

Multi-Robot Systems Optimization and Analysis Using MILP and CLP*

Christian Reinl
Technische Universität Darmstadt
Department of Computer Science
Hochschulstr. 10
D-64289 Darmstadt, Germany
reinl@sim.tu-darmstadt.de

Florian Ruh,
Frieder Stolzenburg
Hochschule Harz
Automation and Computer
Sciences Department
Friedrichstr. 57-59
D-38855 Wernigerode,
Germany
{fruh, fstolzenburg}@hs-
harz.de

Oskar von Stryk
Technische Universität
Darmstadt
Department of Computer
Science
Hochschulstr. 10
D-64289 Darmstadt, Germany
stryk@sim.tu-
darmstadt.de

ABSTRACT

Formal methods for multi-robot system analysis, especially logic-based methods, operate on discrete models. Optimization methods for simultaneous trajectory and task allocation, namely mixed integer dynamic optimization, operate on hybrid dynamical models which take into account a model of the motion dynamics of the physical robot. In this paper, ongoing work towards a coherent treatment of both approaches is described. A benchmark problem from robot soccer is introduced and used as an illustrative example.

Keywords

Multi-robot systems; discrete and hybrid system techniques; state machine descriptions; logic-based methods; mixed integer linear programming; constraint logic programming.

1. INTRODUCTION

The development of formal models and methods for multi-robot systems (MRS) aims to provide tools for MRS analysis, including formal verification, performance, or quantification. These approaches usually operate on a relatively high level of abstraction and use discrete models, for example, logic-based methods [3, 14], sequential decision-making approaches, graph-based formation control [9], or discrete system techniques. Usually within such frameworks, properties of the underlying physical processes, which consist of the physical motion of the robot (namely its kinematics and dynamics) and the interaction with its environment, are either neglected or can only be approximated very roughly. Whether these approximations can lead to a satisfying analysis of the MRS depends on the robots and tasks. Robot soccer is a prominent example for a MRS task in a fast-changing environment, where utilizing the robot motion dynamics in an optimal manner is essential for successful task completion.

*Parts of this research have been supported by the German Research Foundation (DFG) within the research training group 1362 *Cooperative, Adaptive, and Responsive Monitoring in Mixed Mode Environments* and the special priority program 1125 on *Cooperating Teams of Mobile Robots in Dynamic Environments*.

In order to represent MRS formally, we must be able to express concurrency or parallelism of behavior. In addition, since the overall MRS may have high complexity, a hierarchical presentation is desirable, so that different levels of abstraction can be expressed. Statecharts as used in the Unified Modeling Language (UML) [20] combine both features. Nevertheless, it is not possible to express physical, continuous system behavior with them. However, hybrid automata, which are normally used for embedded system description, can do this opportunity [2, 12].

The combination of both types of automata leads us to hierarchical hybrid automata. This is discussed in [10] and a procedure is described that allows us to analyze MRS specifications by translating them into (simple) hybrid automata and employing standard model checking tools, e.g. HyTech [13], for system verification. Using these tools, mainly reachability analysis is possible. However, we are often interested not only in the question of whether a certain state or situation can be reached by the whole system at all, but also e.g. how fast it can be reached, i.e., optimality properties have to be dealt with.

Computer algebra systems together with mixed-integer linear programming (MILP) enable us to deal with optimization tasks adequately. Since these optimization tools, most importantly, allow users to express and solve (optimization) problems expressed as a set of equations, the above-mentioned automaton representation of MRS has to be mimicked by equations, in this case by integer equations. This approach has been described in [21, 22]. Reachability and worst-case analysis of MRS can be performed successfully with this method using MILP.

One disadvantage of this approach is the fact that the state hierarchy and general structure are not expressed directly as in the logic-based approach with standard hybrid model checkers [10]. Therefore, we propose a combination of the logic-based and the MILP approach, namely with constraint logic programming (CLP) [19]. This allows us to perform system analyses, on the one hand, and system optimization on the other hand. MRS can be specified quite naturally with this method. Further, the power of mixed integer programming (MIP), i.e., including nonlinear optimization problems, can be combined by an abstract and logical representation of the whole system.

In the following, we first review the state of the art in modeling multiagent systems and approaches for optimization and verification, respectively (Sect. 2). Then, we introduce a representative example from robot soccer which helps us to explain the methods pro-

sented in this paper. next, we define hierarchical hybrid automata (Sect. 3.2), laying the formal foundation for the rest of the paper. Furthermore, we discuss the MILP method (Sect. 4). Finally, we describe the ongoing work on combining MILP with CLP (Sect. 5) before we present our conclusions.

