Proceedings of the
44th IEEE Conference on Decision and Control, and
the European Control Conference 2005
Seville, Spain, December 12-15, 2005

# Analysis of Logic Controllers by Transformation of SFC into Timed Automata

Olaf Stursberg, Sven Lohmann
Process Control Laboratory (BCI-AST)
University of Dortmund, 44221 Dortmund, Germany.
Email: olaf.stursberg@uni-dortmund.de

*Abstract*— **This paper proposes an approach to connect Sequential Function Charts (SFC), an industrially recognized and used description of logic controllers, to algorithmic verification. Based on a rigorous syntactical and semantical definition of SFC, the paper describes a formal scheme to generate a corresponding model represented by synchronized Timed Automata (TA). The latter model can be composed with a plant model specified as timed or hybrid automata. In order to verify safety properties for the controlled system, existing algorithms for model checking can eventually be applied to the composition.**

*Index Terms*— **Automata, discrete event systems, logic control, timed systems, verification.**

## I. INTRODUCTION

In industry, the typical means to describe and implement logic controllers (ladder diagrams, function block diagrams, sequential function charts, etc. [1]) are somewhat different from the type of models (automata, transition systems, Petri nets, etc.) used for verification techniques like *model checking* [2]. In order to bridge this gap, and thus to make a rigorous analysis of industrially relevant logic controllers possible, a number of groups have recently suggested approaches to transform Sequential Function Charts (SFC), a popular grahical description for logic controllers, into verifiable code. While [3], [4], [5], [6] only consider SFC with untimed actions, a concept which includes timed actions, i.e. control actions that are limited to a certain period of time or are applied after a delay, has first been suggested in [7], [8]. The principle of this method is to transform the SFC into modular Timed Automata (TA), to which model checking can be applied. The complete scheme, in which this transformation is embedded, is shown in Fig. 1: An initial controller design considering the relevant requirements (control objectives) leads to a controller given as SFC, which is subsequently transformed into the modular TA representation. The latter is composed with a plant model which is set up as timed or hybrid automaton (HA), depending on an appropriate level of detail. For the composed model, safety and operability properties can be verified, e.g. by the techniques described in [9] for TA, or counterexample-guided verification for HA [10]. Safety and operability specifications refer to the properties that the controlled system does never reach unsafe plant states, or does eventually reach desired states. If the analysis reveals that the specification is falsified, the controller design has to be modified on the SFC-level – otherwise the SFC can be transferred to a programmable logic controller (PLC).

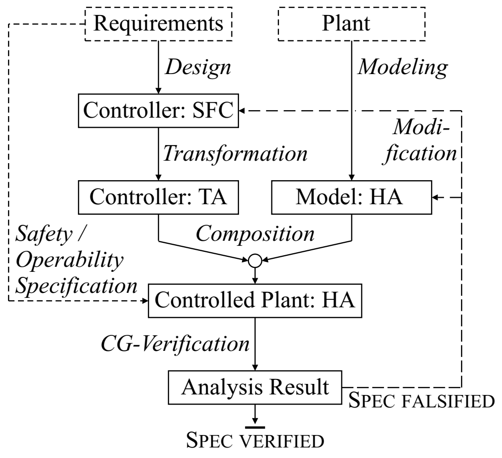


Fig. 1. Analysis scheme (SFC - Sequential Function Chart, TA - Timed Automata, HA - Hybrid Automaton).

This paper focusses on the step of transforming SFC into TA. While the concept of the transformation has already been summarized in [7], [8], this paper provides the formal basis which obviously is required within a verification approach. In detail, this paper first introduces a formal definition of SFC (Sec. II), which is not provided by the standard [1], or in more detailed texts such as [11]. The formal definition explicitly considers the graphical structure by introducing a corresponding grammar. The grammar (and by that the explicit distinction of simultaneous and alternative executions) distinguishes our definition from that in [12]. The grammar is an essential component of the transformation into TA, since it defines the interaction and the number of the synchronizing automata on the TA-level. An important property of the transformation described in Sec. III is that it conservatively abstracts from the execution cycles of the PLC where this simplification does not change the verification result.

## II. A FORMAL DEFINITION OF SEQUENTIAL FUNCTION CHARTS

As shown in Fig. 2 for an example, the building blocks of an SFC are *steps* $s$ with an attached *action block* $b$, *transitions* with an associated *condition* $g$, *parallel executions* enclosed by double horizontal lines, and *alternative executions* marked be single horizontal lines. The vertical lines represent the flow by which the blocks are processed from top to down, with the exception of *loops* which point in the

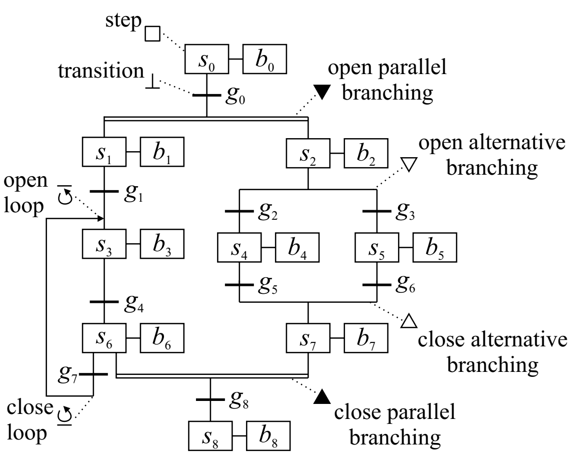

step
transition
open parallel branching
open alternative branching
open loop
close alternative branching
close parallel branching
close loop
$s_0$ $b_0$ $g_0$ $s_1$ $b_1$ $s_2$ $b_2$ $g_1$ $g_2$ $g_3$ $s_3$ $b_3$ $s_4$ $b_4$ $s_5$ $b_5$ $g_4$ $g_5$ $g_6$ $s_6$ $b_6$ $s_7$ $b_7$ $g_7$ $g_8$ $s_8$ $b_8$

Fig. 2. Example of an SFC.

opposite direction. The following grammar first specifies the structures of an SFC which are deemed feasible throughout this paper:

**Definition 1**: The grammar $\Gamma = (\Sigma, \Psi, \psi, \Phi)$ of an SFC consists of the alphabet $\Sigma = \{\square, \perp, \triangledown, \vartriangle, \blacktriangledown, \blacktriangle, \overline{\circlearrowleft}, \underline{\circlearrowright}, [, ]\}$, a set of auxiliary variables $\Psi = \{ST, chain, loop, alt, par\}$, the start symbol $\psi = \ 'ST'$, and the set $\Phi$ of production rules as follows:

$\phi_1 : 'ST' = \{'\square', chain\}$ (start rule)

$\phi_2 : chain = \{'\perp\square', chain \mid' \triangledown', alt,' \vartriangle \square', chain \mid$
$\quad '\perp\blacktriangledown', par,' \blacktriangle', chain \mid loop, chain \mid \emptyset\}$

$\phi_3 : loop = \{'\perp\overline{\circlearrowleft}\square', chain,' \perp\underline{\circlearrowright}'\}$

$\phi_4 : alt = \{'[', chain,' \perp]', ('[', chain,' \perp]')^*,' [', chain,' \perp]'\}$

$\phi_5 : par = \{'[\square', chain,' ]', ('[\square', chain,' ]')^*,$
$\quad '[\square', chain,' ]'\}.$ $\diamondsuit$

As an example, $\Gamma$ produces the following string for the SFC shown in Fig. 2: $\Upsilon := \square\perp\blacktriangledown[\square\perp\overline{\circlearrowleft}\square\perp\square\perp\underline{\circlearrowright}][\square\triangledown[\perp\square\perp][\perp\square\perp] \vartriangle \square]\blacktriangle\perp\square$.

Using the grammar, the syntax of SFC can now be defined formally as follows. (Note that here we only refer to non-hierarchical SFC in order to simplify the notation.)

**Definition 2**: A Sequential Function Chart is given by $M_{SFC} = (S, s^0, T, \Gamma, X, G, A, \alpha, C)$ consisting of:

- the finite set of steps $S = \{s_1, \ldots, s_{n_S}\}$, $n_S \in \mathbb{N}$, with an initial step $s^0 \in S$.
- a finite set $X = X_{in} \cup X_{out} \cup X_{int}$ with the pairwise disjoint sets of input ($X_{in}$), output ($X_{out}$), and internal variables ($X_{int}$); the evaluation of a variable is denoted by $v(x) \in \mathbb{Q}$.
- a finite set $C$ of clocks which contains a clock $c$ for each non-empty time quantifier (see below), and the evaluation of $c$ is denoted by $\omega(c) \in \mathbb{Q}$.
- a finite set of transition conditions $G$, with $g \in G$ as a boolean combination of propositions $v(x) \sim d$, $\omega(c) \sim d$ with $\sim \in \{<, \leq, =, \geq, >\}$; $x \in X_{int} \cup X_{in}$, $d \in \mathbb{Q}$.
- a finite set of transitions $T : 2^S \setminus \{\emptyset\} \times G \rightarrow 2^S \setminus \{\emptyset\}$ such that $T$ is restricted according to the grammar $\Gamma$ as follows (with $\{s, s', s'', s'''\} \subset S$, $\{g, g', g''\} \subset G$, and the assumption that the string $\Upsilon$ produced by $\Gamma$ is finite):
  1) $\square\perp\square$: two steps are connected by a transition $t \in (s, g, s')$.
  2) $\square\perp\overline{\circlearrowleft}\square\perp \ldots \square\perp\underline{\circlearrowright}\square$: the first step $s' \in S$ inside of the loop is reached from the last one preceding the loop ($s$) by $t = (s, g, s')$ and from last step in the loop ($s''$) by $t = (s'', g', s')$. The first step following the loop ($s'''$) is reached by $t = (s'', g'', s''')$.
  3) $\square\triangledown[\perp\square\ldots]\ldots[\perp\square\ldots] \vartriangle$: a separate transition $t = (s, g, s') \in T$ exists from the step preceding the alternative branching to each first step $s'$ of the alternative branches. The set of transitions $T_{ab} \subset T$ associated with the alternative branching is totally ordered, where the order specifies which $t \in T_{ab}$ is taken if more than one transition is enabled.
  4) $\triangledown[\ldots\square\perp]\ldots[\ldots\square\perp] \vartriangle \square$: one transition $t = (s, g, s') \in T$ from each final step of an alternative branch into the first step following the closure of the alternative branching exists; as in (3), a total order on these transitions is assumed to be given.
  5) $\square\perp\blacktriangledown[\square\ldots]\ldots[\square\ldots]\blacktriangle$: the set of steps $S' \subset S$ following a parallel branching is reached from the step preceding the branching through $t = (s, g, S')$.
  6) $\blacktriangledown[\ldots\square]\ldots[\ldots\square]\blacktriangle\perp\square$: all steps preceding the closure of the parallel branching $S'' \subset S$ are left by a single transition $t = (S'', g, s''')$, and the step $s'''$ following the closure is reached.
- a finite set $A$ of actions, each of which is a triple $a = (q, \tau, o, f)$ consisting of an action qualifier $q \in Q = \{N, St^1, R, P, P1, P0, L, D, SD, DS, SL\}$, a time quantifier $\tau \in \{\emptyset, \tau_v\}$ with $\tau_v \in \mathbb{Q}^{\geq 0}$, an operand $o$ which is either a variable $x \in X_{int} \cup X_{out}$ or a function of variables $x \in X_{in} \cup X_{int}$, and a boolean action control flag $f \in \{0, 1\}$ indicating if $a$ has to be executed ($f = 1$), or not ($f = 0$).
- a function $\alpha$: $S \rightarrow 2^A$ which assigns an action block $b = \alpha(s)$ as an ordered subset of $A$ to each $s \in S$. $\diamondsuit$

Feasible executions of $M_{SFC}$ are defined by the following semantics:

**Definition 3**: A *configuration* of $M_{SFC}$ is given by $\gamma = (V, S_a, A_a, A_f, \Omega)$, where $V$ is the set of evaluations for all $x \in X$, $S_a \subset S$ is the set of active steps, $A_a \subseteq A$ is the set of active actions, $A_f \subseteq A$ the set of final actions, and $\Omega$ the set of evaluations for all $c \in C$. A *run* of $M_{SFC}$ is a sequence $r_{SFC} = ((\gamma_0, \chi_0), (\gamma_1, \chi_1), (\gamma_2, \chi_2), \ldots)$ of *timed configurations* $(\gamma_i, \chi_i)$ where:

- $\chi_i \in \mathbb{R}^{\geq 0}$ is the time at which $\gamma_i$ is recorded. The update of $\gamma_i$ to $\gamma_{i+1}$ is time-triggered with a (possibly varying) *cycle time* $\tau_{c,i} = \chi_{i+1} - \chi_i$ with $\tau_{c,i} \in [\tau_c^{min}, \tau_c^{max}]$, $\tau_c^{min} \in \mathbb{Q}, \tau_c^{max} \in \mathbb{Q}$.
- the initial timed configuration consists of $\chi_0 = 0$ and $\gamma_0 = (V_0, S_{a,0}, A_{a,0}, A_{f,0}, \Omega_0)$ in which $V_0$ contains

[1]The qualifier $St$ replaces here the standard symbol $S$ (store) to distinguish it from the set of steps.

initial evaluations $v_0(x)$ for all $x \in X_{in}$ (the *sensor readings*) and $v_0(x) = 0 \ \forall x \in X_{int} \cup X_{out}$. The initial set of active steps is $S_{a,0} = \{s^0\}$, and the action sets are initialized to $A_{a,0} = b(s_0)$, and $A_{f,0} = \emptyset$. The initial clock evaluations $\Omega_0$ are given by $\omega(c)_0 := 0$ for all $c \in C$.

- the configurations $\gamma_{i+1} = (V_{i+1}, S_{a,i+1}, A_{a,i+1}, A_{f,i+1}, \Omega_{i+1})$ for all $i \in \mathbb{N}^{\geq 0}$ result from $\gamma_i$ by:

1) $V_{i+1}$ is updated for all $x \in X_{in}$;
2) Let $\epsilon_i : V \to V$ denote a function that represents the execution of all actions $a \in A_{a,i} \cup A_{f,i}$ according to a given total order of $A_{a,i} \cup A_{f,i}$. Then, $V_{i+1} = \epsilon(V_i)$ is the update of $v(x)$ for all $x \in X_{int} \cup X_{out}$. Given $s \in S$, $\alpha(s) = b \subset 2^A$, $a = (q, \tau, o, f)$, the execution of $a$ for the operand $o$ depends on the qualifier $q$ as follows:
   a) $q = N$ (*not stored*): $a$ is executed while $s \in S_{a,i}$.
   b) $q = St$ (*stored*): $a$ is executed when $s \in S_{a,i}$, and the result is stored in $v(x)$ until another action resets this operand by $q = R$.
   c) $q = R$ (*reset*): the action resets the operand to a default value (0 in case of boolean variables), and this result is stored until the operand is modified by another action; $R$ has always higher priority than any other qualifier.
   d) $q = P$ (*pulse*): $a$ is executed once when $s$ is reached, and once when it is left; for $q = P1$, $a$ is only executed when $s$ is reached, and for $q = P0$ respectively, $a$ is executed once when $s$ is left.
   e) $q = L$ (*limited*): $a$ is executed for the time $\tau_v$, but at most for the duration in which $s \in S_{a,i}$.
   f) $q = D$ (*delayed*): after the time $\tau_v$, the action is executed, but it is limited to the activity of $s$.
   g) $q = SD$ (*stored delayed*): $a$ is executed after the delay $\tau_v$ even if $s$ becomes inactive before, and the result is stored until a reset is applied to $o$.
   h) $q = DS$ (*delayed stored*): $a$ is executed after the delay $\tau_v$ if the step is still active, and the result is stored until a reset is applied.
   i) $q = SL$ (*stored limited*): the action is executed for the time $\tau_v$ even if $s$ is left before.

   If there exists a set $A_{o,i} \subset A_{a,i} \cup A_{f,i}$ of actions with an equal operand and $|A_{0,i}| > 1$, a total order on $A_{o,i}$ determines which action is executed ($f = 1$, otherwise: $f = 0$).
3) Each clock $c \in C$ is updated as follows to obtain $\Omega_{i+1}$:
   a) $\omega(c) := 0$ if an action $a = (q, \tau, o, f)$ has a time-dependent qualifier ($L$, $D$, $SD$, $DS$, or $SL$), and the step $s \in S_a$ was reached in the current iteration $i$.
   b) else: $\omega(c) := \rho$, where $\rho \in \mathbb{R}$ is the time elapsed since $c$ was last reset to zero.
4) The new set of active steps $S_{a,i+1}$ is obtained from $S_{a,i}$ by inserting $s \in S_{a,i}$ into $S_{a,i+1}$ iff: $\exists \ t = (S', g, S'') \in T$ with: $g \in G$ evaluates to true, $S' \subset S_{a,i}$, $S'' \subset (S \setminus S_{a,i})$, and $\nexists \ t^* \in T$ with $t^* \prec t$ according to the total order of $T$ iff $t^*$ and $t$ belong to an alternative branching (case (3.) of the specification of $T$ in Def. 2).
5) The new sets of active and final actions are determined according to $A_{a,i+1} = \{a \mid \ s \in S_{a,i+1} : a \in b = \alpha(s)\}$ and $A_{f,i+1} = \{a \mid \ s \in (S_{a,i} \setminus S_{a,i+1}) : a \in b = \alpha(s) : q = P \vee q = P_0\}$. $\diamondsuit$

For the example in Fig. 2, a possible run $r_{SFC}$ according to Def. II corresponds to the following sequence of active step sets: $S_{a,0} = \{s_0\}$, $S_{a,1} = \{s_1, s_2\}$, $S_{a,2} = \{s_3, s_2\}$, $S_{a,3} = \{s_6, s_5\}$, $S_{a,4} = \{s_3, s_7\}$, $S_{a,5} = \{s_6, s_7\}$, and $S_{a,6} = \{s_8\}$. The actual state sequence depends on the evaluation of the transition conditions $g_i$, which depend on the variable evaluations, and thus on the actions $a \in b(s), s \in S_{a,i}$ which are executed.

