# A Transformational Approach for Generating Non-Linear Invariants*

S. Bensalem[1], M. Bozga[1], J.-C. Fernandez[2], L. Ghirvu[1], and Y. Lakhnech[1**]

[1] VERIMAG,
Centre Equation 2, avenue de Vignate F-38610 Gieres, France. Name@imag.fr
[2] LSR,
681, rue de la Passerelle 38402 Saint Martin d'Hres Cedex, France.
Fernandez@imag.fr

**Abstract.** Computing invariants is the key issue in the analysis of infinite-state systems whether analysis means testing, verification or parameter synthesis. In particular, methods that allow to treat combinations of loops are of interest. We present a set of algorithms and methods that can be applied to characterize over-approximations of the set of reachable states of combinations of self-loops. We present two families of complementary techniques. The first one identifies a number of basic cases of pair of self-loops for which we provide an exact characterization of the reachable states. The second family of techniques is a set of rules based on static analysis that allow to reduce $n$ self-loops ($n \geq 2$) to $n-1$ independent pairs of self-loops. The results of the analysis of the pairs of self-loops can then be combined to provide an over-approximation of the reachable states of the $n$ self-loops. We illustrate our methods by synthesizing conditions under which the Biphase Mark protocol works properly.

## 1 Introduction

This paper proposes techniques for computing over-approximations of the set of reachable states of a class of infinite state systems. The systems we consider are systems whose variables can be seen as *counters* that can be incremented by positive or negative constants or can be reset to some constant.

The problem of computing invariants of arithmetical programs in particular, and infinite state systems in general, has been investigated from the seventies. Abstract interpretation [CC77,CC92] is a precise and a formal framework which has been used to develop techniques to tackle this problem. As pioneering work in this field, one can mention M. Karr's

* Work partially supported by Région Rhône-Alpes, France
** Contact author

work [Kar76] based on constant propagation for computing invariants that are systems of affine equations, P. & R. Cousot's work [CC76] which uses interval analysis to compute invariants of the form $x \in [a, b]$, $x \leq a$, etc., and the work by P. Cousot and N. Halbwachs [CH78] which provides techniques that allow to compute linear constraints that relate the program variables.

In recent years, the subject has known a renewal of interest with the development of symbolic model-checking techniques for some classes of infinite state systems as timed and hybrid automata [HNSY92,HPR94], finite communicating automata [BG96,ABJ98], parameterized networks [KMM$^+$97,ABJN99,BBLS], and automata with counters [BGP97,WB98].

In this paper, we consider transition systems with finite control and with counters as data variables. A transition consists of a guard and a set of assignments. A guard is given by a Presburger formula that may contain parameters, that is, variables that are neither initialized nor modified during execution. Assignments may increment the counters by positive or negative constants or set them to constant values. It should be noticed that this model is fairly general. Indeed, it is computationally equivalent to Turing machines and syntactically subsumes Timed Automa [AD94], Petri Nets with inhebitors, and Datalog Programs [FO97]. Indeed, each of these models can easily translated into our transition systems.

Given a transition syste we are interested in computing over-approximations of the set of reachable states from *parametric* initial states, that is, states of the form $\bar{x} = \bar{x}_0$, where $\bar{x}$ are the variables of the system and $\bar{x}_0$, are freeze variables (also called inactive auxiliary variables). In contrast to almost all the works mentioned above, the techniques we present allow to derive non-linear invariants. We concentrate on characterizing sets of states reachable by $n$-self-loops. This is not an essential restriction, since every system can be transformed into one with a single control location. Moreover, several spécification and programming languages such as UNITY [KJ89] or the synchronous language Lustre [CHPP87] consist of programs where all transitions are self-loops of a single control point. Notice also that it is clear that the combined effect of self-loops cannot in general be characterized by linear constraints. We present two families of complementary techniques. The first one is presented as set of results that identify a number of basic cases of pairs of self-loops for which we provide an exact characterization of the reachable states. The second family of techniques is a set of rules based on static analysis that allow to reduce $n$ self-loops ($n \geq 2$) to $n-1$ independent pairs of self-loops. The results of the analysis of the pairs of self-loops can then be combined to provide an over-approximation of the reachable states of the $n$ self-loops.

The reduction techniques we present are in the same line as the decomposition rules presented by Fribourg and Olsèn in [FO97], where they consider Datalog programs, i.e., transition systems consisting of a single control location and counters and where only $x > 0$ is allowed as guard. Notable differences are, however, the fact that the systems they consider are syntactically more restricted and that their rules are exact.

To illustrate the techniques we present in this paper, we consider the Biphase mark protocol which is a parameterized protocol used as a convention for representing both a string of bits and clock edges in a square wave. Using our techniques we have been able to provide a full parametric analysis of this protocol.

## 2 Preliminaries

We assume an underlying assertion language $\mathcal{A}$ that includes first-order predicate logic and interpreted symbols for expressing the standard operations and relations over some concrete domains. We assume to have the set of integers among these domains. Assertions (we also say predicates) in $\mathcal{A}$ are interpreted in states that assign values to the variables of $\mathcal{A}$. Given a predicate $P$, we denote by $\mathit{free}(P)$ the set of free variables occurring in it. Similarly, if $e$ is an expression in $\mathcal{A}$, we also write $\mathit{free}(e)$ to denote the set of all variables which occur in $e$. As expressiveness is not our issue in this paper, we will tacitly identify a predicate with the set of its models.

