A SAT-Based Hybrid Solver for Optimal Control of Hybrid Systems

Alberto Bemporad and Nicolò Giorgetti

Dip. Ingegneria dell'Informazione, University of Siena,
via Roma 56, 53100 Siena, Italy
{bemporad,giorgetti}@dii.unisi.it

Abstract. Combinatorial optimization over continuous and integer variables was proposed recently as a useful tool for solving complex optimal control problems for linear hybrid dynamical systems formulated in discrete-time. Current approaches are based on mixed-integer linear or quadratic programming (MIP), which provides the solution after solving a sequence of relaxed standard linear (or quadratic) programs (LP, QP). An MIP formulation has the drawback of requiring conversion of the discrete/logic part of the hybrid problem into mixed-integer inequalities. Although this operation can be done automatically, most of the original discrete structure of the problem is lost during the conversion. Moreover, the efficiency of the MIP solver mainly relies upon the tightness of the continuous LP/QP relaxations. In this paper we attempt to overcome such difficulties by combining MIP and techniques for solving constraint satisfaction problems into a "hybrid" solver, taking advantage of SAT solvers for dealing efficiently with satisfiability of logic constraints. We detail how to model the hybrid dynamics so that the optimal control problem can be solved by the hybrid MIP+SAT solver, and show that the achieved performance is superior to the one achieved by commercial MIP solvers.

1 Introduction

Over the last few years we have witnessed a growing interest in the study of dynamical processes of a mixed continuous and discrete nature, denoted as hybrid systems, both in academia and in industry. Hybrid systems are characterized by the interaction of continuous models governed by differential or difference equations, and of logic rules, automata, and other discrete components (switches, selectors, etc.). Hybrid systems can switch between many operating modes where each mode is governed by its own characteristic continuous dynamical laws. Mode transitions may be triggered internally (variables crossing specific thresholds), or externally (discrete commands directly given to the system). The interest in hybrid systems is mainly motivated by the large variety of practical situations where physical processes interact with digital controllers, as for instance in embedded control systems.

Despite the fact that the first paper on hybrid systems appeared in the sixties [1], only in very recent years several modelling frameworks for hybrid systems

R. Alur and G.J. Pappas (Eds.): HSCC 2004, LNCS 2993, pp. 126–141, 2004.

have been proposed, we refer the interested reader to [2, 3] and the references therein. Several authors focused on the problem of solving optimal control problems for hybrid systems. For continuous-time hybrid systems, most of the literature either studied necessary conditions for a trajectory to be optimal, or focused on the computation of optimal/suboptimal solutions by means of dynamic programming or the maximum principle [4, 5, 6].

The hybrid optimal control problem becomes less complex when the dynamics is expressed in discrete-time, as the main source of complexity becomes the combinatorial (yet finite) number of possible switching sequences. In particular, in [7, 8, 9] the authors have solved optimal control problems for discrete-time hybrid systems by transforming the hybrid model into a set of linear equalities and inequalities involving both real and (0-1) variables, so that the optimal control problem can be solved by a mixed-integer programming (MIP) solver.

An MIP solver provides the solution after solving a sequence of relaxed standard linear (or quadratic) programs (LP, QP). A potential drawback of MIP is (1) the need for converting the discrete/logic part of the hybrid problem into mixed-integer inequalities, therefore losing most of the original discrete structure, and (2) the fact that its efficiency mainly relies upon the tightness of the continuous LP/QP relaxations.

Such a drawback is not suffered by techniques for solving constraint satisfaction problems (CSP), i.e., the problem of determining whether a set of constraints over discrete variables can be satisfied. Under the class of CSP solvers we mention constraint logic programming (CLP) [10] and SAT solvers [11], the latter specialized for the satisfiability of Boolean formulas.

While CSP methods are superior to MIP approaches for determining if a given problem has a feasible (integer) solution, the main drawback is their inefficiency for solving optimization, as they do not have the ability of MIP approaches to solve continuous relaxations (e.g., linear programming relaxations) of the problem in order to get upper and lower bounds to the optimum value.

For this reason, it seems extremely interesting to integrate the two approaches into one single solver. Some efforts have been done in this direction [12, 13, 14, 15, 16], showing that such mixed methods have a tremendous performance in solving mathematical programs with continuous (quantitative) and discrete (logical/symbolic) components, compared to MIP or CSP individually. Such successful results have stimulated also industrial interest: ILOG Inc., which is on of the worldwide leaders in software for combinatorial optimization, is currently distributing OPL (Optimization Programming Language), a modeling and programming language which allows the formulation and solution of optimization problems, using both MIP and CSP techniques, combining to some extent the advantages of both approaches.

At the light of the benefits and drawbacks of the previous work in [7,8,9] for solving control and stability/safety analysis problems for hybrid systems using MIP techniques, in this paper we follow a different route that uses a combined approach of MIP and CSP techniques. In particular, we focus on combinations of convex programming (e.g., linear, quadratic, etc.) for optimization over real

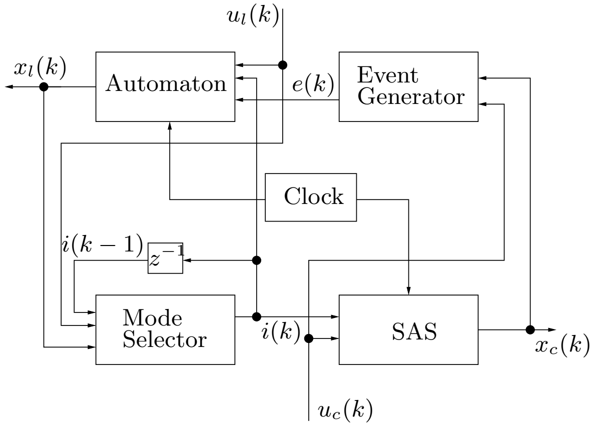

Fig. 1. Discrete-time hybrid system

variables, and of SAT-solvers for determining the satisfiability of Boolean formulas.