2. MODELING MULTI-ROBOT SYSTEMS

We will now briefly review related work, especially those involving robot soccer for we will use this as an example throughout the paper. Formal soccer theories have been developed for or derived from human soccer games in [8, 17]. They define certain strategies for optimal defensive and offensive play. Thus, they can also help to model behavior specifications in robot soccer. Hybrid multi-agent systems with timed synchronization are introduced in [10]. They are used for specification and model checking. For this, the agents can map soccer theories to their own and therefore to the overall behavior. These statecharts allow us to simulate continuous actions in states and discrete actions at state transitions. Synchronization is made possible by using special synchronization points — which can be e. g. a soccer ball — with which each agent can interact and which characterizes the team play.

Using hybrid automata [12] is a well accepted method to model and analyze mobile multiagent systems [1, 2]. Hierarchical, hybrid automata are used for building up and describing multi-layer control architectures based on physical motion dynamics of moving agents [5]. In many applications they form a link between multi-vehicle systems and theories of hybrid systems like in [24].

CLP is a programming paradigm where relations among variables are expressed via constraints. It combines logic programming and constraint solving [7]. CLP has already been applied to modeling of hybrid systems including solving differential equations [14]. However, efficiency can only be expected if a full CLP language is employed as e. g. Eclipse Prolog [16], where several constraint solvers are available, including constraints over continuous and integer variables and optimization on continuous variables.

Hybrid systems, CLP, and optimization are closely related, and especially piecewise affine systems are studied by using MILP. [18] gives a good overview of approaches and chances in combining CLP and MILP. [3] presents a combined approach for investigating the optimal control of hybrid systems.

3. ROBOT SOCCER AND AUTOMATA

3.1 Example from Robot Soccer

Robot soccer is a closed, secure scenario for MRS and also a simple and descriptive example for robotic team play. Thus, it is accepted as a step forward to investigations on open, uncertain, unexplored, and insecure environments.

As a common benchmark problem we are looking at the setting of Fig. 1 with two attacking robots, one defender and one ball. Thus, we are considering a system containing a generalizable characteristic with (in-)directly controllable and non-controllable moving agents $i \in \{1, 2, B, D\}$. Each one is characterized by its continuous, dynamic state \mathbf{x}_i (e. g. position, velocity, etc.) and a discrete state s that denotes a certain subtask or role. Together with the continuous control variable \mathbf{u}_i , the continuous state evolves subject to $\dot{\mathbf{x}}_i = \mathbf{f}_{s,i}(\mathbf{x}_i, \mathbf{u}_i)$. As modes of motion s for the strikers we distinguish `free_moving` and `dribbling` (cf. Fig. 2).

Furthermore, certain logical expressed constraints have to be considered like the fact that a robot has to be close to the ball for being able to kick. Eventually, by defining (usually unknown) switching times t_k and corresponding specifications (how to connect \mathbf{x}_i when s

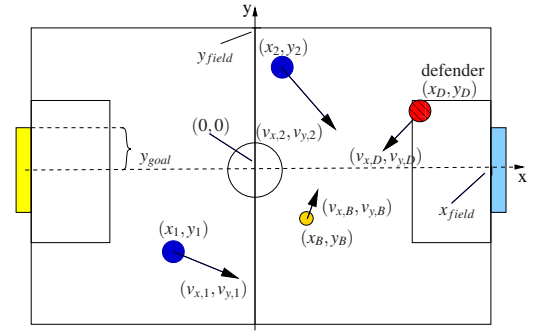


Figure 1: Setting of the robot soccer benchmark problem.

switches at t_k), the individual trajectory for an object is determined. Our intention is to answer questions like the following:

- What is the fastest way to score a goal?
- How many ball contacts are required to score a goal?
- What is the best tactical improvement for fixed time horizon?

3.2 Hierarchical Hybrid Automata

Let us now formally define hierarchical hybrid automata (abbreviated HHA here), following the lines of [10]. In order to describe physical multiagent systems adequately, the definitions combine aspects from UML statecharts and hybrid automata [12]. We start with basic components and the definition of a state hierarchy.

DEFINITION 1 (BASIC COMPONENTS). *The basic components of a state machine are the following disjoint sets:*

S : a finite set of states, partitioned into three disjoint sets: S_{simple} , S_{comp} , and S_{conc} — called simple, composite and concurrent states, containing one designated start state $s_0 \in S_{comp} \cup S_{conc}$;

X : a finite set of variables, partitioned into two disjoint sets: X_{real} and X_{int} — the continuous/real-numbered and the integral/integer variables, respectively; for each $x \in X$ we introduce the variables x' for the conclusions of a discrete change;

T : a finite set of transitions with $T \subseteq S \times S$.