As computational model we use transition systems. We restrict ourselves to transition systems where the expressions occurring in an assignment to variables $x$ are either constants or of the form $x + k$. Thus, a transition system is given by a tuple $(\mathcal{X}, \mathcal{Q}, \mathcal{T}, \mathcal{E}, \Pi)$ where $\mathcal{X}$ is a finite set of typed data variables, $\mathcal{Q}$ is a finite set of control locations, $\mathcal{T}$ is a finite set of transition names, $\mathcal{E}$ associates with each transition $\tau$ a pair $(\mathcal{E}_1(\tau), \mathcal{E}_2(\tau))$ consisting of a source and a target control location, and $\Pi$ associates with each transition a guard $\mathit{gua}(\tau)$ which is an assertion in the Presburger fragment of $\mathcal{A}$ with free variables in $\mathcal{X}$ and a list $\mathit{affe}(\tau)$ of assignments of the form $x := x + k$ or $x := k$ with $x \in \mathcal{X}$ and $k \in \mathbb{Z}$ and such that for each $x \in \mathcal{X}$ there is at most one assignment $x := e$ in $\mathit{affe}(t)$. We denote by $\mathrm{Base}(\tau)$ the set of variables occurring in $\tau$. Notice that we allow *parameters* in the guards of the transitions; parameters can be seen as program variables that are not modified during execution. This allow us to model parameterized protocols as the Biphase protocol, which we consider later on, and to analyze these protocols using our techniques.

Clearly, $(\mathcal{Q}, \mathcal{T}, \mathcal{E})$ builds a labeled graph which we call the *control graph*. Henceforth, we denote the set of transitions $\tau$ with $\mathcal{E}_1(\tau) = \mathcal{E}_2(\tau) = q$ by $L(q)$, i.e., $L(q)$ is the set of self-loops in $q$. Moreover, we write $\tau(\bar{x})$, where $\bar{x}$ is a set of variables, for the projection of $\tau$ on $\bar{x}$, that is, the transition whose guard is obtained from the guard of $\tau$ by existentially quantifying all variables but $\bar{x}$ and whose assignments are obtained from $\tau$ by removing all assignments to other variables than $\bar{x}$.

A transition $\tau$ induces a relation $\xrightarrow{\tau}$ on configurations which are pairs of control locations and valuations of the variables in $\mathcal{X}$. Given a transition $\tau$, and configurations $(q, s)$ and $(q', s')$, $(q', s')$ is called *$\tau$-successor of* $(q, s)$, denoted by $(q, s) \xrightarrow{\tau} (q', s')$, if $\mathcal{E}(\tau) = (q, q')$, $s$ satisfies $gua(\tau)$ and $s'$ satisfies $s'(x) = s(e)$, for each $x := e$ in $affe(\tau)$, $s'(x) = s(x)$, for each $x$ that is not affected by $\tau$. Given a regular language $L$ over $\mathcal{T}$ and given configurations $(q, s)$ and $(q', s')$, we say that $(q', s')$ is $L$-reachable from $(q, s)$, denoted by $(q, s) \xrightarrow{L} (q', s')$, if there exists a word $\tau_1 \cdots \tau_n \in L$ and configurations $(q_i, s_i)_{i \leq n}$ such that $(q_0, s_0) = (q, s)$, $(q_n, s_n) = (q', s')$, and $(q_i, s_i) \xrightarrow{\tau_i} (q_{i+1}, s_{i+1})$. If $\varphi$ and $\varphi'$ are predicates, we write $\varphi \xrightarrow{L} \varphi'$ to denote the fact that there exists a state $s$ that satisfies $\varphi$ and a state $s'$ that satisfies $\varphi'$ such that $s \xrightarrow{L} s'$. Identifying, a state with a predicate characterizing it, we also use the notations $\varphi \xrightarrow{L} s'$ and $s \xrightarrow{L} \varphi'$, respectively. Henceforth, given a control location $q$, in case all transitions in $L$ have $q$ as source and target locations, we omit mentioning $q$ in configurations. Furthermore, given a predicate $\varphi(\bar{x}_0, \bar{x})$, where $x_0$ are freeze variables (also called inactive auxiliary variables), and given a set $L \subseteq L(q)$ of self-loops, we say that $\varphi(\bar{x}_0, \bar{x})$ is an $L$-invariant at $q$, if for every state $s'$ that is $L$-reachable from a state $s$, $\varphi[s(\bar{x})/\bar{x}_0, s'(\bar{x})/\bar{x}]$ is valid. Thus, $\varphi(\bar{x}_0, \bar{x})$ is the set of states reachable from a parametric state $\bar{x} = \bar{x}_0$ by taking sequences of transitions in $L$. The predicate $\varphi(\bar{x}_0, \bar{x})$ corresponds to the strongest postcondition of so-called *most general formulas* used in [Gor75] and investigated in [AM80] in the context of axiomatic verification of recursive procedures.

## 3 Characterizing reachable states of self-loops

Throughout this section, we fix a transition system $\mathcal{S} = (\mathcal{X}, \mathcal{Q}, \mathcal{T}, \mathcal{E}, \Pi)$. Our goal is to transform $\mathcal{S}$ into a transition system $\mathcal{S}^{\#}$ such that $\mathcal{S}^{\#}$ does not contain self-loops and such that the set of states reachable from a state

$s$ in $\mathcal{S}^\#$ is a super-set of the set of states reachable from $s$ in $\mathcal{S}$, that is, $\mathcal{S}^\#$ is an abstraction of $\mathcal{S}$ [CC77]. Thus, we will entirely concentrate on self-loops. The motivation and justification behind this is many-fold. First, it is obvious that our model is as expressive as Turing machines, since a two counter-machine is trivially encoded in this model. Moreover, arithmetical programs, which can easily encoded in our model, represent an interesting class of programs that have been widely investigated starting with the pioneering work [CH78]. Moreover, even if we restrict the control graph to a single node, we obtain, as discussed in [FO97], an interesting class of Datalog programs. Our model allows to encode in a natural way Petri Nets with inhibitors.