We build up a new modeling approach directly tailored to the use of a "hybrid" MIP+SAT solver for solving optimal control problems, and show its computational advantages over pure MIP methods. A preliminary work in this direction appeared in [17], where generic constraint logic programming (CLP) was used for handling the discrete part of the problem.

The paper is organized as follows. Discrete-time hybrid models are introduced in Section 2. In Section 3 the optimal control problem is formulated and in Section 4 it is reformulated in a suitable way for the combined MIP-CSP approach. Section 5 introduces the new solution algorithm and an example showing the benefits of this technique, compared to pure MIP approaches [7, 9] is shown in section 6.

2 Discrete-Time Hybrid Systems

Following the ideas of [9], a hybrid system can be modeled as the interconnection of an automaton (AUT) and a switched affine system (SAS) through an event generator (EG) and a mode selector (MS) (see Figure 1). The automaton describes the logic dynamics of the hybrid system, the SAS describes the continuous dynamics, the EG and MS describe the interactions between these dynamics.

2.1 Automaton

The discrete dynamics of a hybrid system can be modeled as an automaton (or finite state machine). We will only refer to "synchronous automata", where transitions are clocked and synchronous with the sampling time of the continuous dynamical equations. The adjective "synchronous" will be omitted for brevity. The automaton evolves according to the logic state update function

$$x_l(k+1) = f_l(x_l(k), u_l(k), e(k)), \tag{1a}$$

where $k \in \mathbb{Z}^+$ is the time index, $x_l \in \mathcal{X}_l \subseteq \{0,1\}^{n_l}$ is the logic state, $u_l \in \mathcal{U}_l \subseteq \{0,1\}^{m_l}$ is the exogenous logic input, $e \in \mathcal{E} \subseteq \{0,1\}^{n_e}$ is the endogenous input coming from the EG defined below in Section 2.3, and $f_l : \mathcal{X}_l \times \mathcal{U}_l \times \mathcal{E} \to \mathcal{X}_l$ is a deterministic Boolean function. An automaton can be represented as a directed graph (as in Figure 2, for instance). An automaton may also have a logic output

$$y_l(k) = g_l(x_l(k), u_l(k), e(k)), \tag{1b}$$

where $y_l \in \mathcal{Y}_l \subseteq \{0,1\}^{p_l}$, and $g_l : \mathcal{X}_l \times \mathcal{U}_l \times \mathcal{E} \to \mathcal{Y}_l$ is also a Boolean function. In the sequel, with a slight abuse of notation, we will refer to the codomain of Boolean functions both as $\{0,1\}$ and as $\{\texttt{FALSE},\texttt{TRUE}\}$. In the context of Boolean functions and formulas, the equal sign ($=$) should be interpreted as an if-and-only-if condition ($\longleftrightarrow$).

2.2 Switched Affine System

The continuous dynamics can be modeled by a switched affine system (SAS). A SAS is a collection of affine systems:

$$x_c(k+1) = A_{i(k)}x_c(k) + B_{i(k)}u_c(k) + f_{i(k)} \tag{2a}$$

$$y_c(k) = C_{i(k)}x_c(k) + D_{i(k)}u_c(k) + g_{i(k)}, \tag{2b}$$

where $x_c \in \mathcal{X}_c \subseteq \mathbb{R}^{n_c}$ is the continuous state vector, $u_c \in \mathcal{U}_c \subseteq \mathbb{R}^{m_c}$ is the exogenous continuous input vector, $y_c \in \mathcal{Y}_c \subseteq \mathbb{R}^{p_c}$ is the continuous output vector, $i(k) \in \mathcal{I} \triangleq \left\{\begin{bmatrix}1\ 0 \cdots 0\end{bmatrix}^T, \cdots, \begin{bmatrix}0 \cdots 0\ 1\end{bmatrix}^T\right\} \subseteq \{0,1\}^s$SAS is operating, $|\mathcal{I}| = s$ is the number of elements of $\mathcal{I}$, and $\{A_i, B_i, f_i, C_i, D_i, g_i\}_{i\in\mathcal{I}}$ is a collection of matrices of opportune dimensions. The mode $i(k)$ is generated by the mode selector, as described below in Section 2.4. A SAS of the form (2) preserves the value of the state when a switch occurs. However, resets can be modeled in the present discrete-time setting as detailed in [9].

2.3 Event Generator

An event generator is a mathematical object that generates a logic signal according to the satisfaction of a linear affine constraint:

$$[e_j(k) = 1] \longleftrightarrow [a_j^T x_c(k) + b_j^T u_c(k) \le c_j], \tag{3}$$

where the subscript j denotes the jth component of the vector, and $a_j \in \mathbb{R}^{n_c}$, $b_j \in \mathbb{R}^{m_c}$, $c_j \in \mathbb{R}$ define a linear guard (i.e., an hyperplane) in the space of continuous states and inputs.

2.4 Mode Selector

The dynamic mode $i(k)$ of the SAS, that we will also call the *active mode*, is selected through a *mode selector*

$$i(k) = f_{\mathrm{MS}}(x_l(k), u_l(k), i(k-1)), \tag{4}$$

where $f_{MS} : \mathcal{X}_l \times \mathcal{U}_l \times \mathcal{I} \to \mathcal{I}$ is a Boolean function of the logic state $x_l(k)$, of the logic input $u_l(k)$, and of the active mode $i(k-1)$ at the previous sampling instant. We say that a *mode switch* occurs at step k if $i(k) \neq i(k-1)$. Note that contrarily to continuous time hybrid models, where switches can occur at any time, in our discrete-time setting a mode switch can only occur at sampling instants.