DEFINITION 2 (STATE HIERARCHY). *Each state s is associated with zero, one or more initial states $\alpha(s)$: a simple state has zero, a composite state exactly one and a concurrent state more than one initial state. Moreover, each state $s \in S \setminus \{s_0\}$ is associated to exactly one superior state $\beta(s)$. Therefore, it must hold $\beta(s) \in S_{conc} \cup S_{comp}$. A concurrent state must not directly contain other concurrent ones. In this case, the composite state s is called region. Furthermore, it is assumed that all transitions $s_1 T s_2 \in T$ keep to the hierarchy, i. e. $\beta(s_1) = \beta(s_2)$. We write $\alpha^n(s)$ or $\beta^n(s)$ for the n -fold application of α or β to s , in particular, $\alpha^0(s) = \beta^0(s) = s$. Let us now have a closer look at variables. Variables $x \in X$ may be declared locally in a certain state $\gamma(x) \in S$. A variable $x \in X$ is valid in all states $s \in S$ with $\beta^n(s) = \gamma(x)$ for some $n \geq 0$, unless another variable with the same name overwrites it locally.*

In order to keep things simple, we focus on linear system dynamics in the sequel, because otherwise many problems with HHA become undecidable. In addition, we subsume the notions *flow*

condition and invariant from the literature (see e. g. [12]) in the notion *state condition* in our next definition. Invariants are conditions that must hold all the time the automaton remains within one state. For general, nonlinear dynamics, in particular with non-monotonic value patterns of the real-numbered variables, these invariants would be difficult to test. Also, we will not introduce derivatives \dot{x} or \dot{x} for the variables in X in our definitions, although we may use it for the ease of notation. We e. g. allow predicates of the form $x = u \cdot t + x_0$, what can be understood as a solution of the differential equation $\dot{x} = u$, that is the flow condition itself or conjunctions thereof. In our graphical examples, however, we mark flow conditions with **f** and invariants with **i**.

DEFINITION 3 (JUMP AND STATE CONDITIONS). *For each transition, there exists a jump condition. This is a predicate with free variables from the valid variables of $X \cup X'$. Additionally, each state $s \in S$ contains a state condition which describes continuous changes in s . It is a predicate with free variables from $X \cup \{t\}$ where the time t may occur linearly.*

Jump conditions are marked with **j** in our examples. In addition, we annotate so-called events with **e**. Events are well-known in UML statecharts and hybrid automata. They can easily be expressed by (binary) integer variables in our formalism. Therefore, we do not introduce them explicitly in our definitions. In contrast to simple hybrid automata, we introduce hierarchies. The behavior of an HHA cannot be described by a sequence of states but by trees of states, called configuration.

DEFINITION 4 (CONFIGURATION AND COMPLETION). *A configuration c is a rooted tree of states where the root node is the topmost initial state of the overall state machine. Whenever a state s is an immediate predecessor of s' in c , it must hold $\beta(s') = s$. A configuration must be completed by applying the following procedure recursively as long as possible to leaf nodes: if there is a leaf node in c labeled with a state s , then introduce all $\alpha(s)$ as immediate successors of s .*

The semantics of our automata can now be defined by alternating sequences of discrete and continuous steps. Following the synchrony hypothesis, we assume that discrete state changes happen in zero time. Continuous steps (within one state) may last some time. Although this time is not fixed in advanced, often a clocking mechanism is introduced, i. e. the flow of time is discretized. If we do this instead of differential equations, we can use difference equations (see Sect. 4).

DEFINITION 5 (SEMANTICS). *The state machine starts with the initial configuration, that is the completed topmost initial state s_0 of the overall state machine. In addition, an initial condition must be given as a predicate with free variables from $X \cup \{t\}$. The current situation of the multiagent system can be characterized by a triple (c, v, t) where c is a configuration, v is a valuation (i. e. a mapping $v : X \rightarrow \mathbb{R}$, where the integral variables are mapped to integers), and t is the current time. The initial situation at time $t = 0$ is a situation (c, v, t) where c is the initial configuration and v satisfies the initial condition. The following steps are possible in a situation (c, v, t) :*

discrete step: *a discrete/micro-step from one configuration c of a state machine to a configuration (c', v', t) by means of a transition $sT s'$ with some jump condition in the current situation (written $c \rightarrow c'$) is possible iff:*

1. c contains a node labeled with s ;

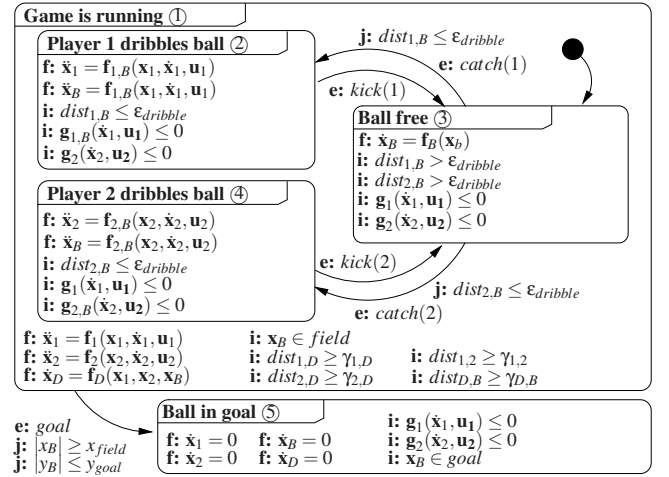


Figure 2: Hierarchical hybrid automaton model of the switched motion dynamics.

