Automatic Synthesis of Multi-Agent Motion Tasks Based on LTL Specifications

Savvas G. Loizou Kostas J. Kyriakopoulos

Control Systems Laboratory, Mechanical Eng. Dept
National Technical University of Athens, Greece
{sloizou, kkyria}@central.ntua.gr

***Abstract*— In this paper we propose a methodology for automatically synthesizing motion task controllers based on Linear Temporal Logic (LTL) specifications. The proposed design of the underlying multi-agent controllers possess a special structure that allows for implicit satisfaction of basic liveness and safety specifications. The resulting closed loop system is of hybrid nature combining the continuous dynamics of the underlying system with the automatically synthesized switching logic that enforces the LTL specification. The effectiveness of the proposed scheme is verified through non-trivial computer simulations.**

I. Introduction

Automatic controller synthesis has recently gained increasing attention from system and control theorists, mainly due to the need for an automated methodology to build controllers satisfying a desired complex behavior. Formal specification of the desired complex behavior is a key issue in this effort. Our main motivation comes from the field of micro robotics, where a team of micro-robotic agents must cooperate to perform various tasks. In [1] the author proposes a methodology to automatically synthesize local controllers achieving a formation with specifications given in terms of graphs. In [2] the authors consider automatic controller synthesis based on Linear Temporal Logic specifications for linear systems.

Several motion description languages have been proposed for specifying motion tasks. In [3] the use of motion description languages for multi-modal control is described and in [4] a general framework is presented for using an extended motion description language in motion control. The reason we chose LTL for describing the behavior of the system, is that it provides a formal specification mechanism with the capability to define desired behaviors quantitatively and due to it's similarity to natural languages, it provides an intuitive and succinct way to express complex behaviors.

Formally, we consider a multi-agent motion task as consisting of one primary objective, like "navigate the agents to the specified positions" , and a number of secondary objectives, like "agent i tracks agent j until agents j,k,l form a specific formation", that are carried out during the execution of the primary objective.

The authors want to acknowledge the contribution of the European Commission through contract IST-2001-33567-MICRON and IST-2001-32460-HYBRIDGE

Our approach is to have a global convergent controller that handles the primary motion task and a set of controllers that lie in it's range of convergence, as intuitively shown in figure 1. Since the range of convergence of the secondary controllers is a subset of the range of the primary, the system is guaranteed to satisfy basic safety specifications, i.e. the robots will never collide and moreover basic liveness specifications, i.e. the robots will eventually converge to the configuration dictated by the global controller.

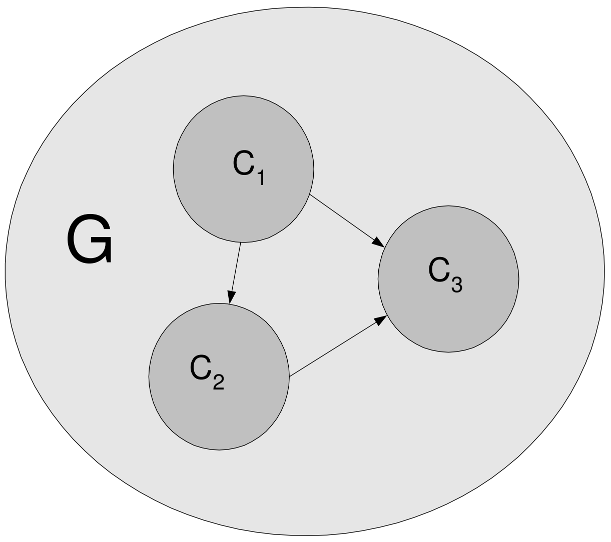

Fig. 1. Secondary controllers C_i in the range of convergence of controller G

The rest of the paper is organized as follows: Section II presents the LTL framework used for specifying motion tasks. Section III discusses a way extract controller sequences from Buchi automata corresponding to LTL specifications. Section IV presents a motion controller suitable to handle the synthesized task and outlines the general system architecture. Section V presents simulation results and the paper concludes with section VI.