3 Optimal Control

A finite-time optimal control problem for the class of hybrid systems introduced in the previous section can be formulated as follows:

$$\min_{\{x(k+1),u(k)\}_{k=0}^{T-1}} \sum_{k=0}^{T-1} \ell_k(x(k+1) - r_x(k+1), u(k) - r_u(k)) \tag{5a}$$

$$\text{s.t. dynamics } (1), (2), (3), (4) \tag{5b}$$

$$h_D(x(0), \{x(k+1), u(k), e(k), i(k)\}_0^{T-1}) \leq 0 \tag{5c}$$

$$h_A(x(0), \{x(k+1), u(k), e(k), i(k)\}_0^{T-1}) \leq 0 \tag{5d}$$

where T is the control horizon, $\ell_k : \mathbb{R}^{n \times m} \to \mathbb{R}$ is a nonnegative convex function, $n = n_c + n_l$, $m = m_c + m_l$, $r_x \in \mathbb{R}^n$, $r_u \in \mathbb{R}^m$ are given reference trajectories to be tracked by the state and input vectors, respectively.

The constraints of the optimal control problem can be classified in three different categories:

Dynamical constraints (5b) . These constraints represent the discrete-time hybrid system dynamics. They may also include other constraints such as saturation constraints on continuous input variables, that are embodied in the variable domain $\mathcal{U}_c$.

Design constraints (5c) . These are artificial constraints imposed by the designer to fulfill the required specifications. Examples of such constraints may be state limits

$$x_{min}(k) \leq x_c(k) \leq x_{max}(k), \;\; k = 1, \ldots, T,$$

where $x_{min}(k)$, $x_{max}(k)$ are bounds that the designer wants to impose on continuous states.

Ancillary constraints (5d). These constraints provide an a priori additional and auxiliary information for determining the optimal solution. They do not change the solution itself, rather help the solver by restricting the set of feasible combinations, and therefore the size of the decision tree in a branch a bound strategy. For example, one may pre-compute all possible mode transitions of the SAS dynamics using reachability analysis, and impose *reachability constraints* of the form $[\delta_h(k) = 1] \rightarrow [\delta_j(k+1) = 0]$ (or equivalently $\delta_h(k) + \delta_j(k+1) \leq 1$) for all $k = 0, \ldots, T-2$ whenever a transition from the hth mode to the jth mode is not possible.

4 Problem Reformulation

Problem (5) can be solved via MILP when the costs ℓ_k are convex piecewise linear functions, for instance $\ell_k(x,u) = \|Q_x x\|_\infty + \|Q_u u\|_\infty$, where Q_x, Q_u are full-rank matrices and $\|\cdot\|_\infty$ denotes the infinity-norm ($\|Qx\|_\infty = \max_{j=1,\ldots,n} |Q^j x|$, where Q^j is the j-th row of Q) [8], or via MIQP (mixed integer quadratic programming) when $\ell_k(x,u) = x' Q_x x + u' Q_u u$, where Q_x, Q_u are positive (semi)definite matrices [7].

Following a different route, in this paper we wish to solve problem (5) by using MIP and SAT techniques in a combined approach, taking advantage of SAT for dealing with the purely logic part of the problem. In order to do this, we need to reformulate the problem in a suitable way.
The automaton and mode selector parts of the hybrid system are described as a set of Boolean constraints so they do not require transformations. The event generator (2.3) can be equivalently expressed, by adopting the so-called "big-M" technique, as

$$(a_j^T x_c(k) + b_j^T u_c(k) - c_j) \leq M_j(1 - e_j(k)), \tag{6a}$$

$$(a_j^T x_c(k) + b_j^T u_c(k) - c_j) > m_j e_j(k), \tag{6b}$$

where $j = 1, \ldots, n_e$, M_j, m_j are upper and lower bounds, respectively, on $a_j^T x_c(k) + b_j^T u_c(k) - c_j$, and $e_j(k) \in \{0, 1\}$. From a computational viewpoint, it may be convenient to have a set of inequalities without strict inequalities. In this case we will follow the common practice [18] of replacing the strict inequality (6) as

$$(a_j^T x_c(k) + b_j^T u_c(k) - c_j) \geq \epsilon + (m_j - \epsilon) e_j(k), \tag{6c}$$

where ϵ is a small positive scalar, e.g., the machine precision, although the equivalence does not hold for $0 < (a_j^T x_c(k) + b_j^T u_c(k) - c_j) < \epsilon$ (i.e., for the numbers in the interval $(0, \epsilon)$ that cannot be represented in the machine). The continuous state update equation of the SAS dynamics (2) can be equivalently written as the combination of linear terms and *if-then-else* rules:

$$w_i(k) = \begin{cases} A_i x_c(k) + B_i u_c(k) + f_i & \text{if } (\delta_i = 1) \\ 0 & \text{otherwise} \end{cases} \tag{7a}$$

$$x_c(k+1) = \sum_{i=1}^{s} w_i(k) \tag{7b}$$

where $w_i(k) \in \mathbb{R}^{n_c}$, $i = 1, \ldots, s$. The output y_c of the SAS dynamics admits a similar transformation. The SAS representation (7) can be translated into a set of constraints by also using the big-M technique [18]:

$$-M_i^j \delta_i(k) + w_i(k) \leq 0, \tag{8a}$$

$$m_i^j \delta_i(k) - w_i(k) \leq 0, \tag{8b}$$

$$m_i^j (1 - \delta_i(k)) + w_i(k) \leq A_i^j x_c(k) + B_i^j u_c(k) + f_i^j, \tag{8c}$$

$$-M_i^j (1 - \delta_i(k)) - w_i(k) \leq -A_i^j x_c(k) - B_i^j u_c(k) - f_i^j, \tag{8d}$$

where M_i^j, m_i^j are upper and lower bounds on $A_i^j x_c(k) + B_i^j u_c(k) + f_i^j$, $\delta_i(k) \in \{0,1\}$, $w_i(k) \in \mathbb{R}^{n_c}$, $x_c \in \mathbb{R}_c^n$, $u \in \mathbb{R}_c^m$, ^{j} denotes the jth component or row, $j = 1, \ldots, n_c$, $i = 1, \ldots, s$, and k is the time index. Note that the vector of (0-1) variables $i(k) = [\delta_1(k) \ \ldots \ \delta_s(k)]' \in \{0,1\}^s$ is subject to the exclusive or condition

$$\delta_1(k) \oplus \delta_2(k) \oplus \ldots \oplus \delta_s(k) = \texttt{TRUE}. \tag{9}$$