2. the jump condition of the given transition holds in the current situation (c, v, t) ;
3. c' is identical with c except that s together with its subtree in c is replaced by the completion of s' ;
4. variables in X' are set according to the jump condition.

continuous step: *a continuous step/flow within the actual configuration to the situation (c, v', t') requires the computation of all x that are valid in c at the time t' , according to the conjunction of all flow conditions of the states $s \in c$, where it must hold $t' > t$.*

3.3 Automaton for the Soccer Example

For the proposed soccer application (cf. Fig. 2) we use two layers as hierarchy. In this modeling we only distinguish whether the ball is rolling free, being dribbled or inside the goal. The respective motion dynamics of a dribbling robot is indexed by B . The initial conditions are defined with the position $\mathbf{x}_i(t_0)$ of the objects $i \in \{1, 2, D, B\}$. Catching and kicking a ball are modeled by events

$$\text{kick}(\diamond) : (\dot{\mathbf{x}}_B)' = 3 \cdot \dot{\mathbf{x}}_\diamond, \quad \text{catch}(\diamond) : \mathbf{x}'_B = \mathbf{x}_\diamond, \quad (\diamond \in \{1, 2\}).$$

All other state trajectories are required to be continuous at t_k .

We are testing our approach with a simple model for the dynamic of robots and ball and set for all states except (5) (cf. 2)

$$\mathbf{f} : \dot{\mathbf{x}}_B(t) = \mathbf{v}_B(t), \quad \mathbf{f} : \dot{\mathbf{x}}_\diamond(t) = \dot{\mathbf{v}}_\diamond(t) = \mathbf{u}_\diamond(t) \quad (1)$$

($\diamond \in \{1, 2\}$). For a dribbling robot the upper bound on its velocity is reduced by a factor $c_{v,d}$.

Further constraints on state or control variables according to the motion modes are modeled as invariants **i**. In this example, all controls and velocities are bounded by quadratic expressions

$$\mathbf{i} : g_{s,i}(\dot{\mathbf{x}}_i) = \|\mathbf{v}_i\|_2 - v_{s,i}^{UB} \leq 0$$

with a constant upper bound $v_{s,i}^{UB}$ ($\mathbf{u}_{s,i}$ respectively). A distance measure between objects i and j is represented by the auxiliary variable $\text{dist}_{i,j}$. It is used to express, for example, collision avoidance (with a constant threshold $\gamma_{i,j}$).

In order to apply efficient linear methods, we are approximating the physical model by linearized reformulations. We are transform-

ing the differential flow conditions \mathbf{f} into difference equations

$$\dot{\mathbf{x}} = \mathbf{f}_s(\mathbf{x}, \mathbf{u}) \rightsquigarrow \frac{\mathbf{x}(t + \Delta_t) - \mathbf{x}(t)}{\Delta_t} = A_s \cdot \mathbf{x}(t) + B_s \cdot \mathbf{u}(t). \quad (2)$$

Additional state variables and binary variables may be necessary in general for accurate approximations. In the context of hybrid automata, these case differentiations could be treated as new sub-knots. This splitting up strongly depends on the degree of nonlinearity and the desired accuracy of the transformed model. Application to the example results in

$$\frac{x_{\diamond}(t + \Delta_t) - x_{\diamond}(t)}{\Delta_t} = v_{x,\diamond}(t), \quad \frac{v_{x,\diamond}(t + \Delta_t) - v_{x,\diamond}(t)}{\Delta_t} = u_{x,\diamond}(t),$$

$$\frac{x_B(t + \Delta_t) - x_B(t)}{\Delta_t} = v_{x,B}(t), \quad v_{x,B}(t + \Delta_t) = \begin{cases} c_{trac} \Delta_t v_{x,B}(t) & \textcircled{5} \\ \Delta_t v_{x,\diamond}(t) & \textcircled{1} \end{cases}$$

($\diamond \in \{1, 2\}$, $c_{trac} \cdot \Delta_t < 1$, $y_i, v_{y,i}$, analogously). The solution space, defined by the invariants, gets approximated in a polygonal manner. This is modeled by logically combined linear inequalities. Jump conditions \mathbf{j} and events \mathbf{e} are linearized respectively.

A simple, reactive defender that is always moving towards the current ball position is controlled by

$$\frac{x_D(t - \Delta_t) - x_D(t)}{\Delta_t} = u_{x,D}(t) := \frac{v_D^{UB}}{D_{max}} (x_B(t) - x_D(t))$$

($D_{max} \geq \max_{x,y,t} \{|x_B(t) - x_D(t)|, |y_B(t) - y_D(t)|\}$; $y_D, v_{y,D}$, analogously). v_D^{UB} again is the constant upper bound for $\|v_D\|_2$.