II. Using LTL to specify Motion Tasks

Linear Temporal logic is an extension of propositional logic suitable for reasoning about infinite sequences of states. We use a fragment of LTL, equipped with the usual propositional connectives $(\wedge, \vee, \neg)$ extended with two temporal operators: "$\bigcirc$" (next) and "$\mathcal{U}$" (unless). The formulas of the logic are built from atomic propositions using propositional connectives and temporal operators.The sequences considered are isomorphic to natural numbers and each state is a propositional interpretation. Purely propositional formulas are interpreted in a single state and the temporal operators indicate in which state of a sequence their arguments must be evaluated. We denote by P the set of atomic propositions and recursively define the well formed formulas (wff) as follows:

- **true**, **false**, p, $\neg p$ are wff for all $p \in P$;
- if φ_1 and φ_2 are wff, then $\varphi_1 \wedge \varphi_2$ and $\varphi_1 \vee \varphi_2$ are wff;
- if φ_1 and φ_2 are wff, then $\bigcirc\varphi_1$, and $\varphi_1 \mathcal{U} \varphi_2$ are wff formulas;

Implication $\varphi_1 \Rightarrow \varphi_2$ is defined as the abbreviation of $\neg\varphi_1 \vee \varphi_2$. From the unless operator, another commonly used operator can be defined:

$$\Box\varphi = \varphi \mathcal{U} \mathbf{false}$$

which is read "always" and requires that its arguments be true at all future points. Wff are interpreted over sequences of states $\sigma : \mathbb{N} \rightarrow 2^P$. For a sequence σ, $\sigma(i)$ denotes the i'th state, $\sigma[i]$ represents the prefix of σ obtained from the first i elements of σ and σ^i represents the suffix of σ obtained by removing the i first states, i.e. $\sigma^i(j) = \sigma(i+j)$. The truth value of a formula on a sequence σ, is taken to be the truth value obtained by starting the interpretation of the formula in the first state of the sequence, and is given by the following rules:

For any $p \in P$, wff formulas φ_1, φ_2 and $i \in \mathbb{N}$:

- For all σ, we have $\sigma \models \mathbf{true}$ and $\sigma \not\models \mathbf{false}$
- $\sigma \models p$ iff $p \in \sigma(0)$
- $\sigma \models \neg p$ iff $p \in \sigma(0)$
- $\sigma \models \varphi_1 \wedge \varphi_2$ iff $\sigma \models \varphi_1$ and $\sigma \models \varphi_2$
- $\sigma \models \varphi_1 \vee \varphi_2$ iff $\sigma \models \varphi_1$ or $\sigma \models \varphi_2$
- $\sigma \models \bigcirc\varphi_1$ iff $\sigma^1 \models \varphi_1$
- $\sigma \models \varphi_1 \mathcal{U} \varphi_2$ iff either $\sigma \models \varphi_1$ or $\sigma \models \varphi_2$ or $\exists i > 0$ such that $\sigma[i] \models \varphi_1$ and $\sigma^i \models \varphi_2$

Let us consider a simple example to illustrate the use of LTL. This example attempts to model a fragment of a possible behavior we expect from a micro-robotic multi-agent system during a biological cell transportation task. Assume robot 1 is carrying the cell and robot 2 is monitoring the procedure by tracking robot 1. If a fault occurs to robot 2 the system must slow down or stop until robot 3 replaces robot 2 and then continue execution of the task. Let predicate G be active when a state feedback global controller taking the robots from an initial to a final configuration avoiding collisions and deadlocks (existence of such controller is discussed in section IV) is active. Let T_{21} and T_{31} predicates be active when state feedback controllers for robot 2 and 3 respectively to track robot 1 are active and St_1 predicate be active when a stalling controller for robot 1 is active. Moreover let predicate E_2 be active when an error occurs to robot 2 and rT_{31} be active when robot 3 assumes it's tracking position. The specification can now be defined as:

$$\Box G \wedge (T_{21} \mathcal{U} (E \wedge \bigcirc (St_1 \mathcal{U} (rT_{31} \wedge \bigcirc\Box T_{31}))))$$

We define the set of predicates P to be the disjoint union:

$$P = \mathcal{O} \cup^* \mathcal{C} \cup^* \{G\}$$

of the set of observation predicates $\mathcal{O}$ with controller predicates $\mathcal{C}$ and the global controller predicate $\{G\}$. We assume that observation predicates and the global controller predicate are uncontrollable events, but controller predicates are controllable events, in the sense that we can turn on and off the corresponding controllers. This has to do with the fact that we consider a controller predicate to be active if and only if a control law associated with it,is active, i.e. we consider that robot 1 being tracked by robot 2 is actually equivalent to controller T_{31} (which is a tracking controller) being active. The global controller predicate must be active throughout the task, hence every formula φ should be equivalent to $\Box G \wedge \varphi$. This will guarantee basic liveness and safety specifications, i.e. the multiagent team will eventually reach the specified target configuration and they will never collide or get deadlocked. The previously mentioned assumptions allows us to design (hybrid) controllers correct by construction, avoiding the need for formal verification.