By using the transformations into mixed integer inequalities described earlier, problem (5) can be cast as the mixed-integer convex program

$$\min_{\substack{\{x(k+1), u(k),\\ w(k), \delta(k)\}\\ k=0,\ldots,T-1}} \sum_{k=0}^{T-1} \ell_k(x(k+1) - r_x(k+1), u(k) - r_u(k)) \tag{10a}$$

$$\text{s.t. } Ax_c(k) \leq b, \ x_c(k+1) = \sum_{i=1}^{s} w_i(k) \tag{10b}$$

$$M_1 x_c(k) + M_2 u_c(k) + M_3 w(k) \leq M_4 e(k) + M_5 \delta(k) + M_6 \tag{10c}$$

$$g(x_l(k+1), x_l(k), u_l(k), e(k), \delta(k)) = \texttt{TRUE} \tag{10d}$$

$$w(k) = [w_1(k) \ldots w_s(k)]', \ w_i(k) \in \mathbb{R}^{n_c}, \ \delta(k) \in \{0,1\}^s,$$

where $\{x_c(k+1), u_c(k), w(k)\}_{k=0}^{T-1}$ are the continuous optimization variables, $\{x_l(k+1),\ u_l(k),\ \delta(k),\ e(k)\}_{k=0}^{T-1}$ are the binary optimization variables, $x_c(0)$, $x_l(0)$ is a given initial state, constraints (10b), (10c) represent the EG and SAS parts (6a), (6c), (7b), (8), and the purely continuous or mixed constraints from (5c), (5d), while (10d) represents the automaton (1a), the mode selector (4), possible purely Boolean constraints from (5c), (5d), as well as the exclusive or condition (9). Matrices M_i, $i = 1 \ \ldots \ 6$, are obtained by the big-M representations (6) and (8).

Problem (10) belongs to the following general class of *mixed logical/convex* problems:

$$\min_{z,\nu,\mu} f(z) \tag{11a}$$

$$\text{s.t. } g_c(x_c(0), z) \le 0,\ h_c(x_c(0), z) = 0 \qquad \text{(Continuous constraints)} \tag{11b}$$

$$g_m(x_c(0), x_l(0), z, \mu) \le 0,\ h_m(x_c(0), x_l(0), z, \mu) = 0 \qquad \text{(Mixed constraints)} \tag{11c}$$

$$g_L(x_l(0), \nu, \mu) = \texttt{TRUE} \qquad \text{(Logic constraints)} \tag{11d}$$

$$z \in \mathbb{R}^{n_z},\ \nu \in \{0,1\}^{n_\nu},\ \mu \in \{0,1\}^{n_\mu}$$

where $g_c : \mathbb{R}^{n_z} \to \mathbb{R}^{q_{gc}}$, $g_m : \mathbb{R}^{n_z+n_\mu} \to \mathbb{R}^{q_{gm}}$ are convex functions, $h_c : \mathbb{R}^{n_z} \to \mathbb{R}^{q_{hc}}$, $h_m : \mathbb{R}^{n_z+n_\mu} \to \mathbb{R}^{q_{hm}}$ are affine functions, and $g_L : \{0,1\}^{n_\nu \times n_\mu} \to \{0,1\}^{n_{CP}}$ is a Boolean function. In the hybrid optimal control problem at hand, z collects all the continuous variables ($x_c(k+1)$, $u_c(k)$, $k = 0, \ldots, T-1$), the auxiliary variables needed for expressing the SAS dynamics, possibly slack variables for upper bounding the cost function in (10a) [8], μ collects the integer variables that appear in mixed constraints ($e(k)$, $\delta_i(k)$, $k = 0, \ldots, T-1$, $i = 1, \ldots, s$), and ν collects the integer variables such as $x_l(k)$, $u_l(k)$ that only appear in logic constraints. Note that in general if the objective function in the the form $f(z, \mu)$ we could consider the new objective function ϵ, $\epsilon \in \mathbb{R}$, and an additional constraint $f(z, \mu) \le \epsilon$ which is a mixed convex constraint that could be included in (11c).

5 SAT-Based Branch and Bound

5.1 Constraint Satisfaction and Optimization

CSP and optimization are similar enough to make their combination possible, and yet different enough to make it profitable. Optimization is primarily associated with mathematics and engineering, while CSP was developed (more recently) in the computer science and artificial intelligence communities. The two fields evolved more or less independently until a few years ago. Yet they have much in common and are applied to solve similar problems. Most importantly for the purposes of this paper, they have complementary strengths, and the last few years have seen growing efforts to combine them [13, 12, 19, 14, 20].

The recent interaction between CSP and optimization promises to affect both fields. In the following subsections we illustrate an approach for merging them into a single problem-solving technology, in particular by combining convex optimization and satisfiability of Boolean formulas (SAT).

Convex Optimization. Convex optimization is very popular in engineering, economics, and other application domains for solving nontrivial decision problems. Convex optimization includes linear, quadratic, and semidefinite programming, for which several extremely efficient commercial and public domain solvers are nowadays available. An excellent reference to convex optimization is the book by Boyd and Vandenberghe [21].