## III. CONTROLLER TRANSFORMATION IN TIMED AUTOMATA

### A. Reduction of Runs

As pointed out in [13], the timescales for the PLC cycle and the plant behavior are usually completely different for many applications. In chemical processing systems, e.g., the cycle time is chosen in the order of milliseconds, while changes of $v(x)$, $x \in X_{in}$ (typically exceeding a limit value) are often separated by minutes or hours. Those runs $r_{SFC}$, which cover the period of time relevant for verifying safety properties, consequently comprise high number of cycles - the negative effect on the resources for verification (time and memory) are obvious. The following definitions thus reduce $r_{SFC}$, in order to obtain a smaller number of configurations that have to be considered for verification.

**Definition 4**: For two successive timed configurations $(\gamma_i, \chi_i)$ and $(\gamma_{i+1}, \chi_{i+1})$, the latter is called a *redundant timed configuration* iff: $V_i = V_{i+1}$, $S_{a,i} = S_{a,i+1}$, $A_{a,i} = A_{a,i+1}$, and $A_{f,i} = A_{f,i+1}$. Given a run $r_{SFC}$, the corresponding *reduced run* $\tilde{r}_{SFC}$ is obtained by eliminating all redundant timed configurations from $r_{SFC}$. $\diamondsuit$

Let $\sigma_x = ((v_0(x), \chi_0), (v_1(x), \chi_1), \ldots)$ denote the *value sequence* of a variable $x \in X$ with $v_i(x) \in V_i$. Assume that a function $\mu_x : \mathbb{R}^{\geq 0} \to \mathbb{R}$ assigns a *time trajectory* to $\sigma_x$ according to: $\mu_x(\iota) = v_i(x)$ for $\iota \in [\chi_i, \chi_{i+1}[$. For $x \in X_{in}$, $\mu_x(\iota)$ is called a *sensor signal*, and for $x \in X_{out}$, $\mu_x(\iota)$ defines a *control signal*.

**Lemma 1**: Given sensor signals $\mu_x(\iota)$ for all $x \in X_{in}$, $\iota \in [0, \chi_f]$, $\chi_f \in \mathbb{R}^{>0}$, and a controller as $M_{SFC}$ according to Def. 1-3, the control signals $\mu_x(\iota)$ for all $x \in X_{out}$ are identical for the runs $r_{SFC}$ and $\tilde{r}_{SFC}$. $\blacksquare$

**Proof of Lemma 1**: For $k > 1$, let $r_{SFC} = ((\gamma_0, \chi_0), \ldots, (\gamma_i, \chi_i), \ (\gamma_{i+1}, \chi_{i+1}), \ldots, (\gamma_{i+k}, \chi_{i+k}), \ldots)$ and $\tilde{r}_{SFC} = ((\gamma_0, \chi_0), \ \ldots, (\gamma_j, \chi_j), (\gamma_{j+1}, \chi_{j+1}), \ldots)$, where: $(\gamma_i, \chi_i) = (\gamma_j, \chi_j)$, $(\gamma_{i+k}, \chi_{i+k}) = (\gamma_{j+1}, \chi_{j+1})$, and the redundant timed configurations $(\gamma_{i+1}, \chi_{i+1}), \ldots, (\gamma_{i+k-1}, \chi_{i+k-1})$ are eliminated from $\tilde{r}_{SFC}$. By the definition of reduced runs, it applies for $r_{SFC}$ that $V_i = V_{i+1} = \ldots = V_{i+k-1}$, and for $\tilde{r}_{SFC}$ that $V_j = V_i$. For any $x \in X_{out}$, the partial value sequence

$\sigma_x = ((v_i(x), \chi_i), \ldots, (v_{i+k-1}(x), \chi_{i+k-1}))$ corresponding to $r_{SFC}$ has equal entries $v_i(x) = \ldots = v_{i+k-1}(x)$, leading to a control signal $\mu_x(\iota) = v_i(x)$ for $\iota \in [\chi_i, \chi_{i+k}[$. For $\tilde{r}_{SFC}$ and the same $x$, the partial value sequence $\sigma_x = ((v_j(x), \chi_j), (v_{j+1}, \chi_{j+1}))$ leads to the control signal: $\mu_x(\iota) = v_j(x)$ for $\iota \in [\chi_j, \chi_{j+1}[$ with $v_j(x) \overset{!}{=} v_i(x)$ and $\chi_j = \chi_i$, $\chi_{j+1} = \chi_{i+k}$. Thus, for given sensor signals $\mu_x(\iota)$, $\iota \in \mathbb{R}^{\geq 0}$, the runs $r_{SFC}$ and $\tilde{r}_{SFC}$ encode the same control signals $\forall\ x \in X_{out}$. ■

**Corollary 1**: If $M_{SFC}$ is connected to a plant model, and the behavior of the controlled plant has to be analyzed, it is sufficient to consider $\tilde{r}_{SFC}$ instead of $r_{SFC}$. ■

As a consequence of the corollary, the following transformation of $M_{SFC}$ into timed automata considers $\tilde{r}_{SFC}$.

## B. Definition of Timed Automata

The following type of timed automaton is employed for the transformation:

**Definition 5**: A timed automaton is defined as $TA = (Z, z_0, Lab, Var, C, E, inv)$ with:

- the finite set $Z = \{z_1, \ldots, z_{n_z}\}$ of states with an initial state $z_0 \in Z$;
- the set $Lab$ of synchronization labels including an *empty* symbol $\varepsilon$;
- the finite set of variables $Var$; the variables $x \in Var$ have evaluations $v(x)$ defined on $\mathbb{N} \cup \{0\}$ or $\mathbb{Q}$;
- the finite set $C = \{c_1, \ldots, c_{n_c}\}$ of clocks, and $\xi \in (\mathbb{R}^{\geq 0})^{n_c}$ a vector of clock evaluations;
- a finite set $E$ of transitions $e = (z, z', l, g, \rho, \kappa) \in E$ in which $z \in Z$ is the source state, $z' \in Z$ the target state, $l \in Lab$ a synchronization label, $g$ a transition guard, $\rho$ a function updating a value assignment $\xi \in \mathbb{R}^n$ for all clocks in $C$, and $\kappa$ a function updating the evaluations $v(x)\ \forall\ x \in Var$; $g$ is a boolean combination of inequalities $\xi(c) \sim k$ with $c \in C$, and $k \in \mathbb{Q}$, and $\sim \in \{<, \leq, =, \geq, >\}$;
- and a function $inv : Z \to 2^{\mathbb{P}(C)}$ that assigns a boolean combination of propositions (the set of which is denoted by $\mathbb{P}(C)$) to each state $z \in Z$.