The main idea behind the transformation of $\mathcal{S}$ into $\mathcal{S}^\#$ is the following. Consider a control location $q$ and let $\varphi(\bar{x}_0, \bar{x})$ be an $L(q)$-invariant at $q$. Then, we obtain $\mathcal{S}^\#$ by applying the following transformations:

1. Add a new list of variables $\bar{x}_0$ with the same length as $\bar{x}$.
2. Remove all transitions in $L(q)$.
3. Let $\tau_1, \cdots, \tau_n$ be all transitions with $\mathcal{E}_2(\tau_i) = q$ and let $\bar{x} := \bar{e}_i$ be the assignment associated to $\tau_i$. Add to $\bar{x} := \bar{e}_i$ the assignment $\bar{x}_0 := \bar{e}_i$.
4. Replace each assignment $\bar{x} := \bar{e}$ of a transition $\tau$ with $\mathcal{E}_1(\tau) = q$ and $\mathcal{E}_2(\tau) \neq q$, by the predicate $\exists \bar{y} \cdot \varphi(\bar{x}_0, \bar{y}) \wedge gua(\tau) \wedge \bar{x}' = \bar{e}[\bar{y}/\bar{x}]$, where $\bar{x}'$ stands for the state variables after taking the transition. Note that $\mathcal{S}^\#$ does not satisfy the syntactic restrictions on assignments as introduced in Section2; it is, however, a transition system in the usual sense.

It is not difficult to check that $\mathcal{S}^\#$ is indeed an abstraction of $\mathcal{S}$. Notice also that in case all predicates $\varphi(\bar{x}_0, \bar{x})$ used in the transformation for characterizing reachable states by self-loops are exact, the obtained system $\mathcal{S}^\#$ is then an exact abstraction of $\mathcal{S}$.

Our approach in computing invariants characterizing the effect of a set of loops is based on the particular case of two self-loops that satisfy syntactic conditions that allow us to analyze each self-loop in isolation and on a set of static analysis techniques which allow us to reduce the analysis of $n$ self-loops to the analysis of a number of particular cases.

Given two transitions $\tau_0$ and $\tau_1$ with $\mathrm{Base}(\tau_0) = \bar{x}$ and $\mathrm{Base}(\tau_1) = \bar{x}\bar{y}$, where $\bar{x}$ and $\bar{y}$ are two disjoint sets of variables, and such that $\bar{x}$ is assigned the list $\bar{c}$ of constants in $\tau_1$. We say that $\tau_0$ *enables* $\tau_1$, if for every state $s$ with $s(\bar{x}) = \bar{c}$, there exists a state $s'$ such that $s \xrightarrow{\tau_0^*} s'$ and $s'$ satisfies the projection on $\bar{x}$ of the guard of $\tau_1$, i.e., $s'$ satisfies $\exists \bar{y} \cdot gua(\tau_1)$. Notice that $\tau_0$ does not enable $\tau_1$ iff for every state $s$ with $s(\bar{x}) = \bar{c}$, there is no state $s'$ such that $s \xrightarrow{\tau_0^*} s'$ and $s'$ satisfies $\exists \bar{y} \cdot gua(\tau_1)$.

**Lemma 1.** *Let $\tau_0$ and $\tau_1$ be two transitions such that $Base(\tau_0) = \bar{x}$, $Base(\tau_1) = \bar{x}\bar{y}$, where $\bar{x}$ and $\bar{y}$ are two disjoint sets of variables, and such that $\bar{x}$ is assigned the list $\bar{c}$ of constants in $\tau_1$.*

*Then, $s \xrightarrow{(\tau_0+\tau_1)^*} s'$ iff $s \xrightarrow{\tau_0^*} s'$ or there exists a state $s''$ such that 1) $s \xrightarrow{\tau_0^*\tau_1} s''$, 2) $\bar{x} = \bar{c} \xrightarrow{\tau_0^*} s'(\bar{x})$ and 3) one of the following conditions holds:*

1. *$\tau_0$ enables $\tau_1$ and $s(\bar{y}) \xrightarrow{\tau_1(\bar{y})^*} s'(\bar{y})$ or*
2. *$\tau_0$ does not enable $\tau_1$ and $s(\bar{y}) \xrightarrow{\tau_1} s'(\bar{y})$.*

□

*Proof.* We prove the implication from left to right by induction on the number of times transition $\tau_1$ is taken from $s$ to $s'$. Thus, suppose we have $s \xrightarrow{(\tau_0+\tau_1)^*} s'$. The induction basis follows immediately, since then we have $s \xrightarrow{\tau_0^*} s'$. Suppose now that $\tau_1$ is taken $n$ times with $n > 0$. Then, we have $s \xrightarrow{\tau_0^*} s_1 \xrightarrow{\tau_1} s'' \xrightarrow{(\tau_0+\tau_1)^*} s'$ and $\tau_1$ is taken $n-1$ times in the computation from $s''$ to $s'$. In case, $\tau_0$ does not enable $\tau_1$, we have $s'' \xrightarrow{\tau_0^*} s'$. Hence, since $\bar{y} \cap \mathrm{Base}(\tau_0) = \emptyset$, $s'(\bar{y}) = s''(\bar{y})$ and $\bar{x} = \bar{c} \xrightarrow{\tau_0^*} s'(\bar{x})$. That is, $s(\bar{y}) \xrightarrow{\tau_1} s'(\bar{y})$ and $\bar{x} = \bar{c} \xrightarrow{\tau_0^*} s'(\bar{x})$.
Now, suppose that $\tau_0$ enables $\tau_1$, then, by induction hypothesis, $s''(\bar{y}) \xrightarrow{\tau_1(\bar{y})^*} s'(\bar{y})$. Since, $\bar{y} \cap \mathrm{Base}(\tau_0) = \emptyset$, $s(\bar{y}) = s_1(\bar{y})$. Consequently, $s(\bar{y}) \xrightarrow{\tau_1(\bar{y})^*} s'(\bar{y})$. Moreover, by induction hypothesis, $\bar{x} = \bar{c} \xrightarrow{\tau_0^*} s'(\bar{x})$.

