Optimization of Discrete Event Systems using Extended Finite Automata and Mixed-Integer Nonlinear Programming

Carl Thorstensson * **Sathyamyla Kanthabhabhajeya** *
Bengt Lennartson * **Petter Falkman** *

* *Department of Signals and Systems, Chalmers University of Technology, Gothenburg, Sweden (e-mail: carl.thorstensson@chalmers.se)*

Abstract:
This paper presents a concept for converting a discrete event model, modeled with Extended Finite Automata (EFA), to mixed-integer linear constraints. The conversion handles the structure of modular EFAs, synchronization of EFAs using shared events and EFA execution order due to logical transition conditions. The paper also presents methods to reduce the number of variables and constraints by automatically analyzing the EFA model and the resulting problem formulation. An example of this is the special case of transition conditions used to model mutual exclusion of shared resources, where the conversion results in a significantly reduced problem formulation. The objective function is then built by summarizing weighted state cost functions and the result is a Mixed-Integer Nonlinear Programming problem. The main contribution of this paper is hence the combination of the simplicity in modeling a system with EFAs and an efficient formulation of the optimization problem that can be solved by standard optimization software.

Keywords: MILP, MINLP, Optimization, Automata, EFA, Scheduling, Discrete event systems

1. INTRODUCTION

Optimization is a way to utilize available resources in an efficient way and to achieve a maximum possible outcome. A variety of methods and tools have been developed to aid decision makers in their work to increase productivity, sustainability and efficiency as well as minimizing costs.

Optimization of discrete event systems has been introduced as a tool for guiding decision makers in designing for instance cycle-time optimal schedules, see Liljenvall [1998]. The most common way to model a discrete event system is to use different types of automata, and one optimization technique has been to find the optimal path by using heuristic graph search algorithms, see Hart et al. [2007]. Several search algorithms for solving job-shop scheduling problem using timed automata have also been introduced, see Abdeddaimi and Maler [2001]. Scheduling based on timed automata has, however, been shown to often be more beneficial using mixed-integer linear programming (MILP) solvers, compared to graph search algorithms, see Kobetski et al. [2006], although the efficiency depends on the characteristics of the system.

Methods to convert timed automata to MILP problems have been developed, see Panek et al. [2006]. A more complete framework for translating a general finite automata model with times to MILP was recently presented by Kobetski and Fabian [2009], where both event synchronization and the most common types of specifications were included in the optimization formulation. MILP has also been used for optimization of hybrid automata, see Bayen and Tomlin [2003] and Bemporad et al. [2005]. Comparing these results with the ones in Kobetski and Fabian [2009] and this paper, the latter are more focused on the complexity of the discrete part.

To decrease the gap between the event based automata theory in academia and the signal based modeling in industry, a finite automaton extended with variables has been introduced by Sköldstam et al. [2007]. The automaton is called Extended Finite Automaton (EFA) and is a powerful framework for modeling for instance shared resources or zones, which is a common situation in many industrial applications. EFAs are similar to Extended Finite State Machines (EFSMs), see Mannani et al. [2006], with the important difference that guards and actions are associated with the events in EFSMs. In EFAs, they are associated with transitions on individual automata, which is more flexible.

The simplified and compact modeling using EFAs has, in this paper, been combined with the strength of existing solvers for mixed-integer linear and nonlinear programming problems. The result is a concept that makes it easier and more straight forward to formulate and solve an optimization problem for a discrete event system. The concept can be divided into two parts; expressing the mixed-integer linear constraints based on the EFA model and to formulate an objective function from the optimization criterion of the system.

The first part of the paper presents the concept of converting EFAs into constraints. It covers conversion of the EFA structure, the synchronous behavior due to shared events and effects from using variables in guards and actions. As a guidance in formulating the objective function, the second part of the paper introduces general state cost functions, available for each location in the EFAs. The state cost functions model the costs to visit a location as a nonlinear function of the time spent in the location and can be used to formulate multi-criteria optimization problems with nonlinear objective functions. This kind of optimization problem is presented further in Miettinen [1998] and methods for solving it are discussed in Fonseca and Fleming [1995]. In this paper it is shown how cost functions can be obtained optimizing cycle time but also energy consumption for industrial robot systems. According to Vergnano et al. [2010] the energy cost is then expressed as a function of the execution times for the individual robot operations.

To summarize, the compact and industrially adapted framework of EFA modeling is shown to be a suitable bridge between discrete event modeling and linear programming. A brief complexity analysis also shows that the approach scales well when there are many parallel sequences without too many alternatives. This is a typical scenario for instance in robot cells, where most often only a few resources are shared among machines and devices.

2. PRELIMINARIES

This paper is built on combining Extended Finite Automata with Mixed-Integer Nonlinear Programming, and the theoretical base for these two topics are presented here.