III. From LTL specifications to Buchi automata

There are established results about converting LTL specifications to Buchi automata. Given an LTL formula φ, it is possible to construct a Buchi automaton A_φ, accepting every string satisfying formula φ. This fact was first shown by Buchi [5]. The resulting automata are in the worst case , exponential in the length of the translated formula. The interested reader is referred to [6] for a detailed description of the translation process.

A Buchi automaton that resulted from the translation of the LTL formula φ, is a tuple $A_\varphi = \{\Sigma, S, \delta, S_0, F\}$, where

- $\Sigma = 2^P$ is the alphabet
- S is the set of states, which is the set of all possible labels, i.e. the subsets s of $2^{cl(\varphi)}$, where $cl(\cdot)$ is the closure operator, that satisfy:
  - **false** $\notin s$
  - if $\varphi_1 \wedge \varphi_2 \in s$ then $\varphi_1 \in s$ and $\varphi_2 \in s$
  - if $\varphi_1 \vee \varphi_2 \in s$ then $\varphi_1 \in s$ or $\varphi_2 \in s$
- $\delta : S \times \Sigma \rightarrow S$ a transition function
- $S \supseteq S_0 = \{s \in S \mid \varphi \in s\}$ the set of initial states
- $F \subseteq S$ a set of accepting states

After the translation, we calculate the largest non-blocking sub-automaton A_φ^{NB} of A_φ. If A_φ^{NB} is empty then we must refine the LTL specification. Using A_{φ}^{NB} as a model, we create a driving function

$$\Delta : S \times \mathcal{O} \to \{0,1\}^{|\mathcal{C}|}$$

as follows: For every state $s \in S$ let $\Sigma(s)$ denote the set of labels defined at state s and let label $l \in \Sigma(s)$. Each label can be evaluated to $true$ or $false$. Since the resulting automaton is deterministic, only one label can evaluate to $true$. If given the current state of the predicates, such a label exists, say l_{active}, then $C(l_{active}) \in \{0,1\}^{|\mathcal{C}|}$ is the truth vector of controller predicates. We define the $\Delta(s, O)$ function with $O \in \{0,1\}^{|\mathcal{O}|}$ the truth vector of observation predicates, to be $\Delta(s, O) = C(l_{active})$. If no label l is active then define by $A(s, O)$ the set of labels that can be activated at state s given observations O and for every $l_a \in A(s, O)$, $T(l_a, s, O) \in \{0,1\}^{|\mathcal{C}|}$ is the truth vector of controller predicates that activates l_a. Then $\Delta(s, O) = T(l_a, s, O)$. Since A_{φ}^{NB} is non-blocking, at least one label can be activated. Let $S_{0,f} \subseteq S_0$ be the set of initial states for which $\Delta(s_{0,f}, O) \neq \emptyset$. The first member of initial states found to belong to $S_{0,f}$ is chosen to be the initial state s_0. Define $P_i = [O_i \; \Delta(s_i, O_i) \; G]$ to be the predicate truth vector.

Proposition 1: Automaton A_{φ} recognizes the sequence of predicates $P_0, P_1, ..P_i, i \in \mathbb{N}$ defined as above.

Proof: Since we have defined the $\mathcal{U}$ operator to not require it's second argument to become true at some future time, eventualities can be avoided, in the sense that $\varphi_1 \mathcal{U} \varphi_2$ can be true without φ_2 ever becoming active. Now the sequence is correct by construction, since for every i, and automaton state s_i, P_i will activate one of the transitions available at this state. ■

Corollary 1: LTL formula φ is trivially satisfied through the sequence P_i, since A_{φ} accepts all and only the infinite traces represented by the LTL formula φ.

IV. Multi-Agent Motion Controllers

As stated before, basic liveness and safety specifications are fulfilled through the use of a global convergent Multiagent-Navigation controller. This is achieved through the use of Multi-Robot Navigation Functions [7], [8].

