localize (e.g., see [13]); they need help from more capable robots to provide the information needed to fulfill the task. In this application, robots exhibit altruistic cooperation for the benefit of the entire team. In the following paragraphs, we detail the application by introducing the environmental sensors and various schema used in this application.

The environmental sensors are: laser scanner with an environmental map (`laser`), omnidirectional camera (`camera`), `DGPS`, and communication sensor (`comm`). The functionalities of the perceptual schemas, communication schemas and motor schemas are defined in Table II. We assume:

- A robot with a `laser` and an environmental map can estimate its current global position in the environment.
- A robot with a `DGPS` can estimate its current global position in the environment.
- A robot with a `camera` or `laser` can estimate the relative position of another robot in the environment, as long as the other robot is within its sensing range.
- A robot has the computational ability to convert a relative position to a global position.

From the description of the task, we define a set of information $F = \{$*Self_Global_Position*, *Other_Global_Position*$\}$. Table II shows the input and output information for each schema used in this application. According to the flow of information, the configuration algorithm generates all the possible connections that can connect the available schemas and lead the robot to achieve its goal. Assuming that all the robots have communication capabilities, two specific connections are shown in Figures 2 and 3. Initially, there are no connections between schemas. After the solution is generated, proper sensors are activated, and the schemas are connected, based on the connection rules in Section II. As an example, in Figure 2, the output of PS_1 is connected to the input of MS_1 because we have $O^{PS_1} = I_1^{MS_1} =$ *Self_Global_Position*.

TABLE II

PERCEPTUAL SCHEMAS, COMMUNICATION SCHEMAS, AND MOTOR SCHEMAS IN MULTI-ROBOT TRANSPORTATION TASK

Schema	Description	Input	Output
PS_1	Calculates self global position	Laser or DGPS	Self_Global_Position
PS_2	Calculates self goal position	Hardcoded	Goal_Position
PS_3	Calculates global position of another robot	Laser or camera and Self_Global_Position	Other_Global_Position
PS_4	Calculates self global position according to the detected relative position of another robot and the global position of the same robot	Camera and Other_Global_Position	Self_Global_Position
CS_1	Communicates self global position to another robot	Self_Global_Position	Other_Global_Position
CS_2	Communicates global position of another robot to that robot	Other_Global_Position	Self_Global_Position
CS_3	Receives global position of another robot	Other_Global_Position	Other_Global_Position
CS_4	Receives global position of itself from another robot	Self_Global_Position	Self_Global_Position
MS_1	Go-to-goal schema that calculates motor command that leads the robot towards the goal.	Self_Global_Position and Goal_Position	Motor commands

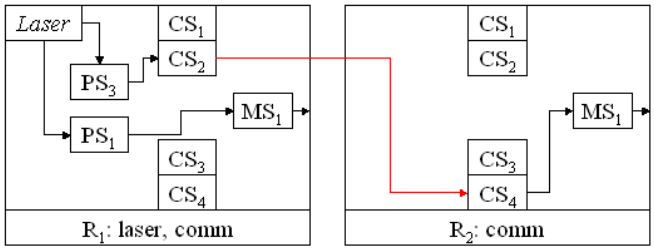

Fig. 2. One solution for connecting the schemas to accomplish the "navigate" goal. This solution involves R_1 using its `laser` to globally localize itself and to calculate a relative position of R_2. With this information, R_1 can calculate the global position of R_2 and communicate this information to R_2, for its use in moving to its goal position.

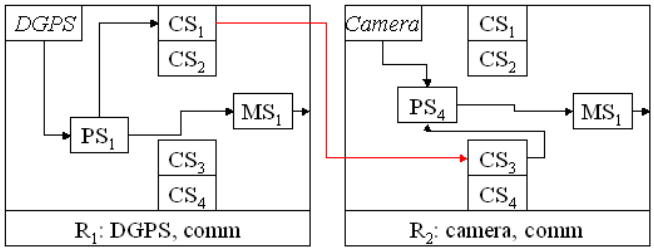

Fig. 3. A second solution for connecting the schemas to accomplish the "navigate" goal. This solution involves R_1 using its `DGPS` to globally localize itself, and then communicating this information to R_2. R_2 uses `camera` to calculate the relative position of R_1, and then combines this with R_1's communicated global position to determine its global position.