2.1 Extended Finite Automata

The general definition of an Extended Finite Automaton is:

Definition 1. An Extended Finite Automaton (EFA) is a 7-tuple

$$E = \langle Q \times V, \Sigma, \mathscr{G}, \mathscr{A}, \rightarrow, (q_0, v_0), M \rangle$$

where the set $Q \times V$ is the extended finite set of states, Q is a finite set of *locations* and V is the finite domain of definition of the variables, Σ is a nonempty finite set of events (the alphabet), $\mathscr{G}$ is a set of guard predicates over V, $\mathscr{A}$ is a collection of action functions, $\rightarrow \subseteq Q \times \Sigma \times \mathscr{G} \times \mathscr{A} \times Q$ is a state transition relation, $(q_0, v_0) \in Q \times V$ is the initial state, and $M \subseteq Q \times V$ is a set of marked desired states c.f Sköldstam et al. [2007].

In this paper, the limitation of not allowing an empty set of events is relaxed, since events are not crucial for the formulation of the mixed-integer linear constraints. Introduction of events is only needed for modeling a synchronized behavior between multiple automata. It is also assumed that all EFAs are action consistent, meaning that they do not have contradicting actions, see Sköldstam et al. [2007].

The state transition relation of an EFA can be written as $(p, v) \xrightarrow{\sigma} (q, v')$, where $p, q \in Q$, $v \in V$, v' is the next value of v after the transition and $\sigma \in \Sigma$. This is extended to strings in Σ^* in the following recursive way:

$$\begin{aligned} &(p,v) \xrightarrow{\varepsilon} (p,v) \;\text{ for all }\; (p,v) \in Q \times V, \\ &(p,v) \xrightarrow{s\sigma} (q,v') \;\text{ if }\; (p,v) \xrightarrow{s} (r,y) \;\text{ and} \\ &(r,y) \xrightarrow{\sigma} (q,v') \;\text{ for some }\; (r,y) \in Q \times V \end{aligned}$$

2.2 Mixed-Integer Nonlinear Programming

Linear programming is the procedure of finding an optimal value of an objective function that is linear due to a set of real variables, where the optimum may not conflict a given set of linear constraints, see Dantzig [1963]. It is a commonly used part of optimization and can be used to a variety of situations. A limitation in linear programming is however that it does not allow alternatives between constraints or in the objective function, i.e. constraints formulated on the form

$$\bigvee_{i=1}^{N} X_i \tag{1}$$

where X_i is a linear constraint, e.g. $x_i \geq x_j$. The constraints in (1) can however be expressed by introducing binary decision variables (BDV), $\gamma_i \in \mathbb{Z}_{[0,1]}$ and a sufficiently large constant $M \in \mathbb{R}$. Equation (1) can then be reformulated as

$$\begin{aligned} x_i &\geq x_j - (1 - \gamma_i)M \\ \sum_{i=1}^{M} \gamma_i &= 1 \end{aligned} \tag{2}$$

When $\gamma_i = 1$, (2) is reduced to the original constraint, $x_i \geq x_j$, and if $\gamma_i = 0$, the expression becomes $x_i \geq x_j - M$, which will not constrain variable x_i if M is sufficiently large. An optimization formulation on this form is usually referred to as a Mixed-Integer Linear Programming (MILP) problem, presented in Schrijver [1986]. It is a common approach in scheduling, where there might exist multiple paths to choose between. To formulate a more sophisticated criterion, the objective function can instead be a general nonlinear function $F(\mathbf{x})$. This leads to the more general optimization problem formulated as

$$\begin{aligned} \mathit{min} \quad & F(\mathbf{x}) \\ \mathit{s.t.} \quad & A\mathbf{x} \leq \mathbf{b} \\ & \mathbf{x} \in \mathbb{R}_{\geq 0} \end{aligned} \tag{3}$$

where $A \in \mathbb{R}^{n \times m}$ is the constraints matrix for the mixed-integer linear constraints, $\mathbf{b} \in R^n$ is the constraints vector of constants and $\mathbf{x} \in R^m$ is the set of decision variables which can be divided into two sets as $\mathbf{x} = [\mathbf{x}_{\mathbb{R}} \quad \mathbf{x}_{\mathbb{Z}}]$ where $\mathbf{x}_{\mathbb{R}} \in \mathbb{R}^r$ is a vector of all real variables and $\mathbf{x}_{\mathbb{Z}} \in \mathbb{Z}^{m-r}$ is a vector of all integer variables in $\mathbf{x}$.

3. CONVERTING EFA TO MIXED-INTEGER LINEAR CONSTRAINTS

A method to automatically formulate Mixed-Integer Linear Constraints (MILC) for a discrete event system, modeled by EFAs is presented in this section. The conversion from EFAs to MILC is divided into,

(1) EFA structure
(2) Shared events (synchronization)
(3) Logical transition conditions (guards and actions)
(4) Resource allocation for shared resources

Two limitations of this work are that loops in EFAs are ignored and the logical transition conditions are limited to only include the basic functionality. These limitations are further discussed in Section 3.1 and 3.3. All steps are performed separately and they only depend on previous steps. This simplifies later extensions of the concept to include more aspects of modeling with EFAs. Extensions could e.g. be to relax some of the limitations introduced on the EFAs in this paper. The real and integer variables generated in this section will be the foundation for formulating the objective function in Section 4.