□

Lemma 1 states conditions under which the set of states reachable by repeated execution of the transitions $\tau_0$ and $\tau_1$ can be exactly characterized by *independently* considering the values of the variables $\bar{x}$ that can be reached by applying $\tau_0$ and the values of $\bar{y}$ that can be reached by applying $\tau_1$.

In the following, we present a lemma that allows us to apply a decomposition similar to Lemma 1 while allowing $\tau_0$ to contain additional variables $\bar{z}$ disjoint from $\bar{x}$ and $\bar{y}$ that are not modified by $\tau_1$.

**Lemma 2.** *Let $\tau_0$ and $\tau_1$ be two transitions such that $Base(\tau_0) = \bar{x}\bar{z}$, $Base(\tau_1) = \bar{x}\bar{y}$, where $\bar{x}$, $\bar{y}$ and $\bar{z}$ are mutually disjoint sets of variables, and such that the following conditions are satisfied:*

1. *For every state $s'$, if true $\xrightarrow{\tau_1} s'$ then $s'$ does not satisfy the guard of $\tau_1$.*
2. *$\bar{x}$ is assigned the list $\bar{c}$ of constants in $\tau_1$.*
3. *There is a list $\bar{c}'$ of constants such that, for every states $s$ and $s'$ with $s(\bar{x}) = \bar{c}$ and $s \xrightarrow{\tau_0^*} s'$, if $s'$ satisfies the guard of $\tau_1$ then $s'(\bar{z}) = s(\bar{z}) + \bar{c}'$.*
4. *For every state $s$ with $s(\bar{x}) = \bar{c}$ there is a state $s'$ such that $s \xrightarrow{\tau_0^*} s'$ and such that $s'$ satisfies the projection on $\bar{x}$ of the guard of $\tau_1$.*
5. *For all states $s$ and $s'$ with $s(\bar{x}) = s'(\bar{x}) = \bar{c}$ and for all $k \geq 0$, $s \xrightarrow{\tau_0^k}$ true iff $s' \xrightarrow{\tau_0^k}$ true.*

*Then, $s \xrightarrow{(\tau_0+\tau_1)^*} s'$ iff $s \xrightarrow{\tau_0^*} s'$ or there exists a state $s''$ such that*

1. *$s \xrightarrow{\tau_0^*\tau_1} s''$, $\bar{x} = \bar{c} \xrightarrow{\tau_0(x)^*} s'(\bar{x})$ and*
2. *there exists $k \in \mathbb{N}$ and a state $s'''$ with $s'''(\bar{z}) = s''(\bar{z}) + k * \bar{c}$, $s(\bar{y}) \xrightarrow{\tau_1^{k+1}} s'(\bar{y})$, and $s'''(\bar{z}) \xrightarrow{\tau_0^*} s'(\bar{z})$.*

□

*Proof.* (sketch).

Using Condition 1.,one can prove that $s \xrightarrow{(\tau_0+\tau_1)^*} s'$ iff $s \xrightarrow{\tau_0^*} s'$ or there are states $s''$ and $s'''$ and $k \geq 0$ such that

$$s \xrightarrow{\tau_0^*\tau_1} s'' \xrightarrow{(\tau_0^+\tau_1)^k} s''' \xrightarrow{\tau_0^*} s'.$$

Let us consider the second case. Here, by Condition 2., we have $s''(\bar{x}) = \bar{c}$. Hence, by Condition 3., in any state reachable from $s''$ by applying $(\tau_0^+\tau_1)$ $k'$-times, the value of $\bar{z}$ is $s''(\bar{z}) + k' * \bar{c}'$. Therefore, $s'''(\bar{z}) = s''(\bar{z}) + k * \bar{c}$.

Notice that Condition 4., is used to prove the "only if" part of the statement. Condition 5. guarantees that $s'''$ is reachable from $s''$ by $(\tau_0^+\tau_1)$ $k$-times. It also guarantees that the number of times $\tau_0$ can be taking starting in a state satisfying $\bar{x} = \bar{c}$ does not depend on $\bar{z}$.

□

*Remark 1.* It is important to notice that Condition 2 is syntactic, so it can be easily checked. Moreover, the remaining conditions can be checked effectively, since the sets of reachable states involved are expressible in Presburger arithmetic. Indeed, if a language $L$ is of the form $L_1 + \cdots + L_n$, where each $L_i$ is either finite or of the form $w^*$, where $w$ is a word, then the set of states reachable by $L$ from a (parametric) state $\bar{x} = \bar{x}_0$ is easily expressible in Presburger arithmetic. Nevertheless, it is easy to give sufficient syntactic conditions that can be easily checked. For instance, Condition 5. is satisfied, if $\bar{z}$ does not occur in the guard of transition $\tau_0$.

*Example 1.*
Let us consider the following self-loops:

$$\begin{cases} \tau_0 : x < T & \rightarrow x := x+1; z := z+1 \\ \tau_1 : x = T \wedge y < C & \rightarrow x := 0; y := y+1 \end{cases}$$

It is easy to check that the premises of Lemma 2 are satisfied. Using the characterization stated by the lemma and after simplification, we obtain the following invariant:

$$\begin{array}{l} (x - z = x_0 - z_0 \wedge x \geq x_0 \wedge z \geq z_0 \wedge y = y_0) \\ \vee\ \exists k \geq 1 \cdot \\ \qquad (y = y_0 + k \wedge y \leq C \wedge z = (z_0 - x_0) + k * T + x \wedge x \leq T) \end{array}$$

□

Lemma 2 can be generalized as follows to the case where $\bar{z}$ is not augmented by the same list $\bar{c'}$ of constants:

**Lemma 3.** *Assume the same premises as in Lemma 2 but condition 3. replaced by:*

*There is a set $I$ of values such that, for every states $s$ and $s'$ with $s(\bar{x}) = \bar{c}$ and $s \xrightarrow{\tau_0^*} s'$, if $s'$ satisfies the guard of $\tau_1$ then*