SAT Problems. An instance of a satisfiability (SAT) problem is a Boolean formula that has three components:

- A set of n variables: $x_1, x_2, \ldots, x_n$.
- A set of literals. A literal is a variable ($Q = x$) or a negation of a variable ($Q = \neg x$).
- A set of m distinct clauses: $C_1, C_2, \ldots, C_m$. Each clause consists of only literals combined by just logical *or* ($\vee$) connectives.

The goal of the satisfiability problem is to determine whether there exists an assignment of truth values to variables that makes the following Conjunctive Normal Form (*CNF*) formula satisfiable:

$$C_1 \wedge C_2 \wedge \ldots \wedge C_m,$$

where $\wedge$ is a logical *and* connective. For a survey on SAT problems and related solvers the reader is referred to [11].

5.2 A SAT-Based "Hybrid" Algorithm

The basic ingredients for an integrated approach are (1) a solver for convex problems obtained from relaxations over continuous variables of mixed integer convex programming problems, and (2) a SAT solver for testing the satisfiability of Boolean formulas. The relaxed model is used to obtain a solution that satisfies the constraint sets (11b) and (11c) and optimizes the objective function (11a). The optimal solution of the relaxation may fix some of the (0-1) variables to either 0 or 1. If all the (0-1) variables in the relaxed problem have been assigned (0-1) values, the solution of the relaxation is also a feasible solution of the mixed integer problem. More often, however, some of the (0-1) variables have fractional parts, so that further "branching" and solution of further relaxations is necessary. To accelerate the search of feasible solutions one may use the fixed (0-1) variables to "infer" new information on the other (0-1) variables by solving a SAT problem obtained by constraint (11d). In particular, when an integer solution of μ is found from convex programming, a SAT problem then verifies whether this solution can be completed with an assignment of ν that satisfies (11d).

The basic branch&bound (B&B) strategy for solving mixed integer problems can be extended to the present "hybrid" setting where both convex optimization and SAT solvers are used. In a B&B algorithm, the current best integer solution is updated whenever an integer solution with an even better value of the objective function is found. In the hybrid algorithm at hand an additional SAT problem is solved to ensure that the integer solution obtained for the relaxed problem is feasible for the constraints (11d) and to find an assignment for the other logic variables ν that appear in (11d). It is only in this case that the current best integer solution is updated.

The B&B method requires the solution of a series of convex subproblems obtained by branching on integer variables. Here, the non-integer variable to branch on is chosen by selecting the variable with the largest fractional part (i.e., the one closest to 0.5), and two new convex subproblems are formed with that variable fixed at 0 and at 1, respectively. When an integer feasible solution of the relaxed problem is obtained, a satisfiability problem is solved to complete the solution. The value of the objective function for an integer feasible solution of the whole problem is an upper bound (UB) of the objective function, which may be used to rule out branches where the optimum value attained by the relaxation is larger than the current upper bound.

Let P denote the set of convex and SAT subproblems to be solved. The proposed SAT-based B&B method can be summarized as follows:

1. **Initialization.** $UB = \infty$, $P = \{(p^0, SAT^0)\}$. The convex subproblem p^0 is generated by using (11a),(11b), (11c) along with the relaxation $\mu \in [0,1]^{n_\mu}$, and the SAT subproblem SAT^0 is generated by using (11d).
2. **Node selection.** If $P = \emptyset$ then go to 7.; otherwise select and remove a (p, SAT) problem from the set P; The criterion for selecting a problem is called *node selection rule.*
3. **Logic inference.** Solve problem SAT. If it is infeasible go to step 2.
4. **Convex reasoning**. Solve the convex problem p, and:
   4.1. If the problem is infeasible or the optimal value of the objective function is greater than UB then go to step 2.
   4.2. If the solution is not integer feasible then go to step 6.
5. **Bounding.** Let $\mu^* \in \{0,1\}^{n_\mu}$ be the integer part of the optimal solution found at step 4.; to extend this partial solution, solve the SAT problem finding ν such that $g(\nu, \mu^*)$ =`TRUE`. If the SAT problem is feasible then update UB; otherwise add to the LP problems of the set P the "no-good" cut [12]
$$\sum_{i \in T^*} \mu_i - \sum_{j \in F^*} \mu_j \leq B^* - 1,$$
where $T^* = \{i|\mu_i^* = 1\}$, $F^* = \{j|\mu_j^* = 0\}$, and $B^* = |T^*|$. Go to step 2.
6. **Branching.** Among all variables that have fractional values, select the one closest to 0.5. Let μ_i be the selected non-integer variable, and generate two subproblems $(p \cup \{\mu_i = 0\},\ SAT\&\{\neg\mu\})$, $(p \cup \{\mu_i = 1\},\ SAT\&\{\mu\})$ and add them to set P; go to step 2.
7. **Termination.** If $UB = \infty$, then the problem is infeasible. Otherwise, the optimal solution is the current value UB.

Remark 1. At each node of the search tree the algorithm executes a three-step procedure: logic inference, solution of the convex relaxation, and branching. The first step and the attempted completion of the solution do not occur in MIP approaches but they are introduced here by the distinction of mixed (0-1) variables μ and pure (0-1) variables ν. The logic inference and the attempted completion steps do not change the correctness and the termination of the algorithm but

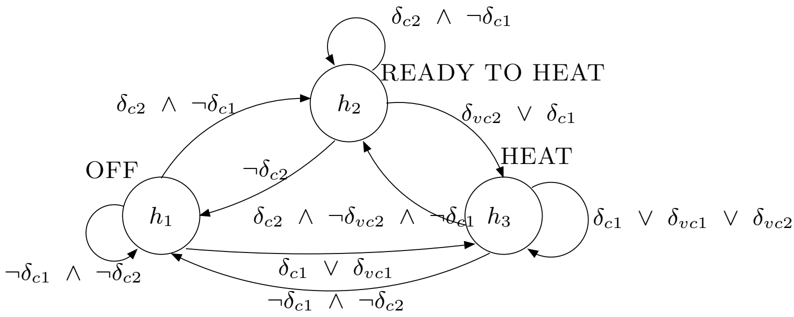

Fig. 2. Automaton regulating the heater

they improve the performance of the algorithm because of the efficiency of the SAT solver in finding a feasible integer solution.