3.1 EFA structure

In the first step of formulating MILC from an EFA model, a reachability analysis is done on each individual EFA to identify all reachable locations that are also co-reachable from a marked state. This analysis is performed locally and is hence excluding any restrictions concerning shared events and guards. It is important to note that this concept does not include unlimited loops in the EFAs, where the same location can be visited infinite number of times.

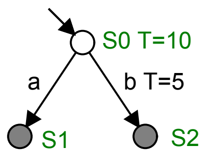


Fig. 1. EFA with cost on location S0 and on one transition.

For every reachable location in the local EFA, a real variable is created. The real variable, t_k^i, models the global time when the location k in EFA i is left in the schedule. The total system will hence consist of $N = \sum_{i=1}^{A} N_i$ real variables, where A is the total number of EFAs in the system and N_i is the number of locations in EFA i. These variables are decision variables in the optimization problem and hence "tuning" parameters for the solver to use for finding the optimal schedule. A transition in EFA i, from location k to location $k+1$ is then modeled by the linear constraint

$$t_{k+1}^i \geq t_k^i + T_{k+1}^i \tag{4}$$

where $T_{k+1}^i \geq 0$ is the minimal time to spend in location $k+1$, specified by physical constraints. The EFA can have a minimal execution time on both locations and transitions, as can be seen in Figure 1. The minimal time to go from location S0 to S1 in Figure 1 will then differ from the minimal time going from S0 to S2. The minimal time allowed to spend in location k in EFA i is hence $T_k^i = T_k^i[location] + T_k^i[transition]$, and will be 10 for firing event a and 15 for firing event b in Figure 1.

Since every location is allowed to have more than one outgoing transition, there will exist a set of possible and equally valid paths through the EFA. The solution to the optimization problem can only include one of these paths and choosing path through the EFAs is part of solving the optimization problem. All paths have to start in the initial location and end in one of the marked locations in the EFA. The EFAs can have multiple marked locations, but are not allowed to have any loops. Considering this, it is guaranteed that the EFA will only have a finite number of possible paths that need to be expressed by constraints. The choice between the different paths through the EFA can thereby be modeled using path choice variables defined as:

Definition 2. A path choice variable, π_p^i, is a binary decision variable having value 1 if path p is chosen for EFA i and 0 otherwise.

In Kobetski and Fabian [2009], another approach was used where binary variables were created for each location with multiple outgoing transitions. This made the paths hidden in the MILC formulation. In this paper, all paths are instead represented in a more similar way to the representation in Panek et al. [2004].

To guarantee that the transition constraint in (4) is only constraining the solution when that transition is used, a path choice variable is added to the constraint as described in (2). Since the transition can be included in multiple paths a more compact constraint can be formulated. In the case of two involved paths, a constraint can be expressed as

$$t_{k+1}^i \geq t_k^i + T_{k+1}^i - (1 - \pi_p^i - \pi_{p+1}^i)M \tag{5}$$

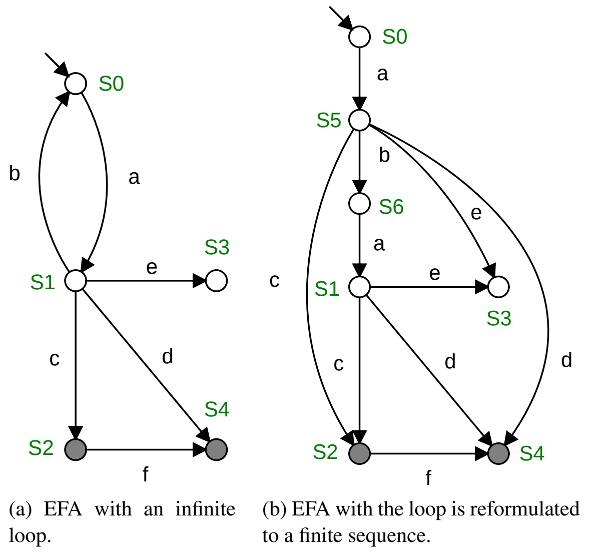

S0
b
a
S1
e
S3
c
d
S4
S2
f
(a) EFA with an infinite loop.
S0
a
S5
b
S6
e
a
c
S1
e
S3
c
d
d
S2
f
S4
(b) EFA with the loop is reformulated to a finite sequence.

Fig. 2. EFAs with several possible paths.

where π_p^i and π_{p+1}^i are the path choice variables for the two paths. See (6) for examples with one and three paths. This compact formulation reduces the number of constraints in the problem formulation. It can be automatically done by comparing every new constraint with previously created constraints before adding them to the constraint matrix.