Navigation functions are real valued maps realized through cost functions, whose negated gradient field is attractive towards the goal configuration and repulsive wrt obstacles. It has been shown (Koditschek and Rimon [9]) that strict global navigation (i.e. with a globally attracting equilibrium state) is not possible and a smooth vector field on any sphere world, which has a unique attractor, must have at least as many saddles as obstacles. Our assumption that we have spherical robots and spherical obstacles does not constrain the generality of this work since it has been proven [9] that navigation properties are invariant under diffeomorphisms. Arbitrarily shaped robots and obstacles, diffeomorphic to spheres, can be handled. Methods for constructing analytic diffeomorphisms are discussed in ([10],[11]) for point robots and in [12] for rigid body robots.

Let us assume the following situation: We have m mobile robots, and their workspace $W \subset R^2$. (The methodology can be applied as well to higher dimensional workspaces.) Each robot R_i, $i = 1 \ldots m$ occupies a disk in the workspace: $R_i = \left\{ q \in R^2 : \|q - x_i\| \leq r_i \right\}$ where $x_i \in R^2$ is the center of the disk and r_i is the radius of the robot. The configuration of each robot is represented by x_i and the configuration space C is spanned by $x = \left[x_1^T \ldots x_m^T \right]^T$.

We assume that the robot kinematics are trivially described through a first order kinematic model:

$$\dot{x} = u \tag{1}$$

Let $\phi(x, x_0, x_f)$ be a Multi-Robot Navigation Function and x_0 and x_f feasible (i.e. robots do not overlap)initial and final states resp. Then

Proposition 2: [7] The control law

$$u = -\nabla \phi \tag{2}$$

where $\nabla \circ = \left[\begin{array}{cccc} \frac{\partial}{\partial x_1} & \frac{\partial}{\partial y_1} & \cdots & \frac{\partial}{\partial y_m} \end{array} \right] \circ$ and ϕ a multi-robot navigation function, navigates the robotic team from almost all[1] feasible initial configurations $x_0 \in W$ to any feasible final states $x_f \in W$, avoiding collisions.

The constructive process for multirobot navigation functions is described in detail in [8].

In this work we are interested in being able to perform secondary objectives while the team of robotic agents is navigating, while being able at any time to guarantee that our system will be collision free, deadlock free and will eventually converge to it's target. The architecture of the proposed methodology is depicted in the information flow diagram of figure 2.

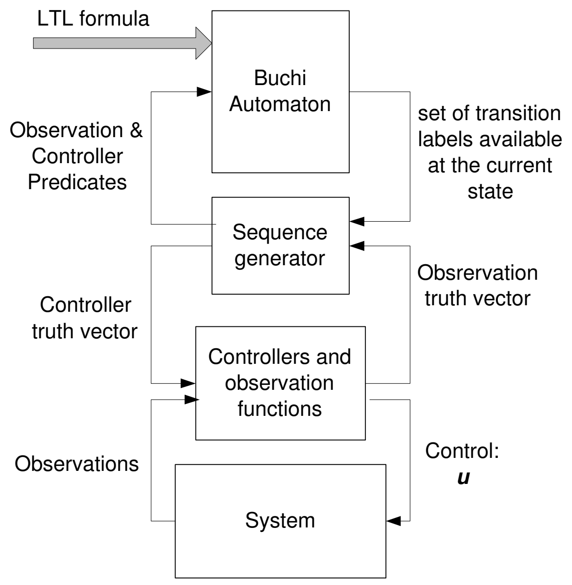
LTL formula
Buchi Automaton
Observation & Controller Predicates
set of transition labels available at the current state
Sequence generator
Obsrervation truth vector
Controller truth vector
Controllers and observation functions
Observations
Control: u
System

Fig. 2. Information flow diagram for the proposed architecture

[1]i.e. all except a set of initial conditions of measure zero.