A *run* of $TA$ is a sequence $r_{TA} = ((z_0, V_0, \xi_0), (z_1, V_1, \xi_1), (z_2, V_2, \xi_2) \ldots)$ of *timed states* $(z_i, V_i, \xi_i)$ consisting of a state $z_i$, the set of evaluations $V_i$ of all variables $x \in Var$, and the clock evaluation vector $\xi_i$. The initial timed state is $(z_0, V_0, \xi_0)$ with the initial state $z_0$, a given initial evaluation $V_0$, and $\xi_0 := 0^{n_c}$. A timed state $(z_{i+1}, V_{i+1}, \xi_{i+1})$ follows from $(z_i, V_i, \xi_i)$ for $i \in \mathbb{N}^{\geq 0}$ by:

1) executing $e = (z_i, z_{i+1}, l, g, \xi_i', \kappa) \in E$ iff (a) $g$ is enabled for $V_i$ and $\xi_i'$ with $\xi_i' = \xi_i + \nu \cdot 1^n$, $\nu \in \mathbb{R}^{\geq 0}$, (b) for $l \neq \varepsilon$: if the label is denoted by $l!$ (*sending label*), $e$ is taken independently of synchronizing transitions; if $l$ is a *receiving label*, denoted by $l?$, $e$ is taken only if a transition of another automaton labeled by $l!$ synchronizes; if $l$ is not supplemented by '!' or '?', $e$ can only be taken if all enabled and identically labeled transitions existing in another automaton synchronize; (c) all $\xi_i''$ with $\xi_i \leq \xi_i'' \leq \xi_i'$ fulfill $inv(z_i)$;
2) $\xi_{i+1}$ results from $\rho(\xi_i')$ by either resetting $\xi(c)$ to zero or leaving the evaluation unchanged;
3) $V_{i+1}$ follows from applying $\kappa$ to $v_i(x)$ for all $x \in Var$.

A state $z \in Z$ is labelled as *urgent* and denoted $z^u$, if it has to be left immediately upon being reached. It applies that $inv(z^u)$ is empty, as is the guard of the outgoing transition(s). Furthermore, a transition $e \in E$ can be labelled as *urgent* meaning that it is taken immediately if its guard $g$ is enabled, and $l = \varepsilon$ applies. ♢

## C. Transformation of SFC in TA

The task is now to establish a scheme that transforms an SFC with reduced runs $\tilde{r}_{SFC}$ into a set of timed automata. The following definition is divided into three parts, the first of which introduces an automaton that triggers the update of all other controller automata if the evaluation of any variable has changed. The second part describes how the sequential and parallel executions are mapped into a set of timed automata, and the third part defines the modeling of the actions.

**Definition 6**: Given $M_{SFC}$ according to Def. 2/3 with a grammar $\Gamma$ as introduced in Def. 1, a model consisting of a set of timed automata according to Def. 5 is obtained as follows:

*Part I - Trigger Automaton*: The cyclic update is realized by a timed automaton $TA^{[1]} = (Z^{[1]}, z_0^{[1]}, Lab^{[1]}, Var^{[1]}, C^{[1]}, E^{[1]}, inv^{[1]})$ with $Z^{[1]} = \{z_1, z_2, z_3\}$, $z_0^{[1]} := z_1$, $Lab^{[1]} = \{\zeta, \varepsilon\}$, $Var^{[1]} = \{u\}$, $u \in \{0, 1\}$, $C^{[1]} = \{c\}$, $inv^{[1]}(z_1) = \emptyset$, and $inv^{[1]}(z_2) = inv^{[1]}(z_3) = (\xi(c) \leq \tau_c^{max})$. The set of transitions $E^{[1]} = \{e_1, e_2, e_3\}$ contains: the urgent transition $e_1 = (z_1, z_2, -, g_1, \rho_1, -)$ with $g_1 = (u = 1)$, and $\rho_1 = (\xi(c) := 0)$; $e_2 = (z_2, z_3, \zeta!, -, \rho_1, -)$; $e_3 = (z_3, z_1, \varepsilon, g_3, -, \kappa_1)$ with $g_3 = (\xi(c) \geq \tau_c^{min})$ and a function $\kappa_u$ that assigns $u := 1$ if $V_i \neq V_{i+1}$, and $u := 0$ otherwise ($V_i, V_{i+1}$ and $i$ as in Def. 3); $u$ is also updated by a plant automaton if any $x \in X_{in}$ changes its evaluation.

*Part II - Structure of the Automata*: The string $\Upsilon$ describing $M_{SFC} = (S, s_0, T, \Gamma, X, G, A, \alpha, C)$ according to Def. 1 is parsed starting from the left, and a set $\Delta = \{TA^{[1]}, TA^{[2]}, \ldots, TA^{[k]}, \ldots, TA^{[n_A]}\}$ is generated by:

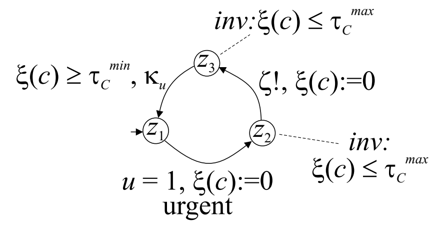
- for $\phi_1$: introduce $TA^{[2]} = (Z^{[2]}, z_0^{[2]}, Lab^{[2]}, Var^{[2]}, C^{[2]}, E^{[2]}, inv^{[2]})$, insert a state $z$ into $Z^{[2]}$ for the first $'\square'$ in $\Upsilon$ and let $z := z_0^{[2]}$. Initialize $Var^{[2]} = C^{[2]} = E^{[2]} = \emptyset$ and $Lab^{[2]} = \{\zeta, \varepsilon\}$.
- for any application of $\phi_2$, assuming that $chain$ follows a step $\square$, for which a state $z'$ has been inserted into $Z^{[k]}$ of automaton $TA^{[k]}$:
  - if $chain =' \perp\square'$: add a new state $z''$ into $Z^{[k]}$ of $TA^{[k]}$; introduce a transition $e = (z', z'', \zeta?, g, \rho, \kappa)$ into $E$ which corresponds to $t = (s', g, s'') \in T$ of $M_{SFC}$, where $g$ is identical in both transitions. (The functions $\rho$ and $\kappa$ are specified in Part III).
  - if $chain =' \triangledown \ldots \vartriangle \square'$ (rule $\phi_3$): for any transition $t = (s', g, s'') \in T$ associated with an alternative branching according to (3.) in the definition of $T$ in Def. 2, add a new state $z''$ into $Z^{[k]}$, and add $e = (z', z'', \zeta?, g, \rho, \kappa)$.