In Figure 2 (a), an EFA with an unlimited loop is presented. Due to the limitation of not considering any loops in the system, the event b will not be included in any paths. A finite looping behavior can be modeled by multiple locations and time variables, but since the number of variables can not be infinite, the maximum number of loops has to be specified in the model. In Figure 2 (b), the loop in (a) is limited to fire at most one time.

From the EFA in Figure 2 (a), three paths ac, ad, and acf will be considered. The path ae will not be considered, since location S3 is blocking. With the reduction of constraints as in (5), the constraints for the three paths will be

$$\begin{aligned}
t_0^1 &\geq T_0^1 \\
t_1^1 &\geq t_0^1 + T_1^1 - (1 - \pi_1^1 - \pi_2^1 - \pi_3^1)M \\
t_2^1 &\geq t_1^1 + T_2^1 - (1 - \pi_1^1 - \pi_3^1)M \\
t_4^1 &\geq t_1^1 + T_4^1 - (1 - \pi_2^1)M \\
t_4^1 &\geq t_2^1 + T_4^1 - (1 - \pi_3^1)M \\
1 &\geq \pi_1^1 + \pi_2^1 + \pi_3^1 \\
1 &\leq \pi_1^1 + \pi_2^1 + \pi_3^1
\end{aligned} \tag{6}$$

where π_p^i is the path choice variables for the three paths. $t_0^1 \geq T_0^1$ is a constraints for the minimal time spent in the initial location and will always be added without any path choice variables, since the EFA has only one initial location.

3.2 Shared events

If two or more EFAs have the same event in their alphabet, the event has to be fired synchronously for all those EFAs. By including shared events in the problem formulation, the synchronization operator for EFA, defined in Sköldstam et al. [2007], is included in the conversion to MILC. This was also included in Kobetski and Fabian [2009], where ordinary automata were considered. Due to differences in how the automata structure is handled, the synchronization is modeled differently in this paper.

A shared event puts the constraints on multiple transitions to fire simultaneously, if the event is fired. This is formulated by the following two constraints forcing the solution to have both events fired at the same time.

$$t_k^i \geq t_\ell^j - \big(2 - \sum_{p=1}^{P_i} \pi_p^i - \sum_{p=1}^{P_j} \pi_p^j\big)M$$
$$t_\ell^j \geq t_k^i - \big(2 - \sum_{p=1}^{P_i} \pi_p^i - \sum_{p=1}^{P_j} \pi_p^j\big)M$$

The transition with a shared event is fired from location k in EFA i and location ℓ in EFA j. $\sum_{p=1}^{P_i} \pi_p^i$ is the sum of all path choice variables in EFA i that include the transition with the shared event.

If one shared event is included in all paths for one EFA, it puts a limitation on the other EFAs to only be able to execute paths that include that event. By identifying these situations, some impossible paths can be excluded from the mathematical formulation. This will reduce the number of path choice variables and possibly the number of real variables and transition constraints, if any location or transition is included only in impossible paths. This reduction is possible to do due to the suggested conversion method for automata structure, see Section 3.1.

3.3 Logical transition conditions

EFA modeling includes the possibility to use guard conditions on firing events. The conditions are logical expressions formed by integer variables and the variables are changed by actions on transitions. In this paper, only the most common types of statements are considered, which is considered to be enough for modeling common system behaviors. The included statements are that a variable can be less than ($<$), greater than ($>$), less than or equal to ($\leq$), greater than or equal to ($\geq$) or equal to ($==$) an integer value. These statements can be combined with AND ($\wedge$) and OR ($\vee$) and this paper assumes that the statements are on conjunctive normal form, expressed as

$$\bigwedge_{i=1}^{N} \left(\bigvee_{j=1}^{M_i} X_{i,j} \right) \tag{7}$$

where $X_{i,j}$ is an integer relation expression for a variable, e.g. $x_i \leq c$. Observe that negations are expressed by the complement to the statements ($<$), ($>$), ($\leq$), ($\geq$) or ($==$).

A guard on the form (7) is suitable for a modeling situation where AND statements are more common than OR statements. This is assumed to be the case since guards origin from a combination of factors and they all have to be considered simultaneously.

The formulation of MILC from logical transition conditions is done in two steps

(1) Separation of guards into integer relation expressions by interpretation of $\wedge$ and $\vee$ statements.
(2) Relating integer relation expressions to execution order

Interpretation of $\wedge$ statements to MILC is straight forward since the $\wedge$ statement is already built in the concept of having multiple constraints, which all need to be satisfied simultaneously, cf. (3). The guard will hence be separated into multiple $\bigvee_{j=1}^{M_i} X_{i,j}$ statements. These statements are then separated into alternative $X_{i,j}$ statements, as in (2). The binary variables needed for separating the alternatives are referred to as the alternative choice variables, β, defined as:

Definition 3. An alternative choice variable, β, is a binary decision variable having value 1 if the corresponding integer relationship is guaranteed to be valid, in a guard with an alternative. The value is 0 otherwise.