**3.a** *there is $\bar{c}' \in I$ with $s'(\bar{z}) = s(\bar{z}) + \bar{c}'$ and*

**3.b** *for every $\bar{c}'' \in I$ there is a state $s''$ with $s''(\bar{z}) = s(\bar{z}) + \bar{c}''$, $s \xrightarrow{\tau_0^*} s''$, and such that $s''$ satisfies the guard of $\tau_1$.*

*Then, $s \xrightarrow{(\tau_0+\tau_1)^*} s'$ iff $s \xrightarrow{\tau_0^*} s'$ or there exists a state $s''$ such that*

1. *$s \xrightarrow{\tau_0^*\tau_1} s''$, $\bar{x} = \bar{c} \xrightarrow{\tau_0(x)^*} s'(\bar{x})$ and*
2. *there exists $k \in I\!N$ and a state $s'''$ with $s'''(\bar{z}) = s''(\bar{z}) + \sum_{i=1}^{i=k} \bar{c}_i$ with $\bar{c}_i \in I$, $s(\bar{y}) \xrightarrow{\tau_1^{k+1}} s'(\bar{y})$, and $s'''(\bar{z}) \xrightarrow{\tau_0^*} s'(\bar{z})$.*

□

*Example 2.*
Let us consider the following self-loops:

$$\begin{cases} \tau_0 : x < T & \rightarrow x := x+1; z := z+1 \\ \tau_1 : t \leq x \leq T \wedge y < C & \rightarrow x := 0; y := y+1 \end{cases}$$

Now, applying Lemma 3 we obtain the following invariant:

$$\begin{array}{l}(x - z = x_0 - z_0 \wedge x \geq x_0 \wedge z \geq z_0 \wedge y = y_0) \\ \vee\ \exists k \geq 1 \cdot \\ \qquad (y = y_0 + k \wedge y \leq C \wedge z \in (z_0 - x_0) + [k * t, k * T] + x \wedge x \leq T)\end{array}$$

$\square$

*Remark 2.* Notice that, if we remove Condition 3.b in Lemma 3, then only the "only if" part of the conclusion is true, that is, we have $s \xrightarrow{(\tau_0+\tau_1)^*} s'$ implies $s \xrightarrow{\tau_0^*} s'$ or there exists a state $s''$ such that

1. $s \xrightarrow{\tau_0^*\tau_1} s''$, $\bar{x} = \bar{c} \xrightarrow{\tau_0^*} s'(\bar{x})$ and
2. there exists $k \in I\!N$ and a state $s'''$ with $s'''(\bar{z}) = s''(\bar{z}) + \sum_{i=1}^{i=k} \bar{c_i}$ with $\bar{c_i} \in I$, $s(\bar{y}) \xrightarrow{\tau_1^{k+1}} s'(\bar{y})$, and $s'''(\bar{z}) \xrightarrow{\tau_0^*} s'(\bar{z})$.

This result can of course be used to derive an invariant that is not necessarily the strongest. $\square$

## 4 Decomposition techniques

We present hereafter heuristics which allow us to reduce the analysis of $n \geq 2$ self-loops to simpler cases such that, finally, we can apply the lemmata introduced in Section 3.

Basically, we consider the case of $n+1$ loosely-coupled self-loops. We show that, their global analysis can be effectively reduced to $n$ analysis of 2 self-loop problems, when some syntactic conditions on the sets of used variables occurs. The decomposition technique is stated by the following lemma and can be seen as a direct generalization of lemma 1.

**Lemma 4.** *Let $\tau_0, \tau_1, \cdots, \tau_n$ be transitions such that $Base(\tau_0) = \bar{x_1} \cdots \bar{x_n}$, $Base(\tau_i) = \bar{x_i}\bar{y_i}$ and for each $i = 1, \cdots, n$, $\bar{x_i}$ is assigned by $\tau_i$ the list $\bar{c_i}$ of constants, and the sets of variables $\bar{x_i}$ and $\bar{y_i}$ are all pairwise disjoint.*

*If each $\varphi_i$ is a $(\tau_0(\bar{x_i})+\tau_i)^*$-invariant, then $\bigwedge_{i=1}^{n} \varphi_i$ is a $(\tau_0+\cdots+\tau_n)^*$-invariant.* $\square$

*Example 3.* Let us consider the following three self-loops borrowed from the description of the Biphase protocol, which we will consider in Section 5:

$$\begin{cases} \tau_0 : x < max \wedge y < max & \rightarrow x := x + 1 \quad y := y + 1 \\ \tau_1 : x \geq min \wedge n < cell & \rightarrow x := 0 \quad n := n + 1 \\ \tau_2 : y \geq min \wedge m < sample & \rightarrow y := 0 \quad m := m + 1 \end{cases}$$

We can easily check that the premises of Lemma 4 are satisfied. Hence, we can split the analysis of the three self-loops into the independent analysis of the following sets each consisting of two self-loops, as shown below:

$$\begin{cases} \tau_0(x) : x < max & \rightarrow x := x + 1 \\ \tau_1 \quad\ : x \geq min \wedge n < cell \rightarrow x := 0 \quad n := n + 1 \end{cases}$$

$$\begin{cases} \tau_0(y) : y < max & \rightarrow y := y + 1 \\ \tau_2 \quad\ : y \geq min \wedge m < sample \rightarrow y := 0 \quad m := m + 1 \end{cases}$$

Each case can be analyzed independently using the results established in the previous section. We obtain that

$$\varphi_1 = (x \leq max \ \wedge \ n \leq cell)$$

is a $(\tau_0(x) + \tau_1)^*$-invariant and that

$$\varphi_2 = (y \leq max \ \wedge \ m \leq sample)$$

is a $(\tau_0(y) + \tau_2)^*$-invariant.