The ordering of the transitions in $T_{ab}$ for $M_{SFC}$ is conveyed into $TA^{[k]}$ correspondingly. Let the final step of an alternative branch $'[\ldots\square\perp]'$ enclosed by $'\triangledown\ldots\vartriangle\square'$ be represented by state $z'' \in Z^{[k]}$ (according to rule $\phi_2$), and add a state $z'''$ into $Z^{[k]}$ to represent the step immediately following $'\vartriangle'$. Insert a transition $e = (z'', z''', \zeta?, g, \rho, \kappa)$ into $E^{[k]}$ for each alternative branch (corresponding to $t = (s, g, s')$ in (4.) of the definition of $T$).

- if $chain =' \perp\blacktriangledown\ldots\blacktriangle'$ (rule $\phi_4$): add two states $z_{u1}$ and $z''$ into $Z^{[k]}$ and the following transitions in $E^{[k]}$: $e = (z', z_{u1}, \zeta?, g, \rho, \kappa)$, where $g$ is the guard assigned to the corresponding $t = (s', g, S'')$ in (5.) of the definition of $T$ in Def. 2, and $e = (z_{u1}, z'', l_1, -, -, -)$. For $chain =' \perp\square'$ following immediately the closure of the parallel branching with $t = (S'', g, s')$ according to (6.) in the definition of $T$ in Def. 2, let $z''' \in Z^{[k]}$ denote the state introduced for the step $'\square'$. Introduce a state $z_{u2}$ into $Z^{[k]}$, as well as $e = (z'', z_{u2}, \zeta?, g, \rho, \kappa)$ and $e = (z_{u2}, z''', l_2, -, -, -)$ into $E^{[k]}$ of $TA^{[k]}$. The synchronization labels $l_1$ and $l_2$ are added to $Lab^{[k]}$. For each parallel branch $'[\square\ldots]'$ enclosed by $'\blacktriangledown\ldots\blacktriangle'$, a separate automaton $TA^{[k+1]} = (Z^{[k+1]}, z_0^{[k+1]}, Lab^{[k+1]}, Var^{[k+1]}, C^{[k+1]}, E^{[k+1]}, inv^{[k+1]})$ is introduced into $\Delta$ with the following initialization: $Z^{[k+1]} = \{z_{in}^{[k+1]}, z^{[k+1]}\}$, $Lab^{[k+1]} = \{\zeta, l_1, l_2\}$, $Var^{[k+1]} = C^{[k+1]} = E^{[k+1]} = \emptyset$. In $Z^{[k+1]}$, $z_{in}^{[k+1]} = z_0^{[k+1]}$ denotes the inactivity of the parallel branch and $z^{[k+1]}$ represents the first step of the latter. A transition $e = (z_{in}^{[k+1]}, z^{[k+1]}, l_1, -, -, -)$ is added to $E^{[k+1]}$ and associated with $t = (s', g, S')$ of $M_{SFC}$; likewise, $e = (z^*, z_{in}^{[k+1]}, l_2, -, -, -)$ is included in $E$. This transition represents the transition $t = (S'', g, s')$ in $M_{SFC}$ by which the parallel branching is left, and it leads from $z^*$ (representing the last step of the parallel branch) into the state of inactivity $z_{in}^{[k+1]}$.
- if $chain =' loop'$ (rule $\phi_5$), i.e. $\Upsilon$ contains $\square\perp\overline{\circlearrowleft}\square\ldots\perp\square\perp\underline{\circlearrowleft}\perp\square$: according to (2.) in the definition of $T$, insert a state $z'$ into $Z^{[k]}$ to represent the first step inside the loop, and add $e = (z, z', \zeta?, g, \rho, \kappa)$ to $E^{[k]}$ where $z'$ corresponds to the step preceding $\perp\overline{\circlearrowleft}$. Let $z''$ denote the state introduced (via the case $chain =' \perp\square'$) as the last step inside the loop, and let $z'$ be the first step inside of the loop, as well as $z'''$ the first step after the loop. Insert the following two transitions into $E$: $e = (z'', z', \zeta?, g'', \rho, \kappa)$, and $e = (z'', z''', \zeta?, g''', \rho, \kappa)$ with their respective guards $g''$ and $g'''$ as in $M_{SFC}$.

*Part III - Actions*: Depending on the actions $a \in A$ of $M_{SFC}$, the components $Var^{[k]}$, $C^{[k]}$, and $inv^{[k]}$, as well as $g$, $\rho$, and $\kappa$ for each $e \in E^{[k]}$ are determined as follows:

- For $TA^{[k]} \in \Delta$: if for any $z \in Z^{[k]}$, there exists a corresponding $s \in S$ for $M_{SFC}$ with $b = \alpha(s)$: iff $a = (q, \tau, o, f) \in b$, where $o$ is a variable $x \in X_{int} \cup X_{out}$ or a function of $x \in X_{in} \cup X_{int}$, then $x$ is included in $Var^{[k]}$.



$inv$: $\xi(c) \le \tau_C^{max}$
$\xi(c) \ge \tau_C^{min}$, $\kappa_u$
$z_3$
$\zeta!$, $\xi(c)$:=0
$z_1$
$z_2$
$inv$: $\xi(c) \le \tau_C^{max}$
$u = 1$, $\xi(c)$:=0
urgent

Fig. 3. Trigger automaton $TA^{[1]}$ according to Def. 6, part I.