The general guard (7), with $N = 4$, $M_1 = M_2 = M_3 = 1$ and $M_4 = 2$ will be $X_{1,1} \wedge X_{2,1} \wedge X_{3,1} \wedge (X_{4,1} \vee X_{4,2})$, and can be reformulated using alternative choice variables as

$$\begin{aligned}
&X_{1,1}\\
&X_{2,1}\\
&X_{3,1}\\
&X_{4,1} - (1-\beta_1)M\\
&X_{4,2} - (1-\beta_2)M\\
1 &\geq \beta_1 + \beta_2\\
1 &\leq \beta_1 + \beta_2
\end{aligned}$$

In the MILC formulation, only guards including $\vee$ statements will need alternative choice variables.

In the next step, the integer relation expressions, $x_i \leq c$, are converted into MILC using the variables introduced in Section 3.1. The integer relations are interpreted as requirements on the EFAs that change the variable value in their actions. In this paper, only actions setting the variable to a defined value is considered, which simplifies the problem. The variable will then have the specified value, in the schedule, from the time it is set by one action on one transition until it is set by another action on another transition. The setting times are specified by the variables in Section 3.1.

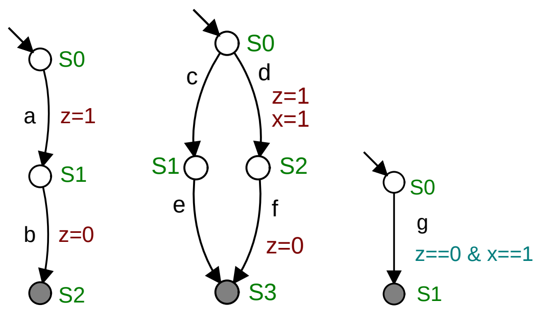
If the integer relation is not an equal statement, there might exist several values on x_i that fulfill the statement. The total number of combinations will then be the sum of all combinations for each value. When there exist more than one combination, an alternative between the combinations will be formulated using M and a set of combination choice variables, α, defined as:

Definition 4. A combination choice variable, α, is a binary decision variable having value 1 if the corresponding combination of actions is chosen and 0 otherwise.

The combinations are then converted into sets of constraints, forming the order of execution of the actions. Each constraint will relate two location variables to each other. One variable will represent the transition with the guard and one variable will represent when the action is set. To guarantee that the constraint is only constraining when both the path for the guard and the path for the action are included in the schedule, the combination constraint is formulated as

$$\begin{aligned}
t_k^i \geq t_\ell^j - (1-\alpha)M - (1 - \sum_{p=0}^{P} \pi_p^i)M - (1 - \sum_{p=0}^{P} \pi_p^j)M\\
\sum_{p=0}^{P} \pi_p^i \geq \alpha
\end{aligned} \tag{8}$$

where $\sum_{p=0}^{P} \pi_p^i$ and $\sum_{p=0}^{P} \pi_p^j$ are the sums of all path choice variables enabling the transition with the action and the guard respectively and α is the combination choice variable for that combination. The second equation in (8) says that the combina-



(a) EFA 1. (b) EFA 2. (c) EFA 3.

Fig. 3. EFAs with guards and actions.

tion itself is only valid if one of the paths including the action is chosen. This constraint is only used for constraints where the action must happen before the transition with the guard and is needed to guarantee that a marked state will be reached.

In Figure 3, a system is presented where two EFAs, EFA 1 and EFA 2, have actions for variable x and z and EFA 3 has a guard for those variables. EFA 2 has two alternative paths and all actions are in path π_2^2. The guard in EFA 3 has two integer relationships with an $\wedge$ statement and the guard can hence be converted to

$$x == 1 : \begin{cases} t_0^3 & \geq t_0^2 - (1-\pi_2^2)M \\ \pi_2^2 & \geq 1 \end{cases}$$

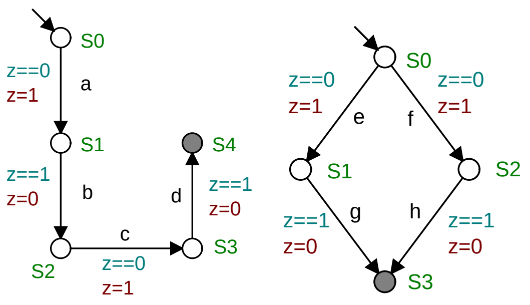
$$z == 0 : \begin{cases} t_0^1 & \geq t_0^3 - (1-\alpha_1)M \\ t_0^2 & \geq t_0^3 - (1-\alpha_1)M - (1-\pi_2^2)M \\ t_0^3 & \geq t_1^1 - (1-\alpha_2)M \\ t_0^2 & \geq t_0^3 - (1-\alpha_2)M - (1-\pi_2^2)M \\ t_0^3 & \geq t_2^2 - (1-\alpha_3)M - (1-\pi_2^2)M \\ t_0^1 & \geq t_0^3 - (1-\alpha_3)M \\ \pi_2^2 & \geq \alpha_3 \\ t_0^3 & \geq t_1^1 - (1-\alpha_4)M \\ t_0^3 & \geq t_2^2 - (1-\alpha_4)M - (1-\pi_2^2)M \\ \pi_2^2 & \geq \alpha_4 \\ 1 & \geq \alpha_1+\alpha_2+\alpha_3+\alpha_4 \\ 1 & \leq \alpha_1+\alpha_2+\alpha_3+\alpha_4 \end{cases}$$