Before specifying such a controller, let us introduce some preliminary definitions: We define the saturation function:

$$sat\left(x\right)=\begin{cases}-1 & x\leq-1\\ x & -1<x<1\\ 1 & x\geq1\end{cases}\tag{3}$$

Based on the previous we define the continuous switch function

$$s_k\left(x,a\right)=\frac{1}{2}\left(sat\left(\frac{2}{k}\cdot\left(x-a\right)-1\right)+1\right)\tag{4}$$

We can now define the following

$$\begin{aligned}&c_1=s_{\varepsilon_1/2}\left(k_1\left\|\nabla\phi\right\|^2-\tfrac{\varepsilon_1}{2},\tfrac{\varepsilon_1}{2}\right)\\&c_{2-}=s_{\varepsilon_2}\left(-u_2\cdot\nabla\phi,0\right)\\&c_{2+}=s_{\varepsilon_2}\left(u_2\cdot\nabla\phi,0\right)\\&c_3=s_1\left(\tfrac{k_1\left\|\nabla\phi\right\|^2-\frac{\varepsilon_1}{2}}{u_2\cdot\nabla\phi},0\right)\\&\beta=c_1\cdot\left(c_{2-}+c_{2+}\cdot c_3\right)\end{aligned}$$

We can now state the following:

Proposition 3: Let $\phi:\mathcal{W}\rightarrow[0,1]$ be a Multirobot Navigation Function [7] and $u_2\in R^{2m}$ a continuous control signal. Then system (1) under the control law

$$u=-k_1\nabla\phi+\beta\cdot u_2\tag{5}$$

where k_1 a positive constant (gain) and β as defined above, is globally asymptotically stable almost everywhere [2].

Proof: See Appendix A ■

The significance of this controller is that it allows for performing secondary tasks while navigating, while ensuring that the system will eventually converge to it's target, that no collisions will occur and that it will never become deadlocked, with the moderate requirement of u_2 being continuous. We name this controller as "G" as this is the one that is going to be used as the global controller predicate used in our LTL task specification.

Secondary controllers used as predicates in the task definition, are defined based on the task they are about to perform. They can be defined in a multitude of ways, linear or non-linear, as long as they are continuous. This is because if we allowed for discontinuous controller then we should have to take account of the Filippov sets created during the discontinuities and hence limit very much the available control amplitude to maintain the convergence properties. To avoid discontinuities between controller switching we use a fading function which ensures a continuous transition of the exiting signal of the previous controller to the current signal level:

$$u_2^*\left(\tau\right)=u^-\left(t_{Tr}\right)\cdot s_{\varepsilon_3}\left(\varepsilon_3-\tau,\varepsilon_3\right)+u_2\left(\tau\right)\cdot s_{\varepsilon_3}\left(\tau,0\right)\tag{6}$$

where $\tau=t-t_{Tr}$, t_{Tr} is time instant the transition was performed and $u^-\left(t_{Tr}\right)$ the control vector of the previous controller at the time the transition was performed. This function takes the current controller value in $\epsilon_3>0$ time units. Equation (3) can now be written as

$$u=-k_1\nabla\phi+\beta\cdot{u_2}^*\tag{7}$$

To demonstrate the effectiveness of our methodology, we will proceed with construction of linear controllers and linear observing functions. Non-linear controllers and/or observing functions could also be used if required. Note that we can also augment the original system with virtual states and use them as timers e.g. setting $\dot{x}_{virt_i}=u_{virt_i}$ with $u_{virt_i}=1$ whenever a controller on x_{virt_i} is activated, and then use observer functions to get their values.

Let us consider the following simple task: assume we have four robots. Robots should navigate from their initial to their final configuration, while robot 1 and robot 2 create triangular formation with robot 3. Robot 4 is stabilized somewhere in the middle of the route and acts as an obstacle (i.e. it has a controller trying to immobilize it). After robot 1 goes a certain distance ahead of robot 4, a new formation is formed and 4 is a part of it. Predicates needed to specify the task:

- $G(g)$: The global controller with g the target configuration of the system
- $p_{f_1}\in\mathcal{C}$: Triangular formation controller for robots $1,2,3$
- $p_{St_4}\in\mathcal{C}$: Stall controller for robot 4
- $p_{f_2}\in\mathcal{C}$: Diamond formation controller for robots $1,2,3,4$
- $p_{d_{14}}\in\mathcal{O}$: Distance observer of robot 1 from robot 4

The LTL specification formula can now be stated as:

$$\varphi=\Box G\wedge\left(\left(p_{f_1}\wedge p_{St_4}\right)\mathcal{U}p_{d_14}\wedge\bigcirc\Box p_{f_2}\right)\tag{8}$$

Note here that if two controllers (except from the global) are active at the same time, their signals are added in u_2.