4. OPTIMIZATION BASED ON MIP

Optimization of mobile cooperative MRS leads us in the general case to mixed integer nonlinear optimal control problems (MIOCP), also known as hybrid optimal control problems (HOCP). One way to solve these problems would be to combine of branch-and-bound techniques with direct collocation [11].

In order to find where it would be possible to combine efficient MIP and CLP for optimizing and analyzing scenarios in multiple mobile robot systems, we are looking on our simplified model to treat with a common benchmark example. There is a need of efficient CLP- and MIP-based control algorithms which consider the intrinsic characteristics of the system, especially the tasks to be performed and the agents' physical abilities.

We are looking for optimal parameters or optimal controls for a cooperative task according to detailed system's characteristics. This also is related to the main challenge for preparing a receding horizon control algorithm, – a well-adapted method for cooperative tasks in permanently changing and unstable environments.

As a general method to build up an optimization based algorithm we are using a reduced linear model. The linear case can be solved much more efficiently (cf. [6]) than the MIOCP.

The linearized problem allows us to answer different questions like finding sub-optimal control inputs or computing costs for reaching certain states. MILP models can cope well with non-convexities and the combinatorial character which is an intrinsic feature in modeling most MRS. Efficient solvers are making MILP applicable even to deal with abrupt changes in uncertain environments [21].

4.1 Discrete-time MIP Formulation

We are introducing a time structure that allows us to discretize the regarded point mass model. Thus we are defining

$$t_0 = 0, \quad t_{k+1} := t_k + \Delta_t$$

as a number of time steps with a constant sampling Δ_t . The time interval $[0, t_f]$ for the system evolution is marked by $t_f = t_{N+1}$

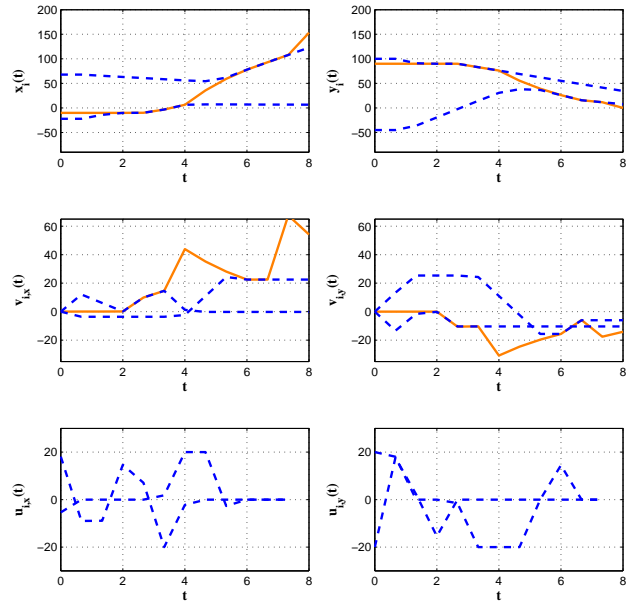


Figure 3: Optimal positions, velocities and controls for the attackers (---) and the ball (—).

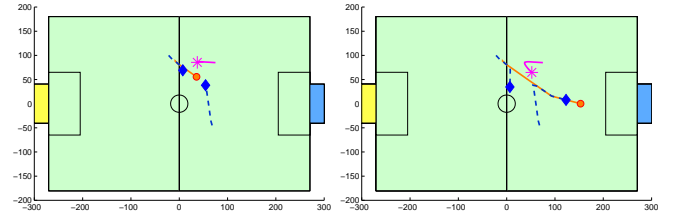


Figure 4: Computed optimal behavior shown at time steps $k = 8$ and $k = 13$. Attacker 1 goes to ball, dribbles and kicks it towards his teammate 2 who catches it there and kicks it towards the desired direction. The defender (—*) follows the ball.

For each states $s \in S_{comp} \cup S_{conc}$ a binary variable $b_s \in \{0, 1\}$ is introduced. $b_s(t_k) = 1$ iff the state s is active at the time t_k . Thus, we can use simple inequalities to model the transition $sT s'$ that occur at t_{k+1}

$$\forall k \in \{1, 2, \dots, N\} : b_s(t_k) \geq b_{s'}(t_{k+1})$$

and hierarchies

$$\forall k \in \{1, 2, \dots, N\} : b_s(t_{k+1}) = \sum_{s' \in \{s' | \beta(s')=s\}} b_{s'}(t_{k+1}).$$

Logical relations combined with inequalities are translated using the *Big-M technique* (cf. [4]). Thus, flows get connected with the respective binary variable, e. g.