The first combination, α_1, is when g is fired before both a and d and the second combination, α_2, is when g is fired after b but before d. All terms $(1-\alpha)M$, $(1-\sum_{p=0}^{P}\pi_p^i)M$ and $\sum_{p=0}^{P}\pi_p^i \geq \alpha$ from (8) have been removed from the constraints if they are 0.

3.4 *Resource allocation*

The most common use of variables is to model a mutual exclusion, e.g. a shared zone between two robots. This situation is also known as resource allocation and mutual exclusion. It is a central part of scheduling. Mutual exclusion of a shared zone z can be modeled on a transition with the guard $z == 0$ and the action $z := 1$ for booking and the guard $z == 1$ and the action $z := 0$ for unbooking of the shared zone.

The mutual exclusion problem can be reformulated as a problem forcing a set of booking and unbooking conditions happening in a sequential order, to prevent concurrent usage of the zone. With multiple sets of booking and unbooking pairs of actions, there will be multiple possible sequential orders. To separate different possible execution orders, a set of combination choice variables will be introduced, as in Section 3.3.



(a) Sequence. (b) Alternative.

Fig. 4. Two EFAs with zone bookings and unbookings.

With two EFAs sharing a zone the corresponding constraints will be

$$\begin{aligned} t_b^i &\geq t_u^j - (1-\alpha_1)M \\ t_b^j &\geq t_u^i - (1-\alpha_2)M \\ 1 &\geq \alpha_1+\alpha_2 \\ 1 &\leq \alpha_1+\alpha_2 \end{aligned}$$

where t_b^i is the time when EFA i books the zone and t_u^i is the time when the EFA unbooks it.

In theory, the number of possible ways that n pairs of booking and unbooking actions can be combined to prevent concurrent usage of a shared zone is $n!$. This can be a lot of combinations, which needs to be considered when creating the EFA model. However, there might be some combinations of booking and unbooking that obviously will conflict the path constraints. In Figure 4, two EFAs with two booking and unbooking pairs each are presented. If no consideration is taken to the path constraints, there will be $4! = 24$ unique combination. However, since the sequential constraints say that e.g. event **c** can not happen before **a**, a lot of combinations can be neglected and the remaining combinations that have a potential to be valid are only six, namely the event sequences

$$abcdeg, abcdfh, abegbc, abfhbc, egabcd, fhabcd$$

The six combinations will be separated by six combination choice variables, which makes the reduction from 4! variables an important reduction of the size of the optimization problem formulation.

By creating combinations globally, for all booking and unbooking transitions, redundancies are avoided compared with the local approach described in Section 3.3. This makes the global combinations very suited for scheduling applications.

4. FORMULATING THE OBJECTIVE FUNCTION

This section covers how the variables created in Section 3 can be used when formulating an objective function for the optimization problem.

4.1 *Cycle time optimization*

In cycle time optimization, the schedule that minimizes the total time when executing one cycle is of interest. To find the total time, a new real variable, t_{total}, is introduced together with a set of constraints on the form

$$t_{total} \geq t_{m_p}^i - (1 - \sum_{p=1}^{P} \pi_p^i)M$$

where $t^i_{m_p}$ is the marked state in the end of path p in EFA i and $\sum_{p=1}^{P} \pi^i_p$ is the sum of all path choice variables for all paths leading to the marked state $t^i_{m_p}$.

The objective function is then formulated as

$$min \quad t_{total}$$

The solution will be a cycle time optimal schedule, under the assumption that all EFAs executes once during the work cycle.

4.2 Multi-criteria objective function

Introducing a state cost function $f^i_k(t^i_k - t^i_{k-1})$, defined as a function of the time spent in the state, a general nonlinear objective function can be formulated as

$$min \quad F(\mathbf{t}) = \sum_{i=1}^{A} \sum_{k=1}^{S_i} \left(c^i_k (1 - \sum_{p=1}^{P_k} \pi^i_p) f^i_k (t^i_k - t^i_{k-1}) \right) \tag{9}$$

where c^i_k is the weight coefficient for location k in EFA i, introduced to rank the importance of $f^i_k(t)$ in F, $\sum_{p=1}^{P} \pi^i_p$ is the sum of the path choice variables for all paths including location k in EFA i. This objective function can express any kind of condition, but linear cost functions are easier for the solvers to handle. For nonlinear objective functions convexity is a crucial property.