6 Numerical Results

In this section we show on an example of hybrid optimal control problem that the hybrid solution technique described in the previous sections has a better performance compared to commercial MIP solvers.

6.1 Hybrid Model

Consider a room with two bodies with temperatures T_1, T_2 and let T_{amb} be the room temperature (this example is an extension of the example reported in [22]). The room is equipped with a heater, close to body 1, delivering thermal power u_{hot} and an air conditioning system, close to body 2, draining thermal power u_{cold}. These are turned on/off according to some rules dictated by the closeness of the two bodies to each device. We want guarantee that the bodies are not cold or hot.

The discrete-time continuous dynamics of each body is described by the difference equation

$$\frac{T_i(k+1) - T_i(k)}{T_s} = -\alpha_i(T_i(k) - T_{amb}) + k_i(u_{\text{hot}}(k) - u_{\text{cold}}(k)) + cu_e(k), \tag{12}$$

where $i = 1, 2$, α_i, k_i, c are suitable constants, T_s is the sampling time, and $u_e(k)$ is an exogenous input that can be used to deliver or drain thermal power manually (e.g. by opening a window or by changing the water flow from a centralized heating system).

The automaton part of the system is described by the two automata represented in Figures 2 and 3, where δ_{ci},δ_{vci},γ_{hi} and γ_{vhi}, for $i = 1, 2$, are logic variables defined as follows

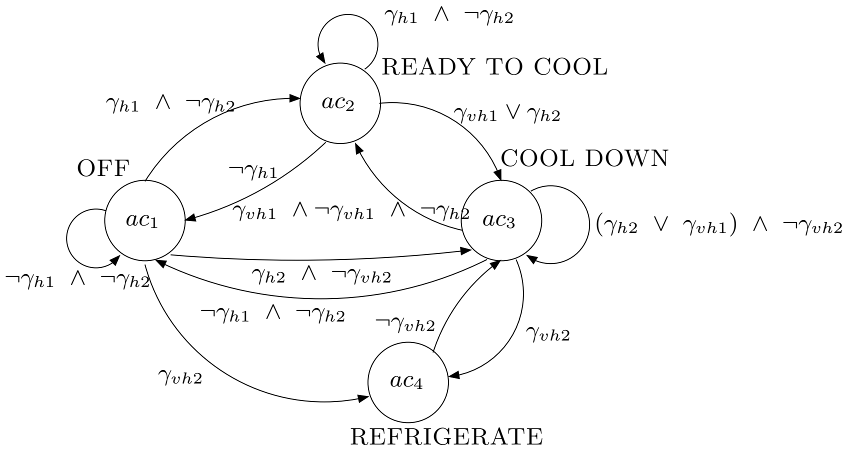

Fig. 3. Air conditioning system automaton

$$[\delta_{vci}(k) = 1] \longleftrightarrow [T_i(k) \le T_{vci}], \tag{13a}$$
$$[\delta_{ci}(k) = 1] \longleftrightarrow [T_i(k) \le T_{ci}], \tag{13b}$$
$$[\gamma_{hi}(k) = 1] \longleftrightarrow [T_i(k) \ge T_{hi}], \tag{13c}$$
$$[\gamma_{vhi}(k) = 1] \longleftrightarrow [T_i(k) \ge T_{vhi}], \tag{13d}$$

and where $T_{vci} \le T_{ci} \le T_{hi} \le T_{vhi}$ are constant thresholds. The automaton for the heater (Figure 2) sets the heater in the "ready to heat" state if body 2 is cold, and will go in "heat" state if body 2 is very cold. If body 1 is cold or very cold the heater is turned on immediately. The automaton of the air conditioning (A/C) system (Figure 3) sets the air conditioning system in the "ready to cool" state if body 1 is hot, unless body 2 is cold, in other words, the A/C system is turned on only when body 1 is very hot. However, the draining thermal power is half of the full power. The A/C system is set to the maximum power if the body 2 is very hot but it is immediately switched to half power as soon as body 2 is only hot (due to energy consumptions of the A/C system).

The heater delivers thermal power and the A/C system drains thermal power according to the following rules:

$$u_{\text{hot}} = \begin{cases} u_H & \text{if } h_3 = 1 \\ 0 & \text{otherwise} \end{cases} \qquad u_{\text{cold}} = \begin{cases} u_C & \text{if } ac_4 = 1 \\ \frac{u_C}{2} & \text{if } ac_3 = 1 \\ 0 & \text{otherwise} \end{cases}. \tag{14}$$

By following the notation of (1), we have $x_l = [h_1\ h_2\ h_3\ ac_1\ ac_2\ ac_3\ ac_4]' \in \{0,1\}^7$, $u_l = \emptyset$ and $e(k) = [\delta_{vc1}\ \delta_{vc2}\ \delta_{c1}\ \delta_{c2}\ \gamma_{h1}\ \gamma_{h2}\ \gamma_{vh1}\ \gamma_{vh2}]' \in \{0,1\}^8$.

The system has six modes: $(u_{\text{hot}}, u_{\text{cold}}) \in \{(0,0),(u_H,0),(0,u_C),(0,u_C/2),(u_H,u_C),(u_H,u_C/2)\}$. The mode selector function is defined as follows

$$i(k) = \begin{bmatrix} \neg h_3(k) \wedge \neg ac_4(k) \wedge \neg ac_3(k) \\ h_3(k) \wedge \neg ac_4(k) \wedge \neg ac_3(k) \\ \neg h_3(k) \wedge ac_4(k) \wedge \neg ac_3(k) \\ \neg h_3(k) \wedge \neg ac_4(k) \wedge ac_3(k) \\ h_3(k) \wedge ac_4(k) \wedge \neg ac_3(k) \\ h_3(k) \wedge \neg ac_4(k) \wedge ac_3(k) \end{bmatrix} \in \{0,1\}^6,$$

which only depends on logic states.