- For $s \in S$, $a = (q, \tau, o, f) \in b(s)$ with $q \in \{P, P1\}$, and a corresponding state $z \in Z^{[k]}$ of $A^{[k]}$ which is reached by $e = (z', z, \zeta?, g, \rho, \kappa)$, the manipulation of variables $x \in X_{int} \cup X_{out}$ by $a$ is included in $\kappa$; for $q \in \{P, P0\}$, and a transition $e = (z, z'', \zeta?, g'', \rho'', \kappa'')$ by which the state $z \in Z^{[k]}$ is left, the variable manipulation by $a$ is included in $\kappa''$.
- Given $s \in S$, $a = (q, \tau, o, f) \in b(s)$ with $q \notin \{P, P1, P0\}$, and the corresponding state $z \in Z^{[k]}$ of $A^{[k]}$ which is reached by $e = (z', z, \zeta?, g, \rho, \kappa)$, add a *step variable* $sv$ into $Var^{[k]}$, add the assignment $sv := 1$ and the manipulation of the operand $o$ by the action $a$ to $\kappa$ of $e$. (Note: if $s$ is the first step following a parallel branching, $sv$ is assigned to the state reached by the inactivity state $z_{in}$ of the branch.) Furthermore, the assignment $sv := 0$ is added to $\kappa$ of the transition by which $z$ is left. For any $a = (q, \tau, o, f) \in b(s)$ with $q \notin \{P, P1, P0\}$, a new automaton $TA^{[k]} = (Z^{[k]}, z_0^{[k]}, Lab^{[k]}, Var^{[k]}, C^{[k]}, E^{[k]}, inv^{[k]})$ is added to $\Delta$ with $z_0^{[k]} = z_1$, $Lab^{[k]} = \{\zeta, \varepsilon\}$, $Var^{[k]} = \{sv, f\} \cup \{X^{[k]}\}$ with $X^{[k]} \subset X$ denoting the variables that are relevant for $a$. Depending on the qualifier $q$, $TA^{[k]}$ is parameterized as follows[2]:
  - $q = N$: $Z^{[k]} = \{z_1\}$, $C^{[k]} = \emptyset$, $E^{[k]} = \{e_1\}$ with $e_1 = (z_1, z_1, \zeta?, (sv = 1 \wedge f = 1), -, \kappa_1)$, $e_2 = (z_1, z_1, \zeta?, (sv = 0 \vee f = 0), -, \kappa_0)$ where $\kappa_1$ updates the action operand according to $a$, $\kappa_0$ resets the operand to a default value, and $inv^{[k]} = \emptyset$.
  - $q = St$: $Z^{[k]} = \{z_1, z_2\}$, $C^{[k]} = \emptyset$, $E^{[k]} = \{e_1, e_2, e_3\}$ with $e_1 = (z_1, z_2, \zeta?, (sv = 1 \wedge f = 1), -, \kappa_1)$, $e_2 = (z_2, z_2, \zeta?, (f = 1), -, \kappa_1)$, $e_3 = (z_2, z_1, \zeta?, (f = 0), -, \kappa_0)$, and $inv^{[k]} = \emptyset$.
  - $q = D$: $Z^{[k]} = \{z_1, z_2\}$, $C^{[k]} = \{c\}$, $E^{[k]} = \{e_1, e_2, e_3\}$ with $e_1 = (z_1, z_2, \zeta?, (sv = 1 \wedge f = 1), (\xi(c) := 0), -)$, $e_2 = (z_2, z_2, \zeta?, (sv = 1 \wedge f = 1 \wedge \xi(c) \ge \tau_v), -, \kappa_1)$, $e_3 = (z_2, z_1, \zeta?, (sv = 0 \vee f = 0), -, \kappa_0)$, and $inv^{[k]}(z_1) = inv^{[k]}(z_2) = \emptyset$.
  - $q = DS$: $Z^{[k]} = \{z_1, z_2, z_3\}$, $C^{[k]} = \{c\}$, $E^{[k]} = \{e_1, \ldots, e_5\}$ with $e_1 = (z_1, z_2, \zeta?, (sv = 1 \wedge f = 1), \xi(c) := 0, -)$, $e_2 = (z_2, z_1, \zeta?, sv = 0 \wedge f = 0, -, \kappa_0)$, $e_3 = (z_2, z_3, \zeta?, sv = 1 \wedge f = 1 \wedge \xi(c) \ge \tau_v, -, \kappa_1)$, $e_4 = (z_3, z_3, \zeta?, f = 1, -, \kappa_1)$, $e_5 = (z_3, z_1, \zeta?, f = 0, -, \kappa_0)$, and $inv^{[k]}(z_1) = inv^{[k]}(z_3) = \emptyset$, $inv^{[k]}(z_2) = (\xi(c) \le \tau_v)$. $\diamondsuit$

The trigger automaton obtained from Part I of the definition is shown in Fig. 3. The urgent transition from $z_1$ to

[2]Due to space limitations, only a few qualifiers are listed here; the remaining qualifiers are modeled correspondingly.

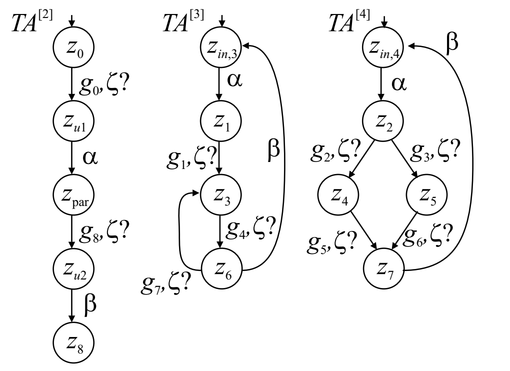


Fig. 4. Automaton structure for the SFC shown in Fig. 2.

$z_2$ is executed whenever $u = 1$ signals that the evaluation of the variables have changed, provided the last triggering event ($\zeta!$) occurred at least the time $\tau_C^{min}$ ago.

For the example in Fig. 2, Part II of Def. 6 leads to the set of automata shown in Fig. 4. The states $z_{u1}$ and $z_{u2}$ in $TA^{[2]}$ are urgent to achieve that the transition labelled by $\alpha$ and $\beta$ are executed immediately after the conditions $g_0$ and $g_8$ become true. The state $z_{par}$ represents that the parallel branching is active, i.e. only when $TA^{[2]}$ is in $z_{par}$, the automata $TA^{[3]}$ and $TA^{[4]}$ leave their states of inactivity $z_{in,4}$, and $z_{in,5}$ respectively.