After translating the LTL formula φ we get the Buchi automaton A_φ shown in figure 3

With initial conditions G and $\neg p_{d_{14}}$, only state A can be activated by setting $\{p_{f_1},p_{St_4},p_{f_2}\}=\{1,1,0\}$. So we choose $s_0=A$. The matrix below shows the values of $\Delta\left(s,O\right)$ for all the states and observations.

Δ	$p_{d_{14}}$	$\neg p_{d_{14}}$
A	$\emptyset$	$\{1,1,0\}$
B	$\{0,0,1\}$	$\{0,0,1\}$
C	$\{0,0,1\}$	$\{0,0,1\}$

We now proceed with constructing the controllers corresponding to predicates. For the linear case we use controllers of the form $u=-k(Ax+B)$ where the main diagonal of matrix A has non-negative elements. The zero level set of $Ax+B$ represents the geometric relation we wish to achieve between the agents or between agents and workspace. k is a positive gain. Controller whose predicates are active at the same time, will have their control signals added. In the linear case, this operation can be performed on their respective matrices. Following is the definition of the controller for each predicate

[2]i.e. everywhere except a set of initial conditions of measure zero

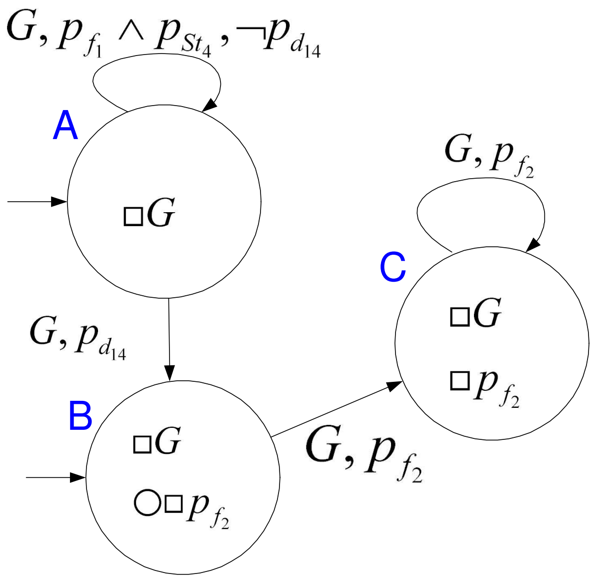

Fig. 3. Buchi Automaton A_φ

- $G(g)$ with $g = [0.9, 0.1, 1.1, 0.1, 0.9, -0.1, 1.1, -0.1]$
- $p_{f_1} \wedge p_{St_4} : B = \begin{bmatrix} -1 & 1 & 0 & -2 & 1 & 1 & 0 & 0 \end{bmatrix}^T$

$$A = \begin{bmatrix} 1 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & -1 & 0 & 0 \\ -1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

- $p_{f_2} : B = \begin{bmatrix} 1 & -1 & 1 & 1 & 1 & -1 & -2 & 0 \end{bmatrix}^T$

$$A = \begin{bmatrix} 1 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & -1 \\ 0 & 0 & 0 & 0 & 1 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & -1 \\ -1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & -1 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

- $p_{d_{14}} : x_1 - x_4 \geq 0.3$

V. Simulation Results

To demonstrate the efficiency of our methodology, we have set up a simulation of the resulting system from the task specification in formula (8) and the setup proposed in the previous section. Disk shaped robots of radius .05 units were used and their initial configuration was set at $x = [-1.1, 0.1, -0.9, 0.1, -1.1, -0.1, -0.9, -0.1]$. Figure 4 depicts the evolution of the system. Fig. 4.a shows the initial and final configurations of the system that are applied to the global controller. Filled disks R_1 through R_4 represent current robot positions while T_1 through T_4 are the