The SAS dynamics (12), i.e., the continuous part of the hybrid system, is translated into a set of inequalities using (8), which provides the set of constraints

$$Ax_c(k) + Bu_c(k) + Cw(k) \le D\delta(k) + E, \tag{15}$$

where $x_c = [T_1 \; T_2]'$, $u_c = u_e$, $w(k) \in \mathbb{R}^3$ contains the auxiliary continuous variables needed to represent the conditions $u_{\text{hot}} = u_H$, $u_{\text{cold}} = u_C$, $u_{\text{cold}} = u_C/2$, and $\delta(k) = [h_3(k) \; ac_3(k) \; ac_4(k)] \in \{0,1\}^3$. Constraints (15) are obtained by employing the HYSDEL compiler [9].

Finally, the event generator is represented by (13a) and (13b). These are translated by HYSDEL into a set of linear inequalities using (6):

$$G'_x x_c(k) + G'_u u_c(k) + D'e(k) \le E', \tag{16}$$

where $e(k) = [\delta_{vc1} \; \delta_{vc2} \; \delta_{c1} \; \delta_{c2} \; \gamma_{h1} \; \gamma_{h2} \; \gamma_{vh1} \; \gamma_{vh2}]' \in \{0,1\}^8$.

6.2 Optimal Control Problem

The goal is to design an optimal control profile for the continuous input u_e that minimizes $\sum_{k=0}^{T} |T_i(k) - T_{amb}|$ subject to the hybrid dynamics and the following additional constraints:

- Continuous constraints on temperatures to avoid that they assume unacceptable values

$$-10 \le T_1(k) \le 50 \qquad\qquad -10 \le T_2(k) \le 50. \tag{17a}$$

  These constraints may be interpreted as dynamical constraints due to physical limitations of the bodies.
- A continuous constraint on exogenous input to avoid excessive variations:

$$-10 \le u_e(k) \le 10. \tag{18}$$

  This constraint may be interpreted as a design constraint of the form (5c).

The above dynamics and constraints are also modeled in HYSDEL [9] to obtain an MLD model of the hybrid system in order to compare the performance achieved by the hybrid solver with the one obtained by employing a pure MILP approach.

The optimal control problem is defined over horizon of T steps as:

$$\min_{\{x,u,z,\delta,\epsilon_T\}} \sum_{k=0}^{T-1} \epsilon_T(k) \tag{19a}$$

$$\text{s.t. } \epsilon_T(k) \begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix} \geq \pm(T_i(k) - T_{amb}), \tag{19b}$$

$$\text{automata Figures 2, 3,} \tag{19c}$$

$$(15),\ (16) \tag{19d}$$

$$(17),\ (18) \tag{19e}$$

where $\{x, u, z,\delta, \epsilon_T\}$=$\{x(k), u(k),z(k), \delta(k),\ \epsilon_T(k)\}_{k=0}^{T-1}$, ϵ_T =$[\epsilon_{T1}(0),\ \epsilon_{T2}(0),\ldots,$ $\epsilon_{T1}(T-1),\ \epsilon_{T2}(T-1)]' \in \mathbb{R}^{2T}$.

Each part of the optimal control problem is managed by either the SAT solver or the LP solver: the cost function (19a), the inequalities (19b), (19d), and the additional constraints (19e) are managed by the LP solver, the logic part (19c) is managed by the SAT solver. In our simulations we have used, respectively, zCHAFF [23] for SAT and CPLEX [24] for LP.

In all our simulations we have adopted depth first search as the *node selection rule*, to reduce the amount of memory used during the search.

For the initial condition $T_1(0) = 5°$ C, $T_2(0) = 2°$ C and for $T_{amb} = 25°$ C we have done simulations for different horizons (the obtained optimal solution is clearly the same both using the SAT-based B&B and the MILP), reported in Table 1.

We can see that the performance of the SAT-based B&B is always better than the one obtained via MILP. The main reason is that the SAT B&B algorithm solves a much smaller number of LPs than the MILP solver. The "cuts" performed by the SAT solver, i.e. the infeasible SAT problems, obtained at step 3 of the algorithm turn out very useful to exclude subtrees containing no integer feasible solution. Moreover, the time spent for solving an integer feasibility problem described as SAT problem is much smaller than solving a pure integer problem, see Table 2. We can also see from Table 1 that the number of feasible SAT solved equals the number of LP solved plus one. This one more SAT is used to complete a feasible solution and it turns out very useful to further reduce the computation time.

The results were simulated on a PC Pentium IV 1.8 GHz running CPLEX 8.1 and zCHAFF 2003.07.22.

7 Conclusions

In this paper we have proposed a new unifying framework for MIP and CSP techniques based on the integration of convex programming and SAT solvers for

Table 1. Optimal control solution: comparison between pure MILP (CPLEX) and SAT-based B&B

T	Int Vars	MILP (s)	MILP LPs	SATbB&B (s)	SATbB&B LPs	SATbB&B SATs	SATbB&B "cuts"
5	75	0.04	60	0.04	15	16	0
10	150	0.22	119	0.38	15	16	0
15	225	0.61	152	0.66	17	18	2
20	300	1.452	248	1.011	17	18	3
25	375	2.594	301	1.512	19	20	2
30	450	4.307	363	2.093	20	21	4
35	525	5.729	367	2.844	20	21	5
40	600	12.058	486	3.505	27	28	9
45	675	13.479	534	4.367	31	32	7
50	750	19.108	607	5.368	43	44	8

Table 2. Computation time for solving a pure integer feasibility problem: comparison between the SAT (zCHAFF) and MILP (CPLEX)

T	Int. Vars	Constraints	SAT (s)	MILP (s)
5	75	460	0	0.03
10	150	920	0.01	0.03
15	225	1380	0.02	0.04
20	300	1840	0.03	0.04
25	375	2300	0.04	0.05
30	450	2760	0.05	0.06
35	525	3220	0.06	0.08
40	600	3680	0.08	0.11
45	675	4140	0.08	0.15
50	750	4600	0.08	0.18

solving optimal control problems for discrete-time hybrid systems. The approach consists of a logic-based branch and bound algorithm, whose performance in terms of computation time is superior in comparison to more standard mixed-integer programming techniques, as we have illustrated on an example.