2) The reasoning process: After searching through all possible connections of schemas, the system generates several methods to accomplish the navigation task:

1) A robot R_i can navigate using `laser` or `DGPS`.
2) A robot R_j can navigate if there is another robot R_i that can navigate and estimate the global position of R_j using `laser` or `camera`.
3) A robot R_j can navigate if there is another robot R_i that can navigate, and R_j can use `camera` to estimate R_i's relative position and calculate its own global position.

With multiple solutions available, a robot needs to determine which solution it should use. This is decided by each robot's current sensing capability and the estimated utility of the particular solution it chooses. We would like each robot to select a solution that is the most efficient, or least costly. The utility is calculated by the combination of sensing costs and success probabilities. Table III lists the values of sensing costs and success probabilities in our application. Here, we have provided fuzzy estimates; in most application, these costs and probabilities will be specific numeric values. The sensing cost is determined by the sensory and computational requirements of the solution. Perceptual processes with a significant amount of sensor processing, such as laser scan matching or image processing, are given higher sensing costs. Perceptual processes with a relatively low processing requirement, such as DGPS, are assigned lower sensing costs. Success probability is an estimated value based upon experiences. Perceptual processes that are easily influenced by environmental factors, such as image processing under different lighting conditions, are given lower success probabilities. Otherwise, they are given higher success probabilities.

TABLE III

SENSORI-COMPUTATIONAL SYSTEMS, SENSING COSTS, AND SUCCESS PROBABILITY

SCS	Sensing Cost	Success Probability
PS_1(Laser)	Sensing_HIGH	Succ_MED
PS_1(DGPS)	Sensing_LOW	Succ_MED
PS_3(Laser)	Sensing_HIGH	Succ_MED
PS_3(Camera)	Sensing_MED	Succ_LOW
PS_4(Laser)	Sensing_HIGH	Succ_MED
PS_4(Camera)	Sensing_MED	Succ_LOW
CS_i(Comm)	Sensing_LOW	Succ_HIGH

B. Box Pushing

1) Task description: Box pushing was studied by Donald, *et al.* [5] in the development of information invariants for analyzing the complexity of alternative cooperation algorithms. To illustrate the connection of our approach to the theory of information invariants, we have defined our box pushing experimentation in a similar manner to [5]. In our box pushing example, a team of robots is brought together to push boxes. The goal is to organize the team into subteams, such that each subteam is able to push a box with exactly two robots, which we call pusher robots. Assume that all the robots

TABLE IV

PERCEPTUAL SCHEMAS, COMMUNICATION SCHEMAS, AND MOTOR SCHEMAS IN MULTI-ROBOT BOX PUSHING

Schema	Description	Input	Output
PS_1	Computes the force when pushing a box	Bumper	force
PS_2	Computes the relative displacement when pushing a box	Odometry	displacement
PS_3	Computes the angle between the line of pushing and the actual moving direction	Gripper	angle
PS_4	Computes the vector of the box relative to the pusher robot	Laser or Camera	box_relative_vector
CS_1	Communicates information from one robot to another	F_i	F_i
MS_1	The Push schema which computes the motor commands needed for the robot to push a box along a line	force, displacement, angle, or box_relative_vector	Motor commands
MS_2	Move along with the box, so that the helper robot can keep track of the box and the pusher robots.	box_relative_vector	Motor commands