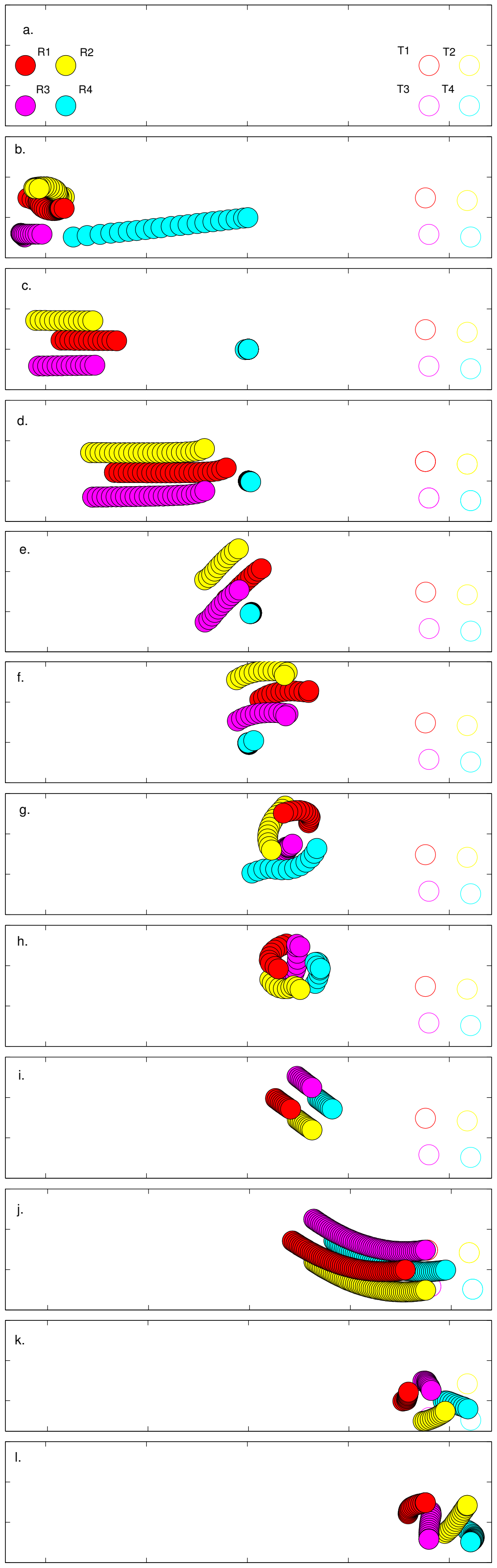

Fig. 4. Simulation results

final positions. Fig. 4.b (state: A), shows the rearrangement of robots R_1, R_2, R_3 in triangular formation (p_{f_1}), while R_4 is stabilized at (0,0) (p_{St_4}). Fig. 4.c-4.e (state: A), shows the formation approaching R_4 and then manoeuvering to avoid collision. In fig. 4.f (state: A), $p_{d_{14}}$ is activated (transition A $\rightarrow$ B followed by transition B $\rightarrow$ C) and in fig. 4.g-4.h (state: C), all four robots start relocating to assume the diamond formation ($p_{d_{14}}$) . Fig. 4.i-4.j (state: C), shows the robots navigating in diamond formation to their target, until in fig. 4.k-4.l (state: C) the signal from the global controller dominates and the robots are guided to their final configurations. As can be seen the system successfully interpreted the LTL specification, while the global controller established a collision free trajectory that converged to the final destination.

VI. Conclusions - Issues for further research

In this paper we have described a methodology for automatically synthesizing motion tasks based on LTL specifications. LTL provides an intuitive way of specifying motion tasks due to it's similarity with natural languages. A controller with guaranteed global convergence and collision avoidance properties was derived, which provided for concurrent execution of secondary controllers. The resulting closed loop system satisfied the LTL specifications by construction, ensuring correct design.

Further research issues include exploiting integer programming methodologies to derive motion plans, i.e. sequences of motion tasks that achieve creation of a specified end product, i.e. automated assembly.

References

[1] E.Klavins, "Automatic synthesis of controllers for assembly and formation forming," *Proceedings of the International Conference on Robotics and Automation*, 2002.

[2] P. Tabuada and G. J. Pappas, "Linear time logic control of linear systems," *Submitted, IEEE Transactions on Automatic Control*, 2004.

[3] M. Egerstedt, "Motion description languages for multi-modal control in robotics," in *Control Problems in Robotics, (A. Bicchi, H. Cristensen and D. Prattichizzo Eds.)*, ser. Springer Tracts in Advanced Robotics. Springer-Verlag, 2002, pp. 75–90.

[4] J. H. V. Manikonda, P.S. Krishnaprasad, "Languages, behaviors, hybrid architectures, and motion control," in *Mathematical Control Theory, Baillieul and Willems eds.* Springer-Verlag, 1998, pp. 199–226.

[5] J. Buchi, "On a decision method in restricted second order arithmetic," *In Proc. Internat. Congr. Logic, Method and Philos. Sci.*, pp. 1–12, Stanford, 1962, Stanford University Press.