$$\begin{aligned} \text{IF } b_s = 1 \text{ THEN } g_s(\mathbf{x}, \mathbf{u}) &\leq 0 \\ \Leftrightarrow [g_s(\mathbf{x}, \mathbf{u}) \leq (1 - b_s)M] \end{aligned}$$

with a constant $M \geq \max_{\mathbf{x}, \mathbf{u}} \{g_s(\mathbf{x}, \mathbf{u})\}$. As we model the moving object using a point mass (cf. Eqs. 1 and 2), these expressions are translated by using finite differences as follows:

$$x(t_{k+1}) = x(t_k) + \Delta_t v_x(t_k), \quad v_x(t_{k+1}) = v_x(t_k) + \Delta_t u_x(t_k).$$

Generally, the nonlinear expression for measuring a Euclidean distance $\sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2} \leq r$ is approximated by a bunch of $n_d \geq 4$ fixed linear constraints

$$\sin\left(\frac{2i}{n_d}\pi\right)(x_1 - x_2) + \cos\left(\frac{2i}{n_d}\pi\right)(y_1 - y_2) \leq r \quad (i = 1, \dots, n_d). \quad (3)$$

Thus, the respective invariants \mathbf{i} and the distances $dist_{i,j}$ got linearized (with $n_d = 4$ in our example).

To answer certain questions to the system (cf. Sect. 3.1), one has to formulate the interesting feature with an objective function to be minimized. For this, all non-fixed discrete or continuous values in the system's model could be weighted and combined, e.g. the values $b_s(t_k)$ which are representing certain states in the automaton, $\mathbf{x}(t)$ denoting positions or any auxiliary variable like the distance $dist_{BD}$. In order to use efficient MILP/MIQP solvers, we are looking for linear and quadratic formulations for the objective. As an example, we looked at the following questions. An additional result for a similar model can also be found in [21], where we asked for the best tactical improvements.

Farthest shoot on goal.

What is the longest possible distance that a ball may roll towards the goal if at least one pass is required? More precisely, we require that each of the attackers dribbles the ball at least one time. For given time steps $t_1 \dots t_{N+1}$ we add the constraints $\sum_{k=1}^{N+1} b_s(t_k) \geq 1$ for $s \in \{\textcircled{2}, \textcircled{4}\}$ to our MILP and minimize the objective function

$$-x_B(t_{N+1}) + q_y(t_{N+1}) + \varepsilon \cdot \sum_{i \in \{1,2\}} \sum_{k=1}^N q_{u,i}, \quad (4)$$

with $-q_y \leq y_B \leq q_y$ and $\sqrt{(u_{i,x}^2 + u_{i,y}^2)} \leq q_{u,i}$ which is transformed into a set of linear constraints following the idea in Eq. (3). The last sum in (4) is added with a low weight $\varepsilon = 0.01$ to get sensible values for all controls – because of many remaining degrees of freedom in the system otherwise.

Results for the linear implementation of the proposed benchmark problem with following parameters are shown in Fig. 3 and 4.

$$\begin{aligned} \Delta_r &= \frac{2}{3}, & N &= 12, & \varepsilon_{dr} &= 5, & c_{v,dr} &= 60.7, & v_{\diamond,x}^{UB} &= 45 \\ \gamma_{2,D} &= 30, & \gamma_{B,D} &= 30, & D_{max} &= 700, & c_{trac} &= 0.8, & v_{B,x}^{UB} &= 135, \\ v_{\diamond,y}^{UB} &= 45, & u_{\diamond,x}^{UB} &= 20, & u_{\diamond,y}^{UB} &= 20, & v_{\diamond,y}^{UB} &= 90, & \diamond &\in \{1, 2\} \end{aligned}$$

The MILP was solved with CPLEX [15] on a PC (Intel(R) Pentium(R) M processor 1.86GHz; 1024 MB RAM) in 9 sec.

The time-optimal case.

Now we are interested in the minimal time for reaching certain (sets of discrete or continuous) states. For instance, we pose the question of the fastest way to score a goal. To answer this question we give up the constant sampling time Δ_k and replace it by a free variable $\Delta_k \in [t_{min}, t_{max}]$ to be minimized. Thus, we are resulting in a MIP with quadratic constraints. Here, the objective function becomes:

$$\sum_{k=1}^N \Delta_k + \varepsilon \cdot \sum_{i \in \{1,2\}} \sum_{k=1}^N q_{u,i} \quad (5)$$

subject to the additional constraint $b_{\textcircled{5}}(t_{N+1}) = 1$. The last sum in (5) is necessary for the same reason as in (4). Otherwise, it would not be clear for the second robot what to do while the first one is handling the ball.