Thus, we can infer that

$$\varphi_1 \wedge \varphi_2 = (x \leq max \ \wedge \ n \leq cell \ \wedge \ y \leq max \ \wedge \ m \leq sample)$$

is a $(\tau_0 + \tau_1 + \tau_2)^*$-invariant. □

However, the invariants obtained in this way are too weak. The reason is that by the decomposition of the set of loops we lost the overall constraint induced on $\bar{x}$ variables by the $\tau_0$ loop. That is, all variables occurring in $\tau_0$ are strongly related by this transition, and it is no more the case when taking the projections. The following lemma solves this problem by adding some *re-synchronization* variables in order to be able to reconstruct (at least partially) the existing relation among the $\bar{x}$ variables.

**Lemma 5.** *Let $\tau_0, \tau_1, \cdots, \tau_n$ be transitions s.t. the premises of Lemma 4 are satisfied. Let $(z_i)_{i=1,n}$ be fresh variables and let $\tau_0'(\bar{x}_i)$ be the transition obtained from $\tau_0(\bar{x}_i)$ augmented with the assignment $z_i := z_i + 1$.*

*If each $\varphi_i'$ is a $(\tau_0'(\bar{x}_i) + \tau_i)^*$-invariant, then $\exists z_1, \cdots z_n.(z_1 = \cdots = z_n \wedge \bigwedge_{i=1}^{n} \varphi_i')$ is a $(\tau_0 + \cdots + \tau_n)^*$-invariant.* □

Intuitively, variables $z_i$ keep track of the number of times the transition $\tau_0$ is executed in each case. In this way, the global invariant can be strengthened by adding the equality on $z_i$ variables. That is, when considered together, the number of times $\tau_0$ is executed must be the same in all $1 \leq i \leq n$ cases.

*Example 4.* Let us consider again the three-loops presented above. After splitting them and augmentation with fresh variables $z_x$ and $z_y$, we obtain the following sets of self-loops to be analyzed:

$$\begin{cases} \tau_0(x) : x < max & \rightarrow x := x + 1 \quad z_x := z_x + 1 \\ \tau_1 \quad\;\; : x \geq min \wedge n < cell & \rightarrow x := 0 \quad n := n + 1 \end{cases}$$

$$\begin{cases} \tau_0(y) : y < max & \rightarrow y := y + 1 \quad z_y := z_y + 1 \\ \tau_2 \quad\;\; : y \geq min \wedge m < sample & \rightarrow y := 0 \quad m := m + 1 \end{cases}$$

Applying, Lemma 3, we obtain that

$$\varphi_1' = (x \leq max \ \wedge \ n \leq cell \ \wedge \ n \cdot min + x \leq z_x \leq n \cdot max + x)$$

is a $(\tau_0'(x) + \tau_1)^*$-invariant and that

$$\varphi_2' = (y \leq max \ \wedge \ m \leq sample \ \wedge \ m \cdot min + y \leq z_y \leq m \cdot max + y)$$

is a $(\tau_0'(y) + \tau_2)^*$-invariant.

The global invariant computed is then $\exists z_x, z_y.(z_x = z_y \wedge \varphi_1' \wedge \varphi_2')$, which can be simplified to

$$\begin{aligned} & x \leq max \ \wedge \ n \leq cell \ \wedge \ y \leq max \ \wedge \ m \leq sample \ \wedge \\ & n \cdot min + x \leq m \cdot max + y \ \wedge \ m \cdot min + y \leq n \cdot max + x. \end{aligned}$$

This invariant is indeed stronger than the one computed in Example 3. □

## 5 The Biphase protocol

The biphase mark protocol is a convention for representing both a string of bits and clock edges in a square wave. It is widely used in applications where data written by one device is read by another. It is for instance used in commercially available micro-controllers as the Intel 82530 Serial Communication Controller and in the Ethernet.

We borrow the following informal description of the protocol from J. S. Moore:

In the biphase mark protocol, each bit of messages is encoded in a *cell* which is logically divided into a *mark subcell* and a *code subcell.* During the mark subcell, the signal is held at the negation of its value at the end of the previous cell, providing an edge in the signal train which marks the beginning of the new cell. During the code subcell, the signal either returns to its previous value or does not, depending on whether the cell encodes a "1" or "0". The receiver is generally waiting for the edge that marks the arrival of a cell. When the edge is detected, the receiver counts off a fixed number of cycles, called *sampling distance*, and samples the signal there. The sampling distance is determined so as to make the receiver sample in the middle of the code subcell. If the sample is the same as the mark, a "0" was sent; otherwise a "1" was sent. The receiver takes up waiting for the next edge, thus *phase locking* onto the sender's clock.

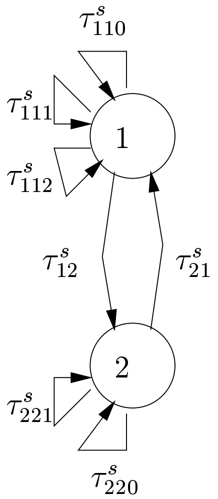
The main interesting aspect (from the verification point of view) of this protocol is the analysis of the tolerable asynchrony between the sender and the receiver. Put more directly, the derivation of sufficient conditions on the jitter between the clock of the sender and the clock of the receiver such that the protocol works properly.

To our knowledge, there has been some work on the verification of instances of the protocol either using theorem-proving techniques [Moo93] or model-checking [IG99,Vaa] and one work presenting full parameter analysis using PVS and the Duration Calculus, however, without clock jitter.

Using the techniques presented earlier in this paper, we have been able to fully analyze the protocol and to derive parameterized sufficient conditions for its correctness.