[6] P. Wolper, "Constructing automata from temporal logic formulas: A tutorial," in *Lectures on Formal Methods in Performance Analysis*, ser. Lecture Notes in Computer Science, vol. 2090. Springer-Verlag, 2001, pp. 261–277.

[7] S. G. Loizou and K. J. Kyriakopoulos, "Closed loop navigation for multiple holonomic vehicles," *Proc. of IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, pp. 2861–2866, 2002.

[8] S. Loizou and K. Kyriakopoulos, "Multi-robot navigation functions," *submitted, IEEE Transactions on Robotics*, 2004.

[9] D. E. Koditschek and E. Rimon, "Robot navigation functions on manifolds with boundary," *Advances Appl. Math.*, vol. 11, pp. 412–442, 1990.

[10] E. Rimon and D. E. Koditschek, "Exact robot navigation using artificial potential functions," *IEEE Trans. on Robotics and Automation*, vol. 8, no. 5, pp. 501–518, 1992.

[11] ——, "The construction of analytic diffeomorphisms for exact robot navigation on star worlds," *Trans. of the American Mathematical Society*, vol. 327, no. 1, pp. 71–115, September 1991.

[12] H. G. Tanner, S. G. Loizou, and K. J. Kyriakopoulos, "Nonholonomic stabilization with collision avoidance for mobile robots," *Proc. of IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, pp. 1220–1225, 2001.

[13] J. Milnor, *Morse theory*, ser. Annals of Mathematics Studies. Princeton, NJ: Princeton University Press, 1963.

[14] S. G. Loizou and K. J. Kyriakopoulos, "Closed loop navigation for multiple holonomic vehicles," NTUA, available at users.ntua.gr/-sloizou/, Tech. Report, 2002.

Appendix A
Proof of Proposition 3

Since ϕ is a Multirobot Navigation Function, it serves as a Lyapunov function candidate. Taking the derivative of ϕ along the trajectories of (1), we have: $\dot{\phi} = \frac{\partial \phi}{\partial t} + \dot{x} \cdot \nabla \phi = \dot{x} \cdot \nabla \phi$ since $\phi = \phi(x)$. Substituting we get:

$$\dot{\phi} = -k_1 \left\| \nabla \phi \right\|^2 + c_1 \cdot (c_{2-} + c_{2+} \cdot c_3)\, u_2 \cdot \nabla \phi$$

We can discriminate the following cases:

I. $k_1 \left\| \nabla \phi \right\|^2 \leq \frac{\varepsilon_1}{2}$. Then $c_1 = 0$ and $\dot{\phi} = -k_1 \left\| \nabla \phi \right\|^2 \leq 0$. The equality holds at the origin and at a set of measure zero of saddle points. Saddle points for Multirobot Navigation Functions constitute the positive limit set of a measure zero set of initial conditions, since they possess the Morse property [13], [14]. Hence for this case $\dot{\phi} \overset{a.e.}{<} 0$.

II. $k_1 \left\| \nabla \phi \right\|^2 > \frac{\varepsilon_1}{2}$. Then $0 < c_1 \leq 1$ and we can now discriminate the following cases:

  i. $u_2 \cdot \nabla \phi \leq 0$ then $\dot{\phi} \leq -\frac{\varepsilon_1}{2} < 0$

  ii. $u_2 \cdot \nabla \phi > 0$ then $c_{2-} = 0$ and $0 < c_{2+} \leq 1$. Then $\dot{\phi} \leq -k_1 \left\| \nabla \phi \right\|^2 + c_3 \cdot u_2 \cdot \nabla \phi$ and we have the following cases:

    a. $\frac{k_1 \left\| \nabla \phi \right\|^2 - \frac{\varepsilon_1}{2}}{u_2 \cdot \nabla \phi} \leq 1$ then substituting and simplifying $\dot{\phi} \leq -\frac{\varepsilon_1}{2} < 0$

    b. $\frac{k_1 \left\| \nabla \phi \right\|^2 - \frac{\varepsilon_1}{2}}{u_2 \cdot \nabla \phi} > 1$ then expanding the inequality we get $u_2 \cdot \nabla \phi - k_1 \left\| \nabla \phi \right\|^2 < -\frac{\varepsilon_1}{2} \Rightarrow \dot{\phi} < -\frac{\varepsilon_1}{2} < 0$