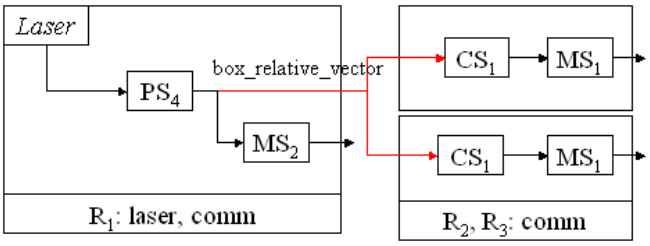

Fig. 4. One solution for connecting the schemas to accomplish the "Push" goal. This solution involves R_1 using its laser to perceive the relative vector of the box, and then communicating this information to the pusher robots. The pusher robots R_2 and R_3 can then take actions.

are programmed with the motor schema *Push*, which pushes the box along a straight line. To use this motor schema, a robot must be able to perceive the box's vector relative to itself. Here, the relative vector includes the applied force, relative displacements, the angle between the actual pushing direction and the line of pushing. For example, the two pusher robots can record the relative displacements while they are pushing the box, and by comparing these two values, they can decide which robot should push harder. Suppose that the environmental sensors are: `bumper`, `odometry`, `gripper`, `comm`, `laser`, and `camera`. Three methods were presented in [5] to push a box with two robots. In addition, we generate another method here where the pusher robots do not have the capabilities to perceive the relative vector of the box, while a helper robot helps them get this information. The box pushing application exhibits both coalescent and altruistic cooperation. The cooperative box pushing methods from [5] are as follows:

1) Using `bumper`, two robots can compute their applied forces and communicate this information to each other, allowing them to decide which robot should push harder.
2) Using `odometry`, two robots can compute the relative displacements along the line of pushing; they exchange the location information and take actions to reduce the difference between the relative displacements.
3) Using special `grippers`, two robots can compute the angles between the line of pushing and its actual moving direction; they take actions to reduce its angle on the next step – no explicit communication is needed in this case.
4) Using a `laser` or `camera`, a helper robot can help the pusher robots calculate the relative vector of the box, enabling the pusher robots to take appropriate actions.

The information set in the box pushing application is $F = \{$*force, displacement, angle, box_relative_vector*$\}$. Various schemas and their input and output information are defined in Table IV. The reasoning process is similar to the multi-robot transportation application, which generates software schema connections based on the connection rules. Figure 4 shows one of the connections that enables the robots to push a box. The result is the decomposition of the team such that each subteam can accomplish the task. To calculate the utility, we assume that the sensing costs and success probabilities for {`gripper, bumper, odometry`} are the same. The only difference is `laser` and `camera`, in which the values are the same as the multi-robot transportation application.

V. RESULTS AND DISCUSSION

With the above setup of the two applications, we now present the results of the ASyMTRe approach to autonomous software reconfiguration. To validate the reasoning system, robots with different sensing capabilities are brought together to form a team. The experiments focus on two aspects of the team: the number of robots *n* and the heterogeneity of robots. In our experiments, *n* varied from 2 to 100. The total reasoning time is always less than one second. To measure the diversity of the robot team, we calculate *H*, the simple social entropy metric of the team [2]. The value of *H* measures the diversity of the robot team and depends on the number of homogeneous subteams it contains and the proportion of robots in each subteam. This measure is proportional to the diversity of the team. For the sensing capability of the robots, we assume every robot has the ability to communicate.

A. Multi-Robot Transportation

In this application, we can choose from {`gps, laser, camera`} to build a robot, which provides up to 2^3 choices of different robot capabilities. The different types of robots are shown in Table V. Among them, a robot of type 1 or 4 cannot localize by itself since it has neither `laser` nor `DGPS`. We present three cases with different team capabilities.