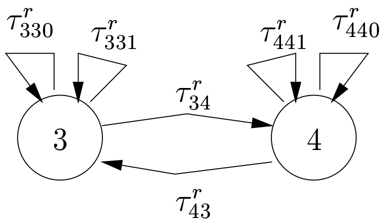
### 5.1 Protocol Modeling

We use extended transition systems to model the protocol which consists of a sender and a receiver exchanging boolean value. Some of the transitions are marked with synchronization labels. Following, Vaandrager we model the clock drifts and jitter using two different clocks which will be reset independently and using two parameters *min* and *max* to bound the drift between these clocks. The models of the sender, the receiver and their product are given in Figure 1, Figure 2, and Figure 3.




$$
\begin{array}{lll}
\tau^s_{110} : x < max & \rightarrow x := x + 1 & \\
\tau^s_{111} : x \geq min \ \wedge \ n < cell & \rightarrow x := 0 \quad n := n + 1 & \\
\tau^s_{112} : x \geq min \ \wedge \ n = cell & \rightarrow x := 0 \quad n := 0 \quad v := \neg v & \\
 & \quad \texttt{get} \ !false & \\
\tau^s_{12} \ \ : x \geq min \ \wedge \ n = cell & \rightarrow x := 0 \quad n := 0 \quad v := \neg v & \\
 & \quad \texttt{get} \ !true & \\
\tau^s_{220} : x < max & \rightarrow x := x + 1 & \\
\tau^s_{221} : x \geq min \ \wedge \ n < mark & \rightarrow x := 0 \quad n := n + 1 & \\
\tau^s_{21} \ \ : x \geq min \ \wedge \ n = mark & \rightarrow x := 0 \quad n := n + 1 \quad v := \neg v & \\
 & \quad \texttt{mark} &
\end{array}
$$

**Fig. 1.** The sender




$$
\begin{array}{ll}
\tau^r_{330} : y < max & \rightarrow y := y + 1 \\
\tau^s_{331} : y \geq min \ \wedge \ v = old & \rightarrow y := 0 \\
\tau^r_{34} \ \ : y \geq min \ \wedge \ v \neq old & \rightarrow y := 0 \\
 & \quad m := 0 \\
\tau^r_{440} : y < max & \rightarrow y := y + 1 \\
\tau^r_{441} : y \geq min \ \wedge \ m < sample & \rightarrow y := 0 \\
 & \quad m := m + 1 \\
\tau^r_{43} \ \ : y \geq min \ \wedge \ m = sample & \rightarrow y := 0 \\
 & \quad \texttt{put} \ !v \neq old
\end{array}
$$

**Fig. 2.** The receiver

$\tau_{130}, \tau_{140}, \tau_{230}, \tau_{240} : x < max \ \wedge \ y < max \rightarrow x := x + 1 \ \ y := y + 1$

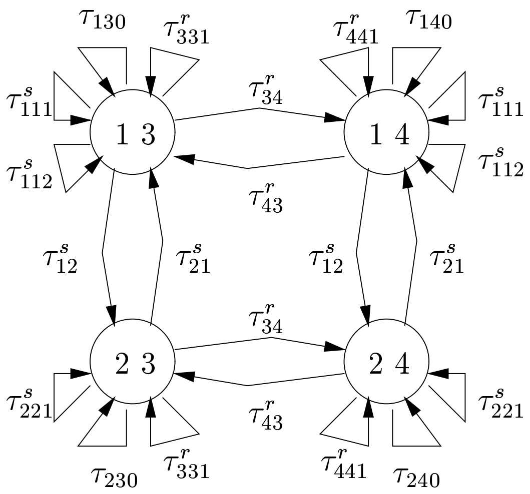



**Fig. 3.** The product

### 5.2 Invariant generation

Using the techniques presented before we are able to construct the following invariants for the product control locations:

$$
\begin{aligned}
\varphi_{13} &= x \leq max \ \wedge \ y \leq max \ \wedge \ n \leq cell \\
\varphi_{14} &= x \leq max \ \wedge \ y \leq max \ \wedge \ n \leq cell \ \wedge \ m \leq sample \\
&\quad m \cdot min + y \leq n \cdot max + x \ \wedge \ n \cdot min + x \leq m \cdot max + y \\
\varphi_{23} &= x \leq max \ \wedge \ y \leq max \ \wedge \ n \leq mark \\
\varphi_{24} &= x \leq max \ \wedge \ y \leq max \ \wedge \ n \leq mark \ \wedge \ m \leq sample \\
&\quad m \cdot min + y \leq n \cdot max + x \ \wedge \ n \cdot min + x \leq m \cdot max + y
\end{aligned}
$$

### 5.3 Parameter synthesis

One of requirements for correctness of the protocol states that *the receiver does not sample too late*. That is, a bad behavior is obtained by allowing to take two consecutive `get` actions by the protocol, without no `put` action in between. For instance, such a scenario is possible when in state 14, the `get` transitions $\tau^s_{112}$ or $\tau^s_{12}$ are enabled before the `put` transition $\tau^r_{43}$. To avoid such a situation, a sufficient condition will be if $\varphi_{14} \wedge (gua(\tau^s_{112}) \vee gua(\tau^s_{12}))$ is not satisfiable. This condition is the following:

$$
\begin{gathered}
x \leq max \ \wedge \ y \leq max \ \wedge \ n \leq cell \ \wedge \ m \leq sample \\
m \cdot min + y \leq n \cdot max + x \ \wedge \ n \cdot min + x \leq m \cdot max + y \\
x \geq min \ \wedge \ n = cell
\end{gathered}
$$

and is equivalent after simplification to :

$$(cell + 1) \cdot min > (sample + 1) \cdot max$$

A second requirement states that *the receiver does not sample too early.* That is, wrong behavior occurs when the receiver samples before the mark sub-cell started. In this case, a bad scenario is that one in state 24 the `put` transition $\tau^r_{43}$ is enabled before the `mark` transition $\tau^s_{21}$. Here also, this behavior can be avoided if the condition $\varphi_{24} \ \wedge \ gua(\tau^r_{43})$ is not satisfiable. We obtained in this case:

$$
\begin{gathered}
x \leq max \ \wedge \ y \leq max \ \wedge \ n \leq mark \ \wedge \ m \leq sample \\
m \cdot min + y \leq n \cdot max + x \ \wedge \ n \cdot min + x \leq m \cdot max + y \\
y \geq min \ \wedge \ m = sample
\end{gathered}
$$

and can be further simplified to the following condition depending only on parameters:

$$(sample + 1) \cdot min > (mark + 1) \cdot max$$

## 6 Conclusions

In this paper, we presented a set of techniques which allow to compute an over-approximation of the set of reachable states of a set of self-loops. The techniques we presented can be partitioned in two classes: 1.) exact techniques that under effectively checkable conditions allow to characterize the set of reachable states of pairs of self-loops without loss of information and 2.) techniques that allow to reduce more general cases of a set of self-loops to the analysis of a set of pairs of self-loops. Using, our techniques we have been able to synthesize a set of conditions on the parameters of the Biphase protocol that are sufficient to ensure its correctness.

We plan to implement our techniques using decision procedures for Presburger arithmetic to decide the conditions necessary for applying them. We also plan to apply these techniques for generating test cases for protocols and test objectives that involve data.

## References


[ABJ98] P. Abdulla, A. Bouajjani, and B. Jonsson. On-the-fly analysis of systems with unbounded, lossy fifo channels. In *CAV'98*, volume 1427 of *LNCS*, pages 305–318, 1998.

[ABJN99] P.A. Abdulla, A. Bouajjani, B. Jonsson, and M. Nilsson. Handling Global Conditions in Parameterized System Verification. In N. Halbwachs and D. Peled, editors, *CAV '99*, volume 1633 of *LNCS*, pages 134–145. Springer-Verlag, 1999.

[AD94] R. Alur and D. Dill. A Theory of Timed Automata. *Theoretical Computer Science*, 126, 1994.

[AM80] K.R. Apt and L.G.L.T. Meertens. Completeness with finite systems of intermediate assertions for recursive program schemes. *SIAM J. Comp.*, 9:665–671, 1980.

[BBLS] K. Baukus, S. Bensalem, Y. Lakhnech, and K. Stahl. Abstracting ws1s systems to verify parameterized networks. In *TACAS'00.*

[BG96] B. Boigelot and P. Godefroid. Symbolic verification of communication protocols with infinite state spaces using QDDs. In *CAV'96*, volume 1102 of *LNCS*, pages 1–12, 1996.

[BGP97] Bultan, Gerber, and Pugh. Symbolic model checking of infinite state systems using presburger arithmetic. In *CAV: International Conference on Computer Aided Verification*, 1997.

[CC76] P. Cousot and R. Cousot. Static determination of dynamic properties of programs. In *Proc. 2nd Int. Symp. on Programming*, pages 106–130, 1976.

[CC77] P. Cousot and R. Cousot. Abstract interpretation: A unified lattice model for static analysis of programs by construction or approximation of fixpoints. In *4th ACM symp. of Prog. Lang.*, pages 238–252. ACM Press, 1977.

[CC92] P. Cousot and R. Cousot. Abstract interpretation frameworks. *J. Logic and Comp.*, 2(4):511–547, 1992.

[CH78] P. Cousot and N. Halbwachs. Automatic discovery of linear restraints among the variables of a program. In *5th ACM symp. of Prog. Lang.*, pages 84–97. ACM Press, 1978.

[CHPP87] P. Caspi, N. Halbwachs, D. Pilaud, and J. Plaice. LUSTRE, adclarative language for programming synchronous systems. In *14th Symposium on Principles of Programming Langiages*, 1987.

[FO97] L. Fribourg and H. Olsèn. A decompositional approach for computing least fized-points of datalog programs with z-counters. *Constraints*, 2(3/4):305–335, 1997.

[Gor75] G. A. Gorelick. A complete axiomatic system for proving assertions about recursive and non-recursive programs. Technical report, Toronto, 1975.

[HNSY92] T.A. Henzinger, X. Nicollin, J. Sifakis, and S. Yovine. Symbolic model-checking for real-time systems. In *Seventh Annual IEEE Symposium on Logic in Computer Science*, pages 394–406. IEEE Computer Society Press, 1992.

[HPR94] N. Halbwachs, Y.-E. Proy, and P. Raymond. Verification of linear hybrid systems by means of convex approximations. In *Proceedings of the International Symposium on Static Analysis*, volume 818 of *LNCS*, pages 223–237. Springer-Verlag, 1994.

[IG99] S. Ivanov and W. O. D. Griffioen. Verification of a biphase mark protocol. Report CSI-R9915, Computing Science Institute, University of Nijmegen, August 1999.

[Kar76] M. Karr. Affine relationships among variables of a program. *Acta Informatica*, 6:133–151, 1976.

[KJ89] K.M. Chandy and J. Misra. *Parallel Program Design.* Addison-Wesley, Austin, Texas, May 1989.

[KMM$^+$97] Y. Kesten, O. Maler, M. Marcus, A. Pnueli, and E. Shahar. Symbolic Model Checking with Rich Assertional Languages. In O. Grumberg, editor, *Proceedings of CAV '97*, volume 1256 of *LNCS*, pages 424–435. Springer-Verlag, 1997.

[Moo93] J. S. Moore. A formal model of asynchronous communication and its use in mechanically verifying a biphase mark protocol. *Formal Aspects of Computing*, 3(1), 1993.

[Vaa] F. Vaandrager. Analysis of a biphase mark protocol with uppaal. Presentation at the meeting of the VHS-ESPRIT Project.

[WB98] P. Wolper and B. Boigelot. Verifying systems with infinite but regular state spaces. In *CAV'98*, volume 1427 of *LNCS*, pages 88–97, 1998.