In ongoing work, we are looking forward to solving this problem with a mixed integer nonlinear programming approach using

```
%%% step(+Config,-Next)
%%% perform transitions
step([State|_],Tree) :-
    trans(State,Next),!,
    complete(Next,Tree).
step([Top|Sub],[Top|Tree]) :-
    maplist(step,Sub,Tree).
step([],[]).

%%% complete(+State,-Tree)
%%% build completed Tree below State
complete(State,[State|Complete]) :-
    init(State,Init),
    maplist(complete,Init,Complete).
```

Figure 5: State machine in Prolog for discrete state changes.

```
start(game_is_running).

init(game_is_running,[ball_free]).
init(player_1_dribbles_ball,[]).
init(ball_free,[]).
init(player_2_dribbles_ball,[]).
init(ball_in_goal,[]).
```

Figure 6: State hierarchy in Prolog for the example in Fig. 2.

sequential quadratic optimization (SQP) which can cope very well with the quadratic structure in the motion dynamics here.

5. OPTIMIZATION CONSTRAINTS AND CONSTRAINT LOGIC PROGRAMMING

With the approach presented so far, it is easily possible to express an entire MRS and perform optimality analyses. However, structure information from the specification is lost because the state machine for discrete state changes cannot be expressed explicitly, but only by equations. Therefore, it is useful to employ CLP [19] as then constraint solving for optimization tasks and logic programming are combined. In the following, we will focus on implementing HHA by using logic programming.

A pure state machine for discrete changes can easily be implemented in the declarative programming language Prolog [7]. Fig. 5 shows a meta-program realizing the state machine. It mimics steps according to Def. 5 in the predicate `step` and completion according to Def. 4 in the predicate `complete`. Configurations are encoded in Prolog lists, where the head of a list corresponds to the root of the respective configuration tree. The initial, completed configuration for the example in Fig. 2 e.g. can thus be represented as `[game_is_running, [ball_is_free]]`.

The Prolog code for the concrete specification (shown in Fig. 6) contains the fact `start` denoting the state s_0 and facts for the initial states (predicate `init`). The latter predicate is also used for simple states (in this case the list of initial states is empty) and concurrent states (then this list contains more than one state – one for each region). The predicate `trans` realizes the transitions; it contains the jump conditions and actions of the respective transition in the body. This Prolog implementation technique has been applied successfully in the RoboCup 2D soccer simulation league (see [23]).

If we employ a CLP system like Eclipse Prolog [16], we can implement the discrete state machine in pure Prolog, while the state

```

evolve(Sit1,Sit2) :-
    discrete_step(Sit1,Sit2)
;   continuous_step(Sit1,Sit2).

discrete_step((Conf1,Var1,T),(Conf2,Var2,T)) :-
    transition(...).

continuous_step((Conf,Var1,T1),(Conf,Var2,T2)) :-
    flow(...),
    T2 >= T1.

```

Figure 7: CLP scheme for HHA.

transitions and the checking of all state and jump conditions can be handled by the constraint solver. There are CLP systems that can even solve differential equations [14]. Furthermore, optimization tasks and more symbolic model checking can be done with constraint languages. The constraint solver supports two tasks: on the one hand, simple Prolog queries allow us to express reachability analysis, as can be done with standard model checkers like HyTech [13]; on the other hand, optimization tasks can be solved by employing built-in constraint solvers for integers and reals [16].

Fig. 7 shows the scheme for HHA in CLP. It allows the treatment of HHA descriptions in Prolog notation. Optimization, jump conditions, and flows can be expressed by constraints. The respective predicates are prefixed with \$. This leads to efficient processing in reachability and optimization analyses. This is the subject of ongoing work.

6. CONCLUSIONS

In computer science, the design and analysis of multiagent systems has been treated extensively. Unlike agents in the virtual world, MRS performance is governed by the laws of physics, real-time and uncertainty. In this paper, MRS are considered, whose physical motion dynamics, disturbances, and uncertainties allow a deterministic model consisting of logic operations, differential, and algebraic equations. The aim is to optimize and analyze system behavior. With a combination of constraint logic (CLP) and mixed integer programming (MIP) the whole system can be modeled for optimization while retaining its structure information. Thus, we can produce highly efficient multiagent systems as far as the motion dynamics can be linearized with sufficient accuracy. Using CLP and MILP, makes it possible, e. g., to create, analyze, and control a team of robots playing soccer or, for a more serious propose, a swarm of robots monitoring and checking a sewage network. Even the design of rescue robots could benefit from this work and lead to a consistent and verified behavior control.