Case 1, n = 2, H = 1.0. We first describe a simple case, where the robot team is composed of only two robots from type 3 and 4 respectively. The solution generated is obvious: the robot with `laser` can navigate by itself, while it helps the

TABLE V

EIGHT TYPES OF ROBOT WITH DIFFERENT SENSING CAPABILITIES

Type	Available Sensor(s)
1	`comm`
2	`DGPS, comm`
3	`laser, comm`
4	`camera, comm`
5	`DGPS, laser, comm`
6	`DGPS, camera, comm`
7	`laser, camera, comm`
8	`DGPS, laser, camera, comm`

Fig. 5. Case 1: The initial setup of the two pioneers robots is on the left. The result when the two robots reach their goal points is shown on the right. Pictures are in courtesy of Chandra [4].

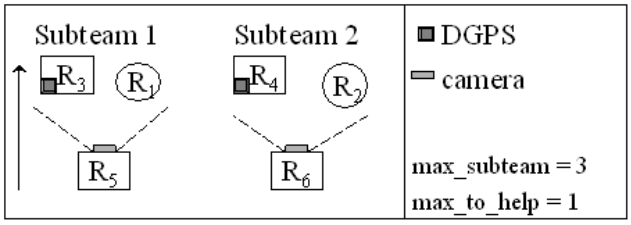

Fig. 6. Results of ASyMTRe applied to Case 2, in which the team is composed of six robots from three different types. The simple social entropy is 1.58, the maximum value is 2.58. According to the automated software reconfiguration process, the team is divided into two subteams to accomplish the transportation task.

TABLE VI

BOX PUSHING: ROBOT TEAM COMPOSITION

Robot	Environmental Sensor(s)
R_1	`comm`
R_2	`bumper, comm`
R_3	`odometry, comm`
R_4	`gripper, comm`
R_5	`laser, camera, comm`
R_6, R_7	`bumper, gripper, odometry, comm`

robot with `camera` navigate by method 2 or 3 (see Section IV-A.2), depending on the weight factor. This particular example has been successfully implemented on two Pioneer robots by Chandra in [4]. Her control algorithm reads in the output from the configuration algorithm and activates the proper sensors and schemas to accomplish the navigation task, according to the described ASyMTRe approach. Part of the navigation process is shown in Figure 5.

Case 2, n = 6, H = 1.58. In this case, the robot team is composed of six robots, with two each of types 1, 2, and 4. If the maximum number of robots allowed in a subteam is 3 and the maximum number of robots that any robot can help is 1, the solution is shown in Figure 6. The team is divided into two subteams, each of which has three robots from each type. The team is moving forward along the direction of the arrow. There are physical constraints, such as the camera's field of view. At the beginning, we need to manually place the robots into proper positions so that the constraints are satisfied. (In future work, we plan to build in automated mechanisms to handle maintaining appropriate fields of view.) For instance, R_3 and R_1 must be in the field of view of R_5. In subteam 1, R_3 with a `DGPS` localizes by itself using method 1, and it helps R_5 with a `camera` localize using method 3. After R_5 has been helped, it helps R_1 localize using method 2. A similar situation occurs with subteam 2.

Case 3, n = 99, H = 1.58. To test the adaptability of the reasoning system, we applied it to a robot team of size 99. To make the setup easier, we still use the three types of robots from type 1, 2, 4, respectively, but we duplicate them 33 times. For the parameters, we have the same setup as in case 2, where the size of the subteam is 3. The reasoning system generates the result in less than one second. It divides the team into 33 subteams, and each with three different types. The result is the same as is shown in Figure 6, except that it has 33 subteams.

B. Box Pushing

In this application, if every box needs exactly two pusher robots, the maximum subteam size is 3 (there might be one helper). The environmental sensors are {`laser, camera, gripper, bumper, odometry`}. We have 2^5 choices of different robot capabilities.