4.3 The resulting schedule

The solution of the optimization problem is derived using a MILP or MINLP solver, and which solver to choose depends on how the objective function looks like. Information about solvers can be found in Floudas and Pardalos [2001].

If the solver finds an optimal solution $F^* = F(\mathbf{x}^*)$ that fulfills all constraints in (3), the values of $\mathbf{x}^*$ will represent how the system should be controlled to obtain optimum. It is important to return the optimal schedule to the user on the same modular form that was given by the user from the beginning to make it easier for the user to analyze the result. This can be achieved by e.g. introducing additional, guards on the original EFAs to restrict the total system to follow the optimal path.

5. SIZE OF PROBLEM FORMULATION

In this section, it will be discussed how the MILP problem formulation scales with increasing number of locations, transitions, branches, guards etc. This is not necessarily directly related to the computational time for the solver, but it is a measure of how large the mathematical problem formulation will be.

The MILP problem will scale linearly with increased sequences. For every location, there will be one additional real variable and one constraint relating the new location to the previous location in the sequence. When introducing further parallelism by adding an additional EFA, no new variables nor constraints are needed to model the parallel relationship. This makes the problem formulation scale very well for both increased sequences and parallelism in the model.

The problem size doesn't scale equally good for alternatives. This is natural, since an alternative means that there are two or more options in the EFA model and a mutual exclusions needs to be included to separate them. The mutual exclusion introduces binary decision variables, which increases the problem size. It is however important to note that most industrial scenarios have quite few alternatives compared to the parallelism. To handle parallelism in an efficient way is therefore essential for industrial systems such as robot cells. The main reason for this is that common zones are rarely shared by more than two resources.

Adding extra transitions between locations in the model will increase the number of possible paths through the EFA and adding path choice variables accordingly. It will however not add to the number of constraints. Additional transitions is a sign of more complexity in the system and will inevitable result in a larger mathematical formulation.

The other binary variables origin from logical statements in guards. In this paper, this is treated by forming alternatives between all theoretical ways to fulfill the statements. This will include alternatives that will not be reachable, which might increase the problem size unnecessarily. The main advantage of this is that global reachability is not necessary, which will save a lot of computational time for the conversion from EFA to MILC.

6. ILLUSTRATIVE EXAMPLE

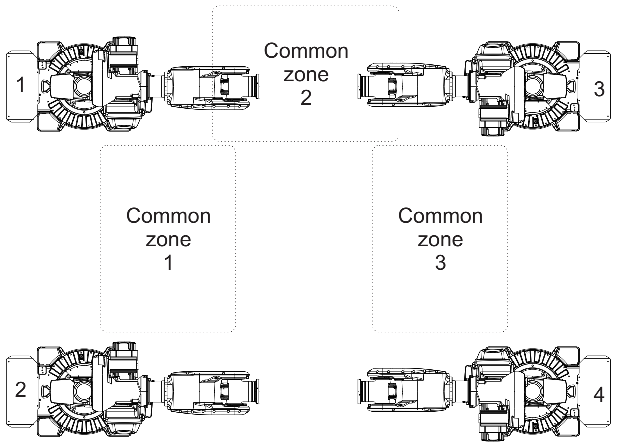


Fig. 5. The example system with four industrial robots working in three shared zones.

As an example of how the presented method can be used to formulate optimization problems from discrete event models, an industrial example has been created, see Figure 5. The example consists of four robots performing a total of 40 moving operations in three zones. The robots should move between predefined coordinates in an order that should be decided by optimization. Due to the risk of collision, only one robot at a time is allowed to use a zone. The order that one robot should perform its moving operations is defined, but it is not given in which order the robots should use the zones. Robot 1 and 3 even have to visit two zones to complete their sequences and it is arbitrary for them which of the two zones to visit first.

The whole system was modeled by EFAs, such that all execution constraints were included as guards and actions on the transitions. The EFA model was then translated into mixed-integer linear constraints using an algorithm based on the method presented in this paper. The result was 349 constraints using 82 real and 32 binary variables. This example can theoretically be viewed as finding the optimal solutions of 2^{32} nonlinear programs. However, by knowing the structure of the problem and how the binary variables were added in groups in the problem formulation, the number of interesting scenarios can be reduced to 1280.

The objective function for the optimization problem is a consequence of the criterion for the scheduling. In this example, the schedule representing the lowest total energy consumption should be derived. The objective function was hence formulated as in Vergnano et al. [2010], which resulted in a sum of approximately 80 nonlinear functions of the real variables.