Ongoing research is devoted to the improvement of the logic-based method by including relaxations of the automaton and MS parts of the hybrid system in the convex programming part, to the investigation of alternative relaxations of the SAS dynamics that are tighter than the big-M method, to the use of SAT solvers for also performing domain reduction (cutting planes), and to the use of the SAT-based B&B algorithm for reachability analysis and for efficiently converting discrete-time hybrid systems to an equivalent piecewise affine form.

Acknowledgment. This research was supported by the European Union through project IST-2001-33520 "CC-Computation and Control".

References

1. H. Witsenhausen. A class of hybrid-state continuous-time dynamic systems. *IEEE Trans. Automatic Control*, 11(2):161–167, 1966.
2. P.J. Antsaklis. A brief introduction to the theory and applications of hybrid systems. *Proc. IEEE, Special Issue on Hybrid Systems: Theory and Applications*, 88(7):879–886, July 2000.
3. C.G. Cassandras, D.L. Pepyne, and Y.Wardi. Optimal control of a class of hybrid systems. *IEEE Trans. Automatic Control*, 46(3):3981–415, 2001.
4. X. Xu and P.J. Antsaklis. An approach to switched systems optimal control based on parameterization of the switching instants. In *Proc. IFAC World Congress*, Barcelona, Spain, 2002.
5. B. Lincoln and A. Rantzer. Optimizing linear system switching. In *Proc. 40th IEEE Conf. on Decision and Control*, pages 2063–2068, 2001.
6. F. Borrelli, M. Baotic, A. Bemporad, and M. Morari. An efficient algorithm for computing the state feedback optimal control law for discrete time hybrid systems. In *Proc. American Contr. Conf.*, Denver, Colorado, 2003.

7. A. Bemporad and M. Morari. Control of systems integrating logic, dynamics, and constraints. *Automatica*, 35(3):407–427, March 1999.
8. A. Bemporad, F. Borrelli, and M. Morari. Piecewise linear optimal controllers for hybrid systems. In *Proc. American Contr. Conf.*, pages 1190–1194, Chicago, IL, June 2000.
9. F.D. Torrisi and A. Bemporad. HYSDEL - A tool for generating computational hybrid models. *IEEE Transactions on Control Systems Technology*, 2003. To appear.
10. K. Marriot and P.J. Stuckey. *Programming with constraints: an introduction.* MIT Press, 1998.
11. J. Gu, P.W. Purdom, J. Franco, and B. Wah. Algorithms for the satisfiability (SAT) problem: A survey. In *DIMACS Series on Discrete Mathematics and Theoretical Computer Science*, number 35, pages 19–151. American Mathematical Society, 1997.
12. J. Hooker. *Logic-based methods for Optimization.* Wiley-Interscience Series, 2000.
13. A. Bockmayr and T. Kasper. Branch and infer: A unifying framework for integer and finite domain constraint programming. *INFORMS Journal on Computing*, 10(3):287–300, Summer 1998.
14. R. Rodosek, M. Wallace, , and M. Hajian. A new approach to integrating mixed integer programming and constraint logic programming. *Annals of Oper. Res.*, 86:63–87, 1997.
15. F. Focacci, A. Lodi, and M. Milano. Cost-based domain filtering. In *J. Jaffar, editor, Principle and Practice of Constraint Programming*, volume 1713 of *Lecture Notes in Computer Science*, pages 189–203. Springer Verlag, 2001.
16. I. Harjunkoski, V. Jain, , and I.E. Grossmann. Hybrid mixed-integer/constraint logic programming strategies for solving scheduling and combinatorial optimization problems. *Comp. Chem. Eng.*, 24:337–343, 2000.
17. A. Bemporad and N. Giorgetti. A logic-based hybrid solver for optimal control of hybrid systems. In *Proc. 43th IEEE Conf. On Decision and Control*, Maui, Hawaii, USA, Dec. 2003.
18. H.P. Williams. *Model Building in Mathematical Programming.* John Wiley & Sons, Third Edition, 1993.
19. G. Ottosson. *Integration of Constraint Programming and Integer Programming for Combinatorial Optimization.* PhD thesis, Computing Science Department, Information Technology, Uppsala University, Sweden, 2000.
20. E. Tsang. *Foundations of Constraint Satisfaction.* Academic Press, 1993.
21. S. Boyd and L. Vandenberghe. *Convex Optimization.* In press., 2003. `http://www.stanford.edu/\char126\relaxboyd/cvxbook.html`.
22. A. Bemporad. Efficient conversion of mixed logical dynamical systems into an equivalent piecewise affine form. *IEEE Trans. Automatic Control*, 2003. In Press.
23. M. Moskewicz, C. Madigan, Y. Zhao, L. Zhang, and S. Malik. Chaff: Engineering an efficient sat solver. In *39th Design Automation Conference*, June 2001. `http://www.ee.princeton.edu/\char126\relaxchaff/zchaff.php`.
24. ILOG, Inc. *CPLEX 8.1 User Manual.* Gentilly Cedex, France, 2002.