To demonstrate how the box pushing task can be achieved, we designed the following robot team. As shown in Table VI, the team is composed of 7 robots from 6 different types, the simple social entropy is 2.52. Figure 7 presents the results, in which the team is divided into three subteams. In subteam 1, robot R_1 and R_4 push the box together. Since R_1 cannot perceive the box's relative vector by itself, they are helped by R_5 using either `laser` or `camera`. In subteam 2, robot R_3 and R_6 push the box together using method 2, since they both have `odometry`. In subteam 3, robot R_2 and R_7 push the box together using method 1, since they both have `bumper`. The team has been divided into several subteams and each subteam can fulfill the box pushing task. We notice that in subteam 1, R_5 can use either `laser` or `camera` to help the pusher robots. The correct choice depends on its utility of using a certain sensor. In this example, if we focus on sensing cost, `camera` is a better choice. Otherwise, if we focus on the success probability, `laser` is a better choice.

VI. RELATED WORK

Research specific to heterogeneous robots often focuses on the issue of *task allocation*, which is the problem of determining a suitable mapping between robots and tasks. Several approaches have been developed in the last decade. Since it has been shown that developing the optimal mapping of tasks to robots is NP-hard [11], existing mechanisms

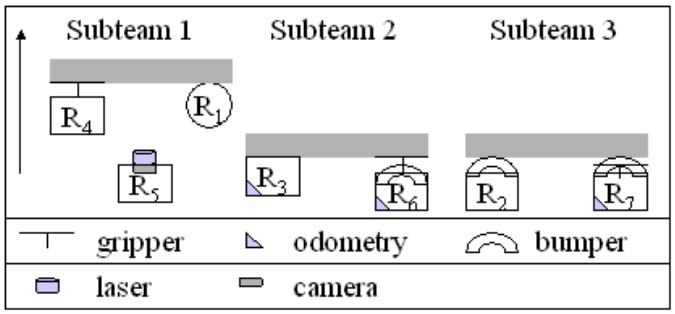

Fig. 7. Results of applying ASyMTRe to an instance of the box pushing task, in which the team is composed of seven robots from six different types. The simple social entropy is 2.52, and the maximum value is 2.81. The team is autonomously divided into three subteams, and each heterogeneous subteam can successfully push a box.

for multi-robot task allocation use some type of heuristic greedy strategy to achieve the mapping. There are behavior-based approaches (e.g., [10], [17]), auction-based approaches (e.g., [3], [8]), and First-price auctions (e.g., [18]). In all of these approaches, each robot is given the utility measure for it to perform a particular task. The various task allocation approaches then provide distributed decision-making processes that use these utility measures to derive an allocation of tasks to robots. Another approach is for the human designer to subdivide the application into roles that define action strategies for achieving part of the application. Robot team members then determine autonomously which robots should perform which roles. Collaborative architectures provide general mechanisms to coordinate the robots depending upon the requirements of the roles they are fulfilling and the types of collaboration that are necessary (e.g., [15]). In other related work, research in multi-agent coalition formation [14] is related to the approach of ASyMTRe, in that the objective is to generate coalitions of agents needed to accomplish joint intentions. However, these approaches do not address the autonomous synthesis of cooperative behaviors at the low level of environmental, perceptual, and motor schemas. The automatic generation of these cooperative task solutions is needed since required low-level cooperative control approaches are dependent upon the combination and distribution of sensors across the robot team members. Our representation of the robot capabilities is essentially the same as in STRIPS planning [6], although we abstract the problem in a very different manner. In common STRIPS planning, the robot capabilities are usually represented by predicates such as *Has(laser)* and *Action(Goto)*, which requires solution strategies to be previously incorporated into the STRIPS rules. We represent the capabilities at the level of schema, which allows the planner to be independent of precompiled solution strategies, enabling ASyMTRe to generate *how* to solve tasks in a much more general manner.

VII. Conclusions and Future Work

This paper has presented ASyMTRe – a mechanism for the automatic generation of task solution for heterogeneous robot teams. Built upon schema and information invariants theories, our approach enables the robot team to dynamically connect schemas within and across robots to accomplish a task. We have successfully demonstrated the solution capability of the robot team in many scenarios. Two specific applications were presented to validate our algorithms.