The mixed-integer nonlinear programming problem can be solved by solving individual programs and compare the solutions. In this case, every program took on average 3 seconds to solve, which would make a brute force algorithm solve the problem in about one hour. With a branch and bound technique, the calculation time would be considerably reduced to a reasonable level. The result from the optimization was the starting and stopping times for each robot operation for the optimal schedule. This information can then be used when programming the robot movements in the real factory.

7. CONCLUSIONS AND FUTURE WORK

This paper presents a method to convert a model of Extended Finite Automata to mixed-integer linear constraints. The concept includes conversion of the structure for modular EFAs, synchronized execution from shared event and execution order due to guards and actions on the transitions. For the special case of guards and actions used to model shared resources, a reduced conversion is presented using fewer variables and constraints than for the case of general guards.

The optimization objective function is then constructed using real and binary variables. The paper suggests usage of state cost functions, where the costs for the locations are modeled individually as nonlinear functions of the time spent in them. The optimization problem will then be a mixed-integer nonlinear programming problem with an objective function constructed from multiple criteria. The main contribution of this paper is the concept of combining the easy method of using EFAs to model a discrete event system with the utilization of powerful commercial solvers for mixed-integer nonlinear programming problems.

This mathematical problem formulation shows to be powerful for systems with a lot of parallelism but a limited number of alternatives. This is an important contribution to the automation industry, since for instance robotic systems have a lot of parallel activities and to optimize such systems is of decisive importance.

In the future, the conversion could be further improved by neglecting more impossible situations than presented here. For instance shared events and order of transitions with guards and actions can be examined. Another topic could be to use the resulting matrix representation to analyze the system's behavior even more than suggested in this paper. The matrix could be used to identify and combine similar constraints to reduce the number of constraints and binary decision variables. Finally, it is important to further evaluate in which situations MILP/MINLP algorithms are preferable compared to graph search algorithm and vice versa for optimization of EFA models.

ACKNOWLEDGEMENTS

This work has been carried out within the Sustainable Production Initiative, the Wingquist Laboratory VINN Excellence Centre, and the Production Area of Advance at Chalmers. It has been supported by the Vinnova and General Motors. This support is gratefully acknowledged.

REFERENCES

Y. Abdeddaimi and O. Maler. Job-shop scheduling using timed automata. *CAV*, LNCS 2102:478–492, 2001.

A. Bayen and C. Tomlin. Real-time discrete control law synthesis for hybrid systems using MILP: application to congested airspace. *American Control Conference*, pages 4620–4626, 2003.

A. Bemporad, S. Cairano, and J. Júlvez. Event-driven optimal control of integral continuous-time hybrid automata. *Proc. of the 44th IEEE Conference on Decision and Control, and the European Control Conference*, 2005.

G. B. Dantzig. *Linear programming and extensions*. Princeton University Press, Princeton, New Jersey, 1963.

C. Floudas and P. Pardalos. *Encyclopedia of optimization*. Kluwer Academic Publishers, Dordrecht, 2001.

C. Fonseca and P. Fleming. An overview of evolitionary algorithms in multiobjective optimization. *Evolutionary Computations*, 3:1–16, 1995.

P. Hart, N. Nilsson, and B. Raphael. A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on systems science and cybernetics*, 4:100–107, 2007.

A. Kobetski and M. Fabian. Time-optimal coordination of flexible manufacturing systems using deterministic finite automata and mixed integer linear programming. *Discrete Event Dyn Systs*, 19:287–315, 2009.

A. Kobetski, D. Spensieri, and M. Fabian. Scheduling algorithms for optimal robot cell coordination - a comparison. *Proc. of IEEE Conference on Automation Science and Engineering*, 2006.

T. Liljenvall. *Scheduling for production systems*. Lic. thesis 293L, School of Electrical and Computer Engineering, Chalmers University of Technology, Gteborg, 1998.

A. Mannani, Y. Yang, and P. Gohari. Distributed extended finite-state machines: Communication and control. *Proc. of Workshop on Discrete Event Systems*, pages 161–167, 2006.

K. Miettinen. *Nonlinear multiobjective optimization*. Kluwer Academic Publishers, Dordrecht, 1998.

S. Panek, O. Stursberg, and S. Engell. Job shop scheduling by combining reachability analysis with linear programming. *In Proc. IFAC workshop on discrete event systems*, pages 199–204, 2004.

S. Panek, O. Stursberg, and S. Engell. Efficient synthesis of production schedules by optimization of timed automata. *Control Engineering Practice*, 14:1183–1197, 2006.

A. Schrijver. *Theory of linear and integer programming*. Wiley, Chichester, 1986.

M. Sköldstam, K. Åkesson, and M. Fabian. Modeling of discrete event systems uding finite automata with variables. *Proc. of IEEE Conference on Decision and Control*, 2007.

A. Vergnano, C. Thorstensson, B. Lennartson, P. Falkman, M. Pellicciar, C. Yuan, S. Biller, and F. Leali. Embedding detailed robot energy optimization into high-level scheduling. *IEEE Conference on Automation Science and Engineering*, 2010.