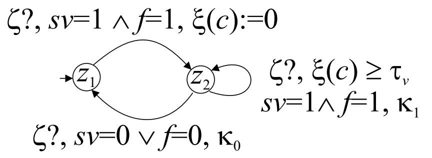
As an example for modeling the actions, Fig. 5 illustrates the timed automaton introduced for the action qualifier $q = D$. The initial state $z_1$ is left, if the corresponding step is reached ($sv = 1$), and the execution flag of the action is set ($f = 1$). The manipulation of the operand by $\kappa_1$ is performed after the time $\tau_v$, unless the step is left ($sv = 0$) or the action is deactivated ($f = 0$) before. All transitions are synchronized with $TA^{[1]}$ by the label $\zeta$.

**Lemma 2**: Given $M_{SFC}$ according to Def. 1-3 with an arbitrary reduced run $\tilde{r}_{SFC}$, and let $\Delta = \{TA^{[1]}, \ldots, TA^{[n_A]}\}$ be a set of $TA$ derived according to Def. 6. The set of runs possible for the parallel composition of $TA^{[1]}, \ldots, TA^{[n_A]}$ contains a run that is semantically equivalent to $\tilde{r}_{SFC}$. ■

The proof of this lemma is out of the scope of a conference paper, however, the proof scheme is as follows: (a) it is shown first that every step of $M_{SFC}$ is mapped onto a state of the parallel composition $TA_{comp} = TA^{[1]} \| \ldots \| TA^{[n_A]}$, where $n_A$ is the total number of automata introduced according to Def. 6; (b) it is shown for all possible transitions $t \in T$ executed in $M_{SFC}$ that a corresponding transition $e$ between the respective states exists for $TA_{comp}$, and that the time of occurrence of $t$ is possible for $e$ in $TA_{comp}$; (c) for all actions manipulating an operand $o$ (and thus modifying $V_i$ according to $\tilde{r}_{SFC}$), it is shown that the action automata included in $TA_{comp}$ manipulate the evaluations of the variables $x \in X_{int} \cup X_{out}$ correspondingly.

## IV. CONCLUSIONS

The paper has introduced a formal definition of SFC, the reduction of runs of $M_{SFC}$ to configurations that are relevant



$\zeta?, sv{=}1 \wedge f{=}1, \xi(c){:=}0$
$z_1$
$z_2$
$\zeta?, \xi(c) \geq \tau_v$
$sv{=}1 \wedge f{=}1, \kappa_1$
$\zeta?, sv{=}0 \vee f{=}0, \kappa_0$

Fig. 5. Automaton for the delay action $q = D$.

for verification, and a transformation procedure into timed automata. Due to lemma 2, the TA model can be used for safety verification in the following sense: if $M_{SFC}$ leads to control signals that drive a plant model into an unsafe state, the verification will reveal that also $TA_{comp}$ produces the same control signal, and thus the same verification result.

The TA model encodes a superset of the control signals of $M_{SFC}$, due to the following fact: sending the label $\zeta!$ by the trigger automaton is not bound to the actual cycle time $\tau_{c,i}$ but can occur in the complete interval $[\tau_c^{min}, \tau_c^{max}]$. Thus, the verification of $TA_{comp}$ may lead to the result that the safety property is violated while it is not for $M_{SFC}$. However, from a practical point of view and assuming small values of $\tau_c^{max}$, the SFC is 'close' to violating the property in this case, and a modification of the logic controller is recommended anyway.

Current work aims at implementing a tool that realizes the transformation algorithmically.

## REFERENCES

[1] Int. Electrotechn. Commission (Techn. Com. No. 65), *Programmable Controllers - Programming Languages, IEC 61131-3*, 2003.

[2] E. Clarke, O. Grumberg, and D. Peled, *Model Checking*. MIT Press, 1999.

[3] K. Fujino, K. Imafuku, Y. Yamashita, and H. Nishitani, "Design and verification of the SFC program for sequential control," *Comp. Chem. Eng.*, vol. 24, pp. 303–308, 2000.

[4] S. Bornot, R. Huuck, Y. Lakhnech, and B. Lukoschus, "Verification of sequential function charts using SMV," in *Int. Conf. on Parallel and Distributed Processing Techn. and Applic.*, 2000, pp. 2987–2993.

[5] S. Lampérière and J. Lesage, "Formal verification of the sequential part of PLC programs," in *Discrete Event Systems*. Kluwer Acad. Publ., 2000, pp. 247–254.

[6] D. L'Her, P. Le Parc, and L. Marce, "Proving sequential function chart programs using automata," in *3rd Int. Workshop on Automata Implementation*, ser. Springer-LNCS, vol. 1660, 1998, pp. 149–163.

[7] M. Remelhe, S. Lohmann, O. Stursberg, and S. Engell, "Algorithmic verification of logic controllers given as sequential function charts," in *IEEE Conf. on Comp.-Aided Control System Design*, 2004, pp. 53–58.

[8] N. Bauer, S. Engell, R. Huuck, S. Lohmann, B. Lukoschus, M. Remelhe, and O. Stursberg, "Verification of PLC programs given as sequential function charts," in *Integration of Software Spec. Techn. for Applic. in Eng.*, ser. Springer-LNCS, vol. 3147, 2004, pp. 517–540.

[9] G. Behrmann, A. David, K. Larsen, O. Moeller, P. Pettersson, and W. Yi, "UPPAAL - present and future," in *40th IEEE Conf. on Decision and Control*, 2001, pp. 2881–2886.

[10] O. Stursberg, A. Fehnker, and Z. H. und B.H. Krogh, "Specification-guided analysis of hybrid systems using a hierarchy of validation methods," in *IFAC Conf. on Analysis and Design of Hybrid Systems*, 2003, pp. 289–295.

[11] K. John and M. Tiegelkamp, *IEC 61131-3: Programming Industrial Automation Systems*. Springer, 2001.

[12] N. Bauer, R. Huuck, B. Lukoschus, and S. Engell, "A unifying semantics for sequential function charts," in *Integration of Software Spec. Techn. for Applications in Eng.*, ser. Springer-LNCS, vol. 3147, 2004, pp. 400–418.

[13] O. Stursberg, S. Lohmann, and S. Engell, "Improving dependability of logic controllers by algorithmic verification," in 16$^{th}$ *IFAC World Congress*, no. Mo-E17-TO/6, 2005.