A more formalized ASyMTRe approach and its performance analysis can be found at [16]. Our ongoing work is to implement ASyMTRe as a distributed reasoning system and compare it with the current centralized reasoning system. Additionally, we plan to consider motion constraints in the reasoning process to facilitate physical implementation.

References

[1] R. C. Arkin, T. Balch, and E. Nitz. Communication of behavioral state in multi-agent retrieval tasks. In *Proceedings of the International Conference on Robotics and Automation*, pages 588–594, 1993.

[2] T. Balch. Hierarchic social entropy: An information theoretic measure of robot team diversity. *Autonomous Robots*, 8(3):209–238, 2000.

[3] S. Botelho and R. Alami. M+: A schema for multi-robot cooperation through negotiated task allocation and achievement. In *Proceedings of IEEE International Conference on Robotics and Automation*, pages 1234–1239, 1999.

[4] M. Chandra. Software reconfigurability for heterogeneous robot cooperation. In *MS Thesis of DILab*. Computer Science Department at the University of Tennessee, May 2004.

[5] B. R. Donald, J. Jennings, and D. Rus. Information invariants for distributed manipulation. *International Journal of Robotics Research*, 16(5):673–702, 1997.

[6] R. E. Fikes and N. J. Nilsson. STRIPS: A new approach to the application of theorem proving to problem solving. *Artificial Intelligence*, 2:189–208, 1971.

[7] C. H. Fua, S. S. Ge, and K. W. Lim. Boas: Backoff adaptive scheme for task allocation with fault tolerance and uncertainty management. In *Proceedings of the IEEE Intl. Symposium on Intelligent Control*, Taipei, Taiwan, September 2004.

[8] B. Gerkey and M. J. Mataric. Sold! auction methods for multi-robot coordination. *IEEE Transactions on Robotics and Automation*, 18(5):758–768, 2002.

[9] R. R. Murphy. In *Introduction to AI Robotics*. MIT Press, Cambridge, MA, 2000.

[10] L. E. Parker. ALLIANCE: An architecture for fault-tolerant multi-robot cooperation. *IEEE Transactions on Robotics and Automation*, 14(2):220–240, 1998.

[11] L. E. Parker. Toward the automated synthesis of cooperative mobile robot teams. In *Proceedings of SPIE Sensor Fusion and Decentralized Control in Robotic Systems II*, volume 3525, pages 82–93, 1998.

[12] L. E. Parker. The effect of heterogeneity in teams of 100+ mobile robots. In A. Schultz, L. E. Parker, and F. Schneider, editors, *Multi-Robot Systems Volume II: From Swarms to Intelligent Automata*. Kluwer, 2003.

[13] L. E. Parker, B. Kannan, F. Tang, and M. Bailey. Tightly-coupled navigation assistance in heterogeneous multi-robot teams. In *Proceedings of the IEEE International Conference on Intelligent Robots and Systems*, 2004.

[14] O. Shehory. Methods for task allocation via agent coalition formation. *Artificial Intelligence*, 101(1-2):165–200, 1998.

[15] R. Simmons, S. Singh, D. Hershberger, J. Ramos, and T. Smith. First results in the coordination of heterogeneous robots for large-scale assembly. In *Proc. of the Seventh International Symposium on Experimental Robotics*, 2000.

[16] F. Tang and L. E. Parker. Coalescent multi-robot teaming through ASyMTRe: a formal analysis. In *submitted*, 2004.

[17] B. B. Werger and M. J. Mataric. Broadcast of local eligibility for multi-target observation. *Distributed Autonomous Robotic Systems 4*, pages 347–356, 2000.

[18] R. Zlot, A. Stentz, M. B. Dias, and S. Thayer. Multi-robot exploration controlled by a market economy. In *Proceedings of IEEE International Conference on Robotics and Automation*, pages 3016–3023, 2002.