7. REFERENCES

- [1] R. Alur, J. M. Esposito, M. Kim, V. Kumar, and I. Lee. Formal modeling and analysis of hybrid systems: A case study in multi-robot coordination. In *World Congress on Formal Methods (1)*, pages 212–232, 1999.
- [2] R. Alur, T. A. Henzinger, and P.-H. Ho. Automatic symbolic verification of embedded systems. *IEEE Transactions on Software Engineering*, 22(3):181–201, 1996.
- [3] A. Bemporad and N. Giorgetti. Logic-based methods for optimal control of hybrid systems. *IEEE Trans. Automatic Control*, 51:963–976, 2006.
- [4] A. Bemporad and M. Morari. Control of systems integrating logic, dynamics, and constraints. *Automatica*, 35(3):407–427, 1999.
- [5] J. Borges de Sousa, K. H. Johansson, J. Silva, and A. Speranzon. A verified hierarchical control architecture for coordinated multi-vehicle operations. *International Journal of Adaptive Control and Signal Processing*, 21(2–3):159–188, 2007. Special issue on autonomous adaptive control of vehicles.
- [6] F. Borrelli, D. Subramanian, A. U. Raghunathan, and L. T. Biegler. Milp and nlp techniques for centralized trajectory planning of multiple unmanned air vehicles. *American Control Conference*, June 2006.
- [7] W. F. Clocksin and C. S. Mellish. *Programming in Prolog*. Springer, Berlin, Heidelberg, New York, 4th edition, 1994.
- [8] F. Dylla, A. Ferrein, G. Lakemeyer, J. Murray, O. Obst, T. Röfer, S. Schiffer, F. Stolzenburg, U. Visser, and T. Wagner. Approaching a formal soccer theory from behaviour specifications in robotic soccer. In P. Dabnichcki and A. Baca, editors, *Computers in Sport*, chapter 6, pages 161–185. WIT Press, London, 2008. To appear.
- [9] G. Ferrari-Trecate, M. Egerstedt, A. Buffa, and M. Ji. Laplacian sheep: A hybrid, stop-go policy for leader-based containment control. hybrid systems: Computation and control. *Hybrid Systems: Computation and Control*, pages 212–226, 2006.
- [10] U. Furbach, J. Murray, F. Schmidsberger, and F. Stolzenburg. Hybrid multiagent systems with timed synchronization – specification and model checking. In M. Dastani, A. E. F. Seghrouchni, A. Ricci, and M. Winikoff, editors, *Post-Proceedings of 5th International Workshop on Programming Multi-Agent Systems held with 6th International Joint Conference on Autonomous Agents & Multi-Agent Systems*, LNAI 4908, pages 205–220. Springer, Berlin, Heidelberg, New York, 2008. To appear.
- [11] M. Glocker, C. Reinl, and O. von Stryk. Optimal task allocation and dynamic trajectory planning for multi-vehicle systems using nonlinear hybrid optimal control. In *Proc. 1st IFAC-Symposium on Multivehicle Systems*, pages 38–43, Salvador, Brazil, October 2-3 2006.
- [12] T. Henzinger. The theory of hybrid automata. In *Proceedings of the 11th Annual Symposium on Logic in Computer Science*, pages 278–292, New Brunswick, NJ, 1996. IEEE Computer Society Press.
- [13] T. A. Henzinger, P.-H. Ho, and H. Wong-Toi. HyTech: The Next Generation. In *IEEE Real-Time Systems Symposium*, pages 56–65, 1995.
- [14] T. J. Hickey and D. K. Wittenberg. Using analytic CLP to model and analyze hybrid systems. In *FLAIRS Conference*. AAAI Press, 2004.
- [15] ILOG. *CPLEX 10.0, User’s Manual*, 2006.
- [16] M. W. Krzysztof R. Apt. *Constraint Logic Programming Using Eclipse*. Cambridge University Press, Cambridge, UK, 2007.
- [17] M. Lucchesi. *Coaching the 3-4-1-2 and 4-2-3-1*. Reedswain Publishing, 2002.
- [18] L. Magatao. *Mixed Integer Linear Programming and Constraint Logic Programming towards a Unified Modeling Framework*. PhD thesis, The Federal Center of Technological Education of Parana, May 2005.
- [19] K. Marriott and P. J. Stuckey. *Programming with Constraints*. MIT Press, Cambridge, MA, London, 1998.
- [20] Object Management Group,

<http://www.omg.org/spec/UML/2.1.2/Infrastructure/PDF/>.
*OMG Unified Modeling Language (OMG UML),
Infrastructure, V2.1.2, 2007.*

- [21] C. Reinl and O. von Stryk. Optimal control of cooperative multi-robot systems using mixed-integer linear programming. In *Proc. RoboMat 2007 - Workshop on Robotics and Mathematics*, Coimbra, Portugal, Sept. 17-19 2007.
- [22] C. Reinl and O. von Stryk. Optimal control of multi-vehicle systems under communication constraints using mixed-integer linear programming. In *Proceedings of the First International Conference on Robot Communication and Coordination (RoboComm)*, Athens, Greece, Oct. 15-17 2007. ICTS.
- [23] F. Stolzenburg. Multiagent systems and RoboCup: Specification, analysis, and theoretical results. Habilitation, Universität Koblenz-Landau, Koblenz, 2005. Reviewers: Armin Cremers, Ulrich Furbach, and Klaus Troitzsch.
- [24] S. Zelinski, T. J. Koo, and S. Sastry. Hybrid system design for formations of autonomous vehicles. In *Proceedings. 42nd IEEE Conference on Decision and Control*, volume 1, pages 1–